1-byte instr^n
2-byte instr^n
3-byte instr^n

8085 Microprocessor Instruction Set Quick Reference

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		

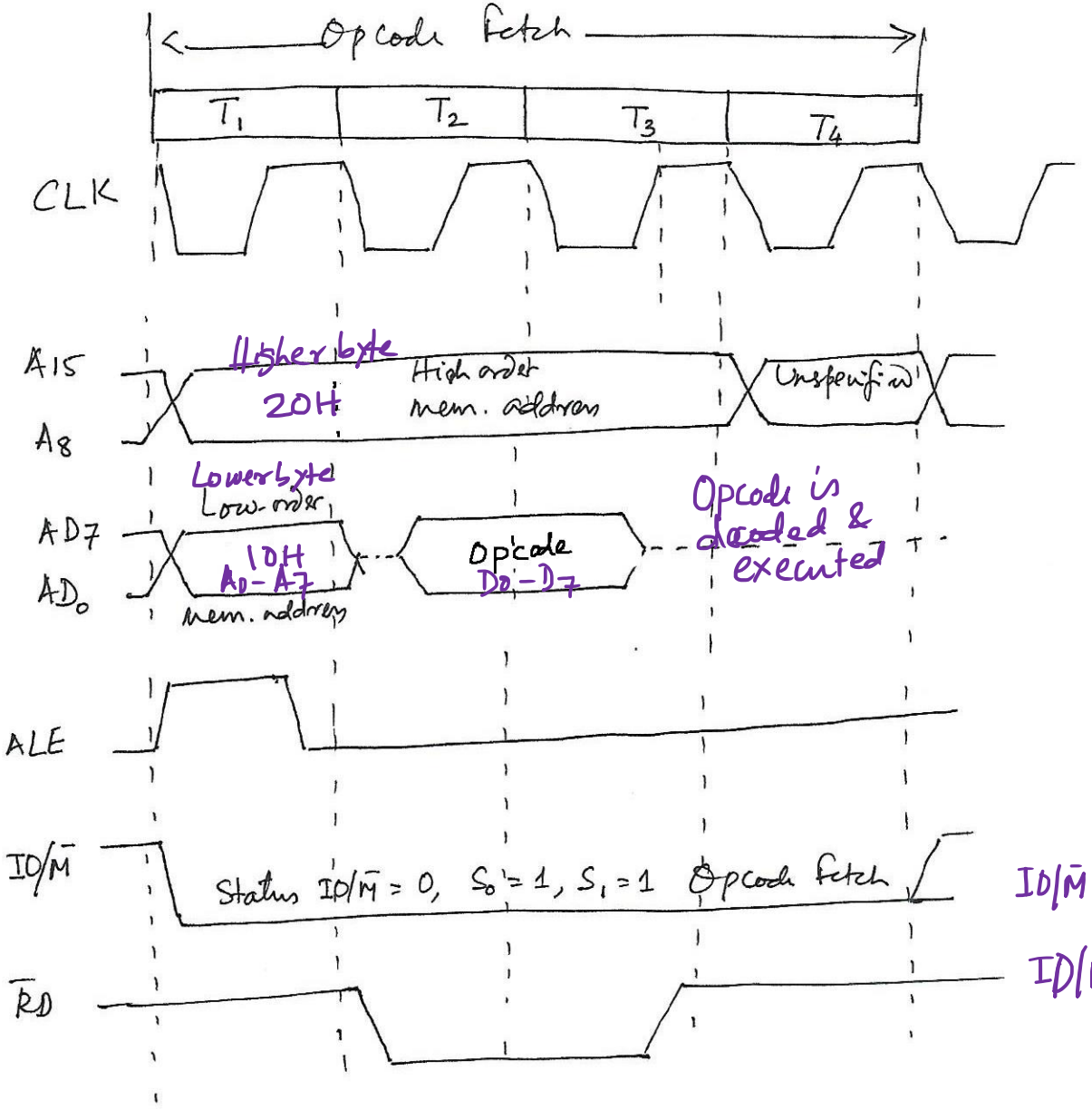
Machine Cycles in 8085

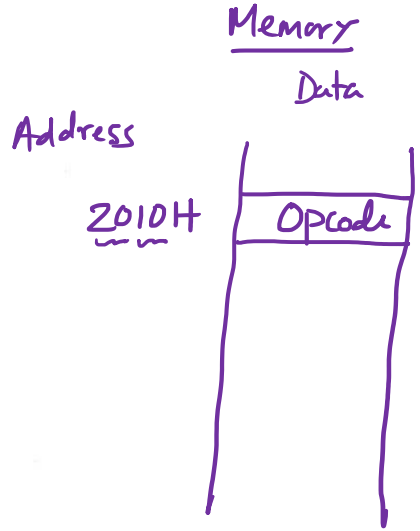
How does 8085 execute an instruction ?

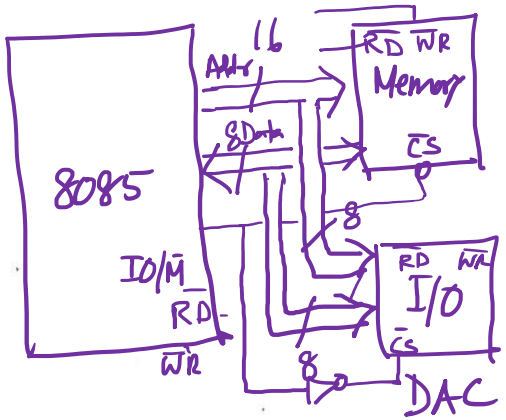
8085 takes a few machine cycles) to execute an instruction.

1. OpCode Fetch (OF) : Fetches the opcode from memory and decodes/executes it : Takes 4 clock cycles (4T)
2. Memory Read (MR) : Reads a byte from the memory : Takes 3 clock cycles (3T)
3. Memory Write (MW) : Writes a byte to the memory : Takes 3 clock cycles (3T)
4. I/O Read (IOR) : Reads a byte from an I/O device : Takes 3 clock cycles (3T)
5. I/O Write (IOW) : Writes a byte to an I/O device : Takes 3 clock cycles (3T)
6. Interrupt Acknowledge (IA) : Acknowledges an interrupt
7. Bus Idle (BI) : Does some internal operation while the data/address Buses remain idle.

