Interrupts It is an internal or external signal which may disturb or alter the sequence of execution of a process. Interrupts are classified as ① Hardware & Software
② Maskable & Non maskable
③ Vectored & Non vectored.

① Hardware Interrupts There are 5 hardware interrupts TRAP to INTR (according to priority). They must be physically connected to a practical application.

Software Interrupts There are 8 software interrupts which can be used either as instruction or along with INTR (RST0 – RST7).

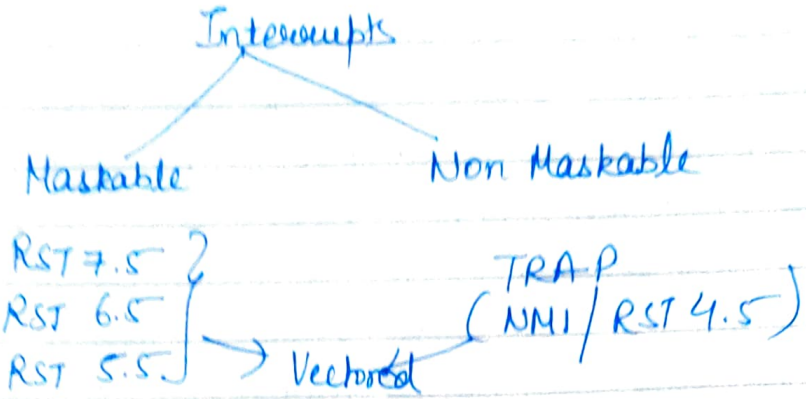
Maskable Interrupts Interrupts which can be ignored, neglected or avoided even they are triggered.

Non Maskable Interrupts Interrupts which cannot be ignored or avoided when they are triggered. TRAP is the only non-maskable interrupt in 8085. It must be connected to highly priortized event in practical application. It is also known as NMI.

Vectored Interrupts Interrupts which have specific address location

Non Vectored Interrupts Interrupts which do not have specific address location.

NOTE INTR is the only non vectored interrupt in 8085.

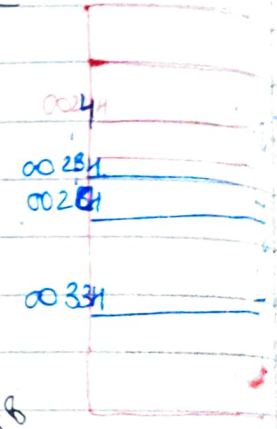


INTR ⇒ Non Vectored

Imp → (1 Question)

ISR → Interrupt Service Routine

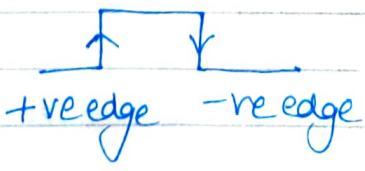
Priority	Interrupts	Types of Triggering	Address Location
1	TRAP (NMI/RST4.5)	→	0024 H
2	RST 7.5		003C H
3	RST 6.5	→	0034 H
4	RST 5.5	→	002C H
5	INTR	→	