Opcode Fetch Machine Cycle



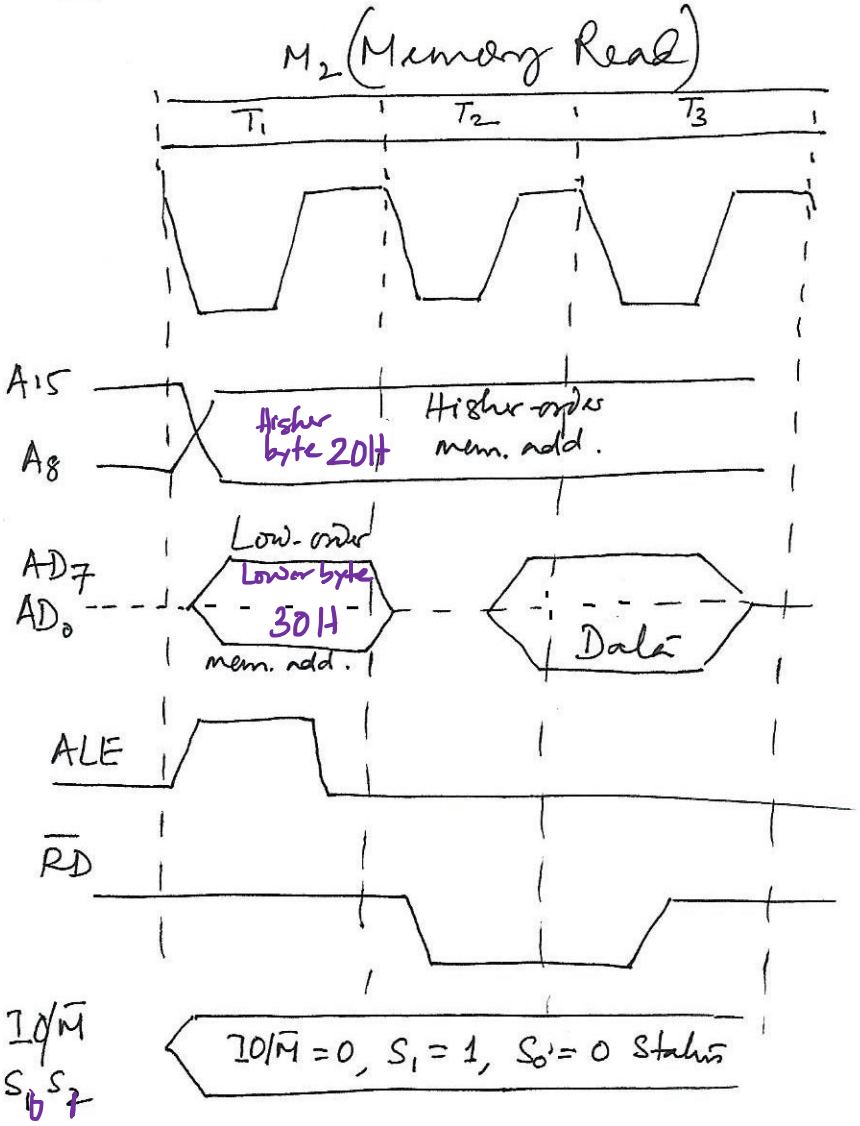


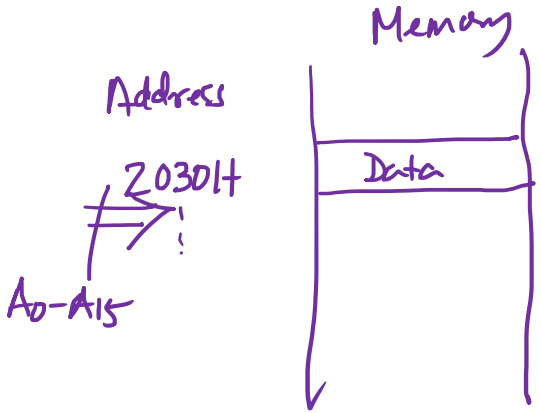


IO/M = 0 00 10H Memory address

IO/M = 1 10H I/O address

Memory Read Machine Cycle





Mnemonic	Number of bytes	Machine cycles	Number of Clock cycles
MOV A, B	1	OF	4
MVI A, 32H	2	OF, MR	4+3 = 7
MOV A, M ← HL	1	OF, MR	4+3 = 7
LDA 4000H	3	OF, MR, MR, MR	4+3+3+3 = 13
STA 4020H	3	OF, MR, MR, MW	4+3+3+3 = 13
IN 40H	2	OF, MR, IOR	4+3+3 = 10

Interrupts

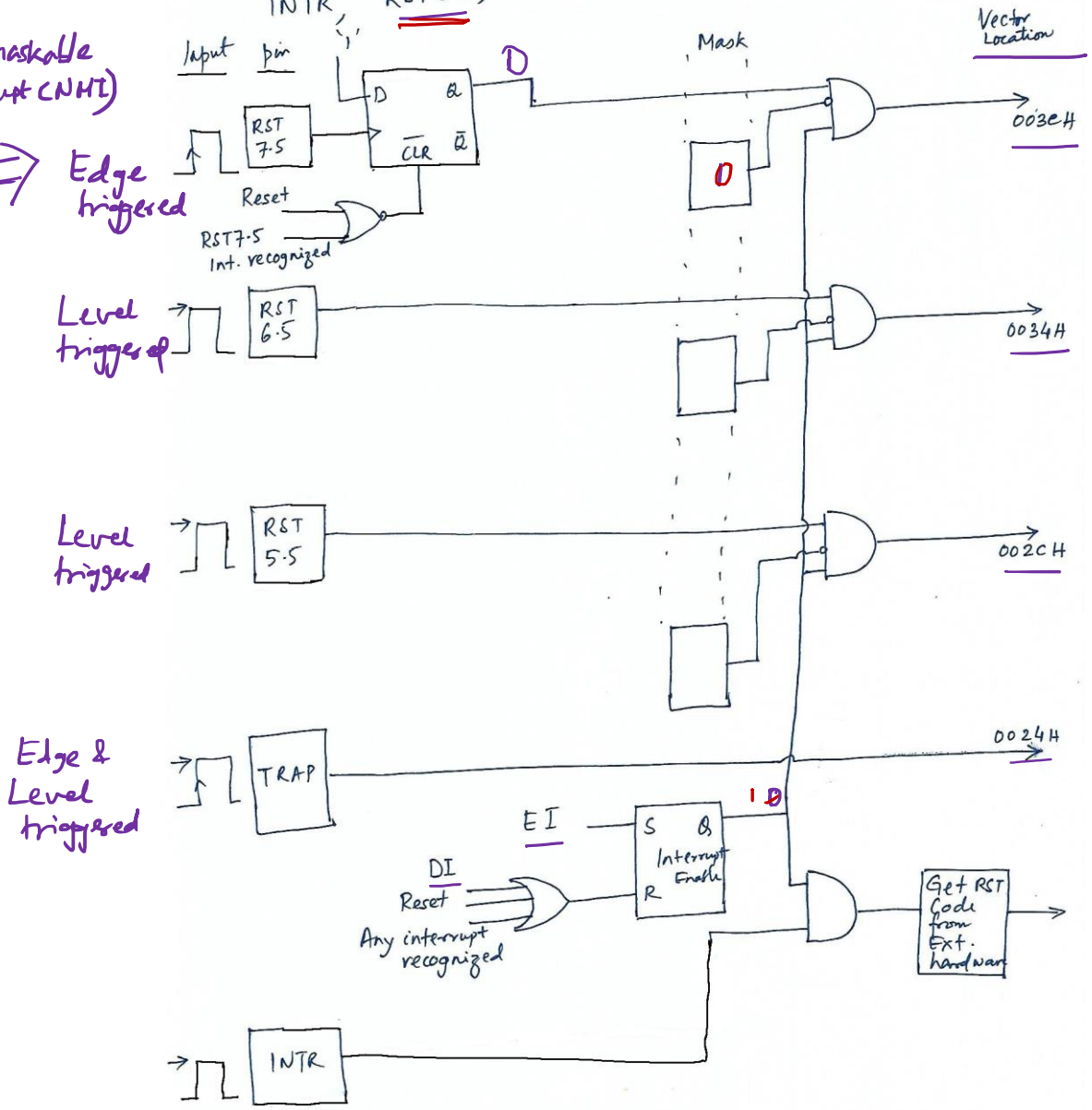
There are 5 hardware interrupts in 8085.