7.5 x 8

① Level Triggering

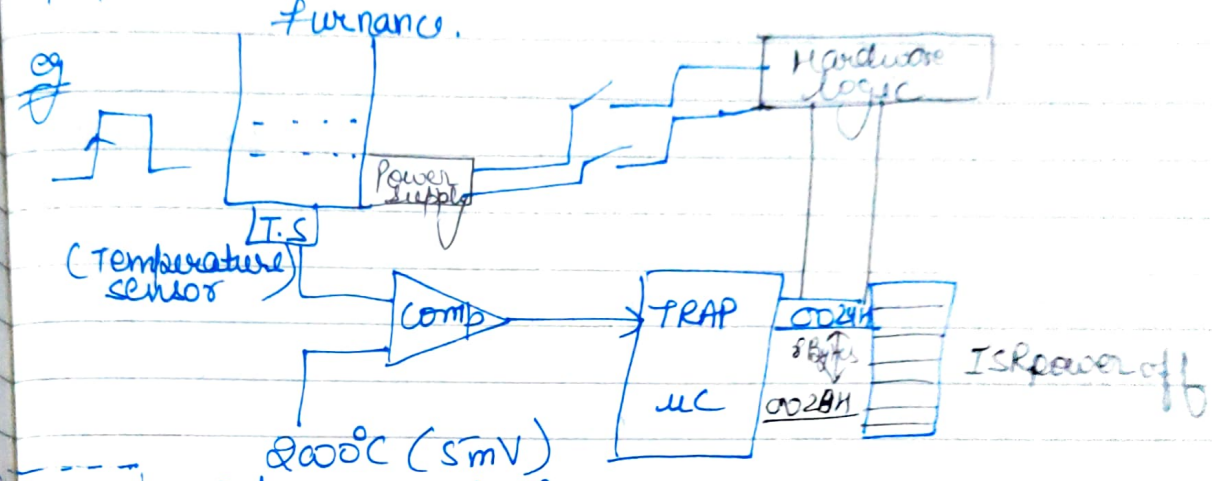
② Edge



⇒ Every vectored interrupt in 8085 is alloted 8 bytes in the memory in order to store the relevant program that must be ~~stored~~ executed in response to an interrupt which is known as interrupt service routine (ISR)

If ISR is not sufficient in 8 bytes, an unconditional jump instruction can be included such that program may be continued.

26/10/13



Tolerance ± 100°C

TRAP, 5.5 — If trap is only edge triggered, then if a spike of noise will come then TRAP will be affected.

So, to avoid this, TRAP is edge & level triggered & the pulse should be high for at least 3 clock cycles to detect at TRAP

⇒ TRAP is both edge and level triggered. It is edge triggered such that it may be responded quickly. It is level triggered in order to differentiate original signal from practical application & error signal due to noise. The signal on TRAP pin should be high for at least 3 clock periods in order to avoid error signal due to noise.

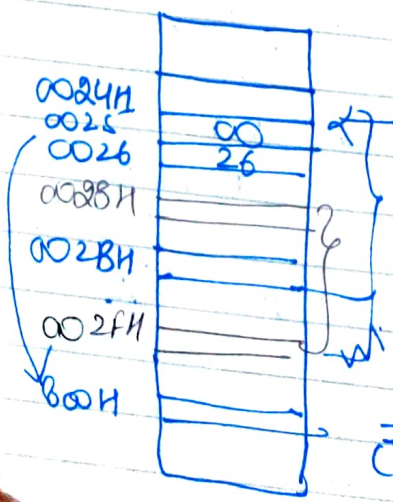
⇒ Every vectored interrupt in 8085 is given 8 bytes in the memory to store the corresponding

program that must be executed in response to an interrupt which is known as interrupt service routine.

⇒ If ISR is not sufficient 8 bytes an unconditional jump instruction can be included such that the program may be continued further.

⇒ When both hardware & software interrupts are to be used proper programming logic must be implemented to avoid errors

RST 0 →	0000H	– 0007H	64KB (Data, program, ISR)
RST 0.5 →	0004H		
RST 1 →	0008H	– 000FH	
RST 1.5 →	000CH		
RST 2 →	0010H	– 0017H	Assumption
RST 2.5 →	0014H		
RST 3 →	0018H	– 001FH	
RST 3.5 →	001CH		
RST 4 →	0020H	– 0027H	TRAP
RST 4.5 →	0024H		
RST 5 →	0028H	– 002FH	
RST 5.5 →	002CH		
RST 6 →	0030H	– 0037H	Hardware interrupts
RST 6.5 →	0034H		
RST 7 →	0038H	– 003FH	
RST 7.5 →	003CH		



Hardware interrupts ~~are~~ not expected but software interrupt care expected.

Vector address calculation for interrupts,

RSTn → 0 to 7

$$n \times 8 = (X)_{10} = (y)_{16}$$

eg. RST 6

$6 \times 8 = (48)_{10} = (0030H)_{16}$

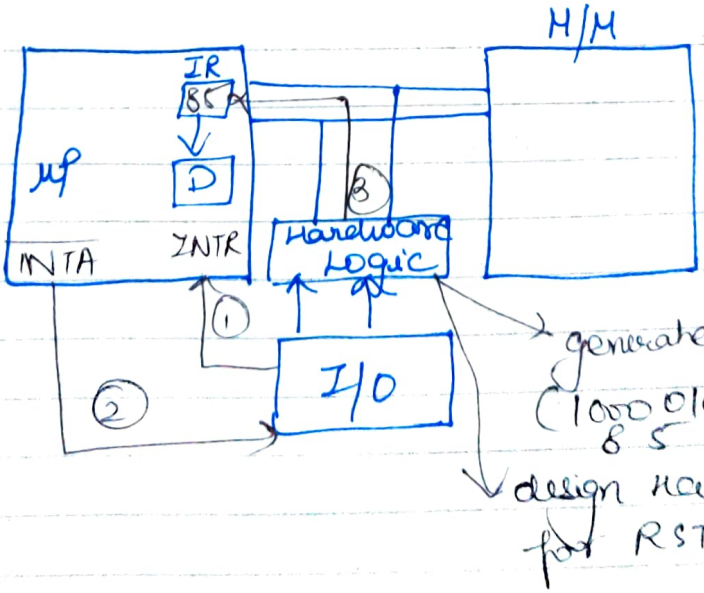
RST 7.5

$\Rightarrow 7.5 \times 8 = (60.0)_{10}$

$= (003C)_{16}$

16|60 (2)
3 (C)