INTR, RST5.5, RST6.5, RST7.5 and TRAP

Priority

1. TRAP - Nonmaskable Interrupt (NMI)
2. RST7.5
3. RST6.5
4. RST5.5
5. INTR

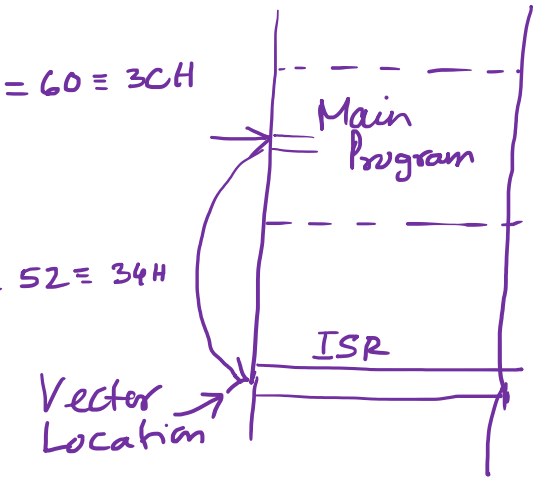
Schematic



7.5x8 = 60 ≡ 3CH

6.5x8 = 52 ≡ 34H

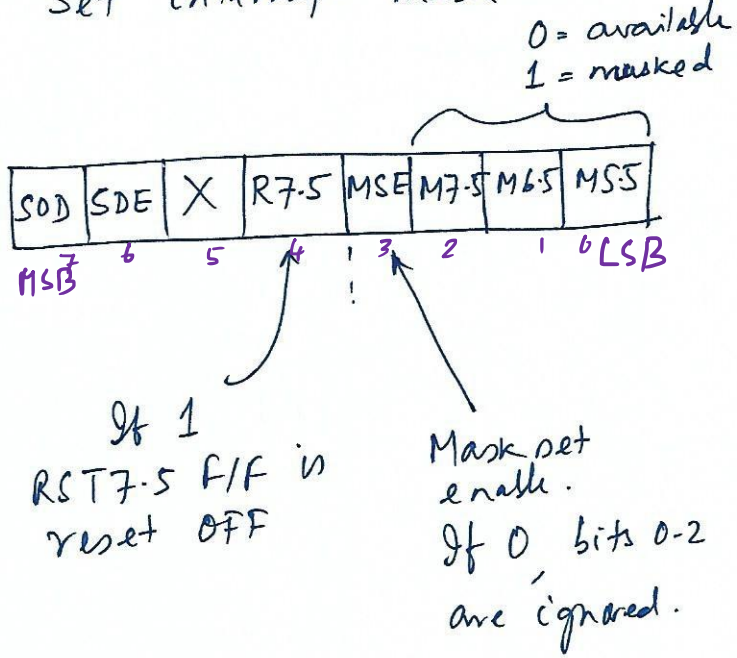
5.5x8 = 44 ≡ 2CH



NMI

SIM

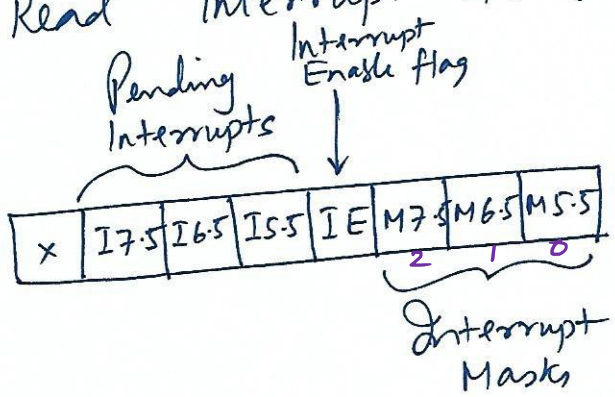
Set interrupt mask



Content of A

RIM

Read Interrupt Mask



Content of A

EI : Enable Interrupt

- Sets the int. enable F/F and enables int. process
- System reset or an interrupt disables the int. process.