INTR → Hardware interrupt but we can't use the vector addres of software interrupts from RST0 - RST7.



RST 5 → 8

INTR is non vectored interrupt because it goes to any vector address from RST0 - RST 7 for which hardware logic is designed.

"Only for INTR, $\overline{INTA}$ is used"

⇒ In order to use INTR, the programmer must select vector addresses of one of the software interrupts to store ISR for the I/O device.

⇒ The external hardware logic can be designed to produce or generate the opcode of selected RSTn.

⇒ When a processor receives INTR, it responds with interrupt acknowledgement & wait for the response from I/O.

⇒ The external hardware logic provide the opcode which is accessed into instruction register, decoded & control of the program is transferred to vector address of RSTn, as there is no specific address for INTR, it is known as non vectored interrupt.

NOTE $\overline{INTA}$ is required only for INTR not for vectored interrupts.

Instructions Related to Interrupts

EI → Enable interrupts
DI → Disable interrupts
SIM → Set interrupt Mask
} only for Maskable interrupts.

SIM is used to mask the interrupts or make them available.

RIM, Read interrup Mask.
It is used to know status of pending interrupts

NOTE TRAP cannot be enabled or disabled.

Interrupts → objective / conventional (both)

SIM & RIM are only valid for RST 7.5, RST 6.5, RST 5.5

Programming Model

Describe the programming model for 8085. (16 marks)
(write GPR, SPR, PC, IR, SP)

Program Set of instructions

Instruction is a command given to a computer to perform specific task.

(low level languages which are machine dependent)

Machine Level Language: It is a binary medium of comm. with a computer through a designed set of instructions specific to a system.

Assembly Level Language Intructions are written in separate words known as Mnemonics which are partially understood by the programmer.

MOV A, B

↓

Mnemonics

Add C (add the contents of C to accumulator)

Both assembly & machine level languages are together known as low level languages.

High Level Language Instructions or statements are written in English like words. High level language is machine independent.

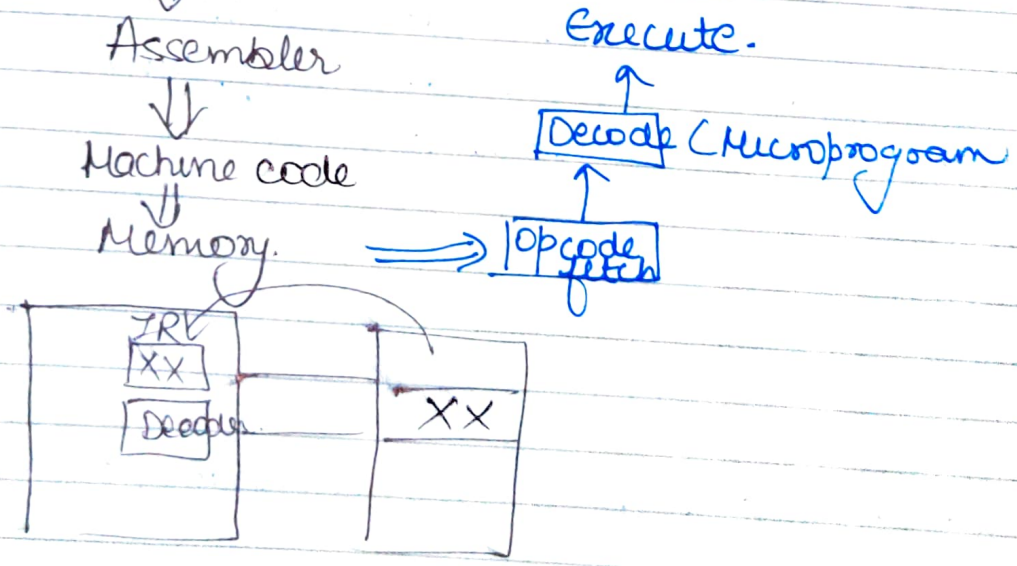
eg, C, C++, JAVA

Compiler Software which converts high level language to machine level language where entire program is converted at a time
eg. TURBO-C, ~~EXCEL-C~~ XLC, Java C.