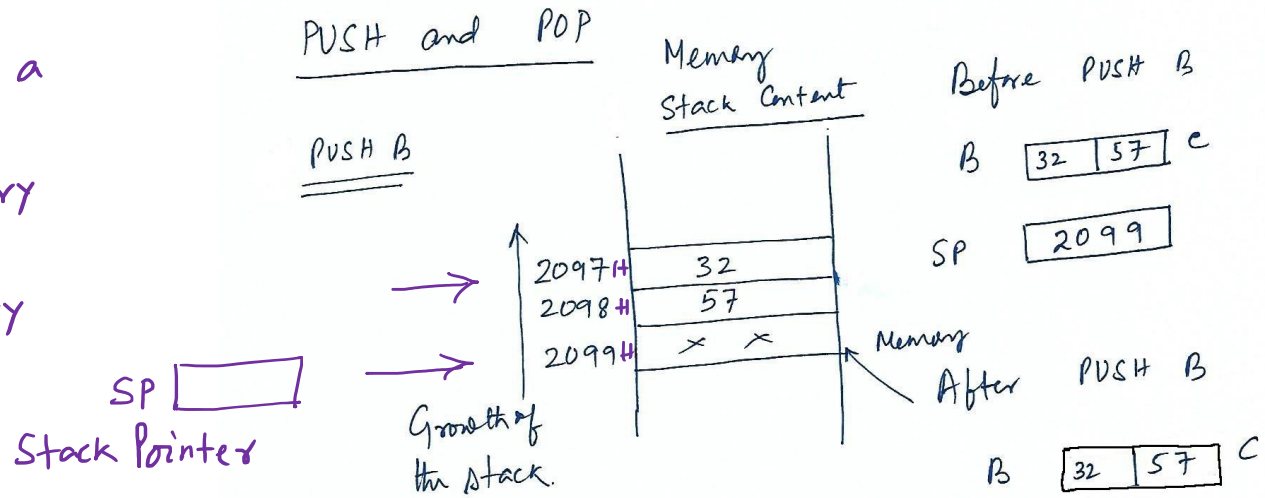
DI : Disable Interrupt.

- Resets the int. enable F/F and disables the interrupt.

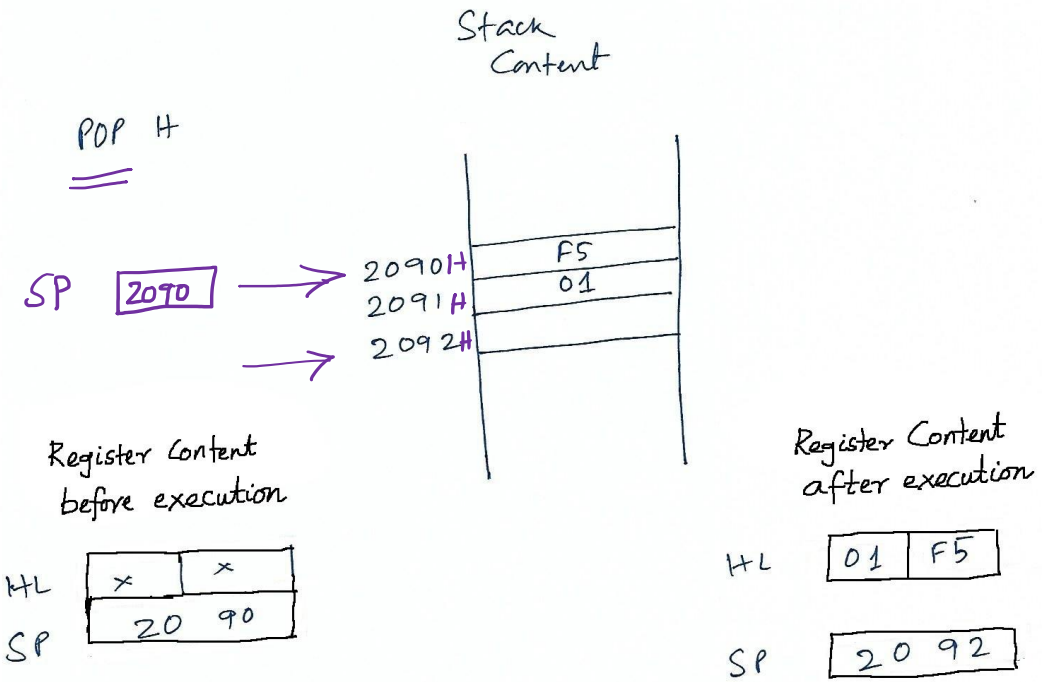
Stack is a part of RAM memory used for temporary storage.

PUSH and POP

PUSH B



POP H



System Memory Map (8085 Kit)

Device	Address Range	
27128 at U1 (EPROM)	0000 - 3FFF H	16K byte
62256 at U3 (RAM)	8000 - FFFF H	32K byte

Memory from 8F90H to 8FFFH is utilized by the system and rest of the RAM is available to the user.

Keyboard Interrupt

KBINT → RST 7.5 → 8FBFH (Jump Location)

7.5 × 8 = 60 = 003CH → JMP 8FBFH

8FBFH ↖ Location for Interrupt Service Routine

PART 1 : 8085 PROGRAMMING USING HEX KEY PAD

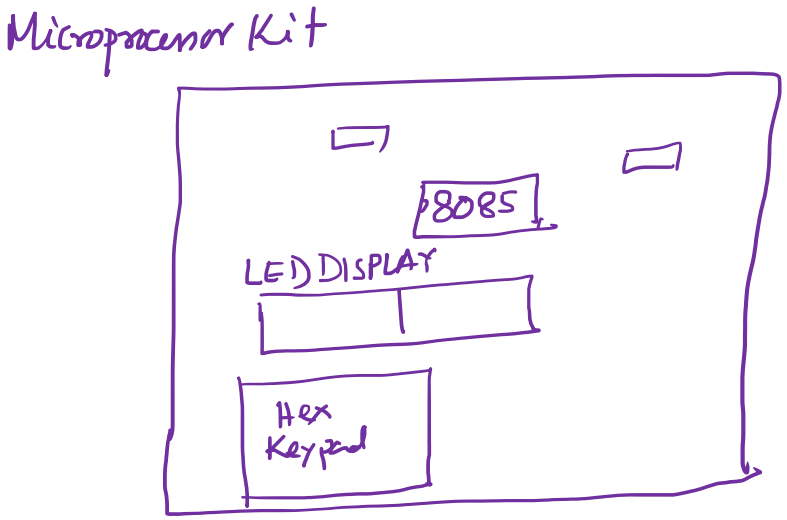
(Switch 4 of DIP switch in OFF mode)

A) FAMILIARIZATION

Familiarize yourself with the following Monitor Commands of the 8085 kit.

EXAMINE/MODIFY MEMORY
EXAMINE/MODIFY REGISTER
SINGLE STEP
GO

The above Monitor commands are required for operations using the Hex Keypad. If any wrong key is pressed, the LED Display will show “Err”. You will need to press the RESET key to get back to normal mode.



B) PROGRAMMING

Program 1

NORMAL EXECUTION

(a) Write a program to find the largest of the 10 numbers stored in locations 8040H to 8049H. Your output of the program (largest number) should be stored in 804AH.

(b) You will do the coding part (i.e. 8085 mnemonics to Hex code) for this program manually. Enter your assembly language program in a tabular form. Instruction set and opcodes are provided in one of the attached sheets. The table should have the following columns:

Memory address	Hex Opcode	Assembly Language Instructions		Comments/Remarks
		Label(if any)	8085 Mnemonics with operands	
8000 H	78 H		MOV A, B	

The Origin of your program should be 8000H. Enter some data in locations 8040H to 8049H. Ensure that the initial data in 804AH is 00H.