Intesperter Software which converts high level language to machine level language line by line.
M-
eg → BASIC

Assembler converts assembly level language to machine code.
eg. MASM → Macro assembler.

Assemble Program
↓
Assembler
↓
Machine code
↓
Memory.



Basic steps required for the execution of instruction
1. Opcode fetch
2. Decode
3. Execute.

Explain the steps involved for execution for instruct

Describe the programming model of 8085

① GPR , ② SPR → (SP, IR, PC, FR)

Instruction Format

Opcode	Operands

Opcode : Operation code
It indicates the type of operation to be performed for an instruction.

Operands It is the data on which the operation is to be performed.
Operand can be Reg / Reg pair / 8 bit data or Address / 16 bit data/address.

① MOV B, C → 1 byte. → XX
MOV → opcode, B, C → operands

② MVI A, 66H → YY, 66H.
MVI → Move immediate. → opcode.
A → operand
YY → opcode.
66H

③ LXI H, 3456H → ZZ, 56H, 34H
LXI H → opcode
3456H → operand. 2 Bytes

Length of instruction

No. of bytes an intruction occupies in the memory

There are 3 types of instruction in 8085 classified according to length.

① 1 Byte / word Instruction
MOV B, C

② 2 Byte / word "
MVI A, 66H

③ 3 Byte / word "
LXI H, 3456H.

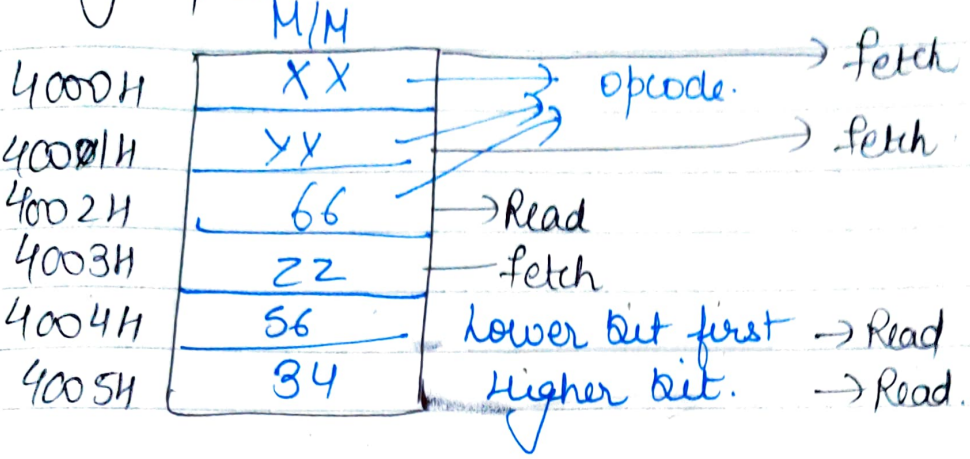
eg. ① RET — 1 bytes

② OUT 60H — 2 Bytes

③ LDA 1234H → 3 bytes.

4000H	MOV B, C	Total program 6 Bytes
4001H	MVI A, 66H.	
4002	(66H)	
4003H	LXI H, 3456H	
4004	(56H)	
4005	(34H)	

Memory Representation



eg STA 2546H.
F R R

eg. IN 20H by knowing length we get, F R ← min. operation.

Memory Rule In all memory related operation, the data present in lower byte or register is transferred to lower address location. Similarly, higher byte of register is transferred to higher address location or vice versa.

	Code
B	000
C	001
D	010
E	011
H	100
L	101
A	111
M/M Related	110

BC — 00
DE — 01
HL — 10
SP — 11

① MOV Rd, Rs
01 DDD SSS

M/M

4000H. 41H.

Eg. MOV B, C

↓

Assembler.

01 000 001

4 1 H.

MOV A, M

↓

Opcode

01 111 110

7 E H.

Eg MOV A, A

MOV A, B

A, C

:

MOV B, A

B, B

:

MOV M, M XX ← not exist.

62 + 8.

ADD R

10 000 RRR.

Ex ADD E

↓

Assembler

10000 011

83 H

MVI R / 8 bit data

LXI Rp

74

Every register is given unique code. There are 74 different opcodes which result in 246 instructions in 8085.

Addressing Modes