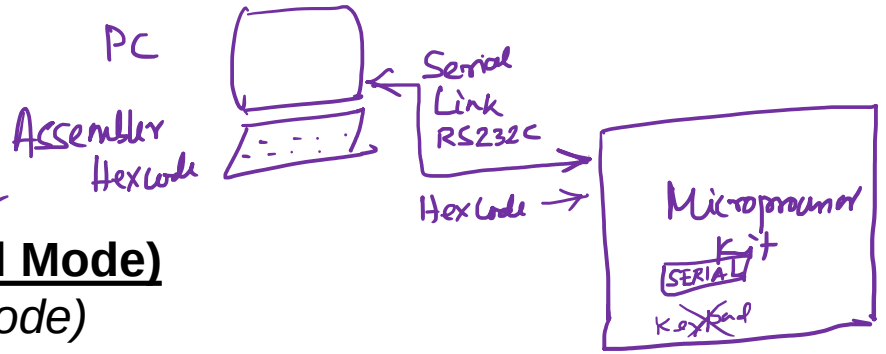
(c) Execute your program using GO command from the Hex keypad.

(d) Check whether the largest number has been stored in 804AH.

SINGLE STEPPING MODE

(a) Once again write 00H in location 804AH.

Use Single Step mode and check your program for two or three loops of the program to make yourselves familiar with this mode. Note that you can examine/modify registers as well as memory locations in between the single stepping operations.



PART 2 : MPS 85-3 KIT PROGRAMMING USING PC(Serial Mode)

(*Switch 4 of DIP switch in ON mode)*

A) FAMILIARIZATION

Familiarize yourself with the following Monitor Commands of the 8085 kit in the Serial mode.

D	: display memory command;
G	: command to execute a program
S	: Substitute memory command;
X	: Examine/modify register command

The above Monitor commands are frequently required for serial operations of the kit.

Command in the utility program XT853.EXE to download a .HEX file: Ctrl+ D

Use of Assembler

B) PROGRAMMING

Program 1

(a) Write your program for finding the largest number as a .TXT file (use NOTEPAD) in the C:\8085\PGM folder. Copy your file to the C:\8085\ASM folder.

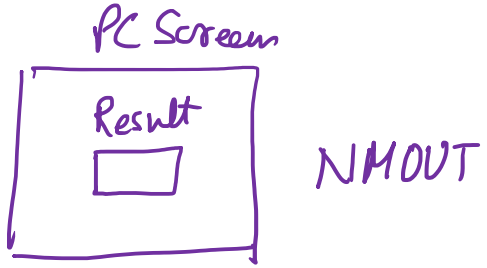
(b) Assemble your file. See the procedure mentioned under the “Assembling” section in the attached sheets. The cross assembler will generate .LST and .OBJ files.

(c) The .LST file gives the original assembly language program, hex codes, and label addresses. Open this file using Notepad and see the opcodes and addresses generated for each line. Check whether the opcodes in this file are the same as you wrote down manually under Part 1.

(d) Viewing the .HEX file: Use the Hex Editor XVI32.EXE in the C:\8085\HEXEDIT folder to view all the hex codes in your .OBJ file. Note the extra characters due to the Intel Hex format. Also notice 0D and 0A characters (CR and LF) at the end of every line.

(e) Download your file to the 8085 kit (after renaming it to .HEX).

(f) Execute it and check that you are getting the result as in Part 1.



Modification to Program 1

(a) Make a small modification to the above program such that the result is displayed on the PC screen. You can use the Serial Monitor routine NMOUT for this purpose.

(b) Assemble the modified program, download and execute the same. Check whether the result is displayed on the screen.

Program 2 (Use of RST7.5 Interrupt)

RST7.5 interrupt of the 8085 Microprocessor is provided on the kit in the Hex Keypad – KBINT. Write an RST7.5 ISR which will increment the accumulator each time KBINT is pressed (initial value of A=00H)and display the current value of A on the PC Screen. You will need to use SIM and EI instructions in a main program which runs in an infinite loop. Note that when RST7.5 interrupt comes the monitor will go to the RAM location 8FBFH location. Again, you will need to store a JUMP instruction in 8FBFH location to your ISR subroutine (it is better to do this step by hand assembly). Use the A85.EXE Cross-assembler for the Main program and the ISR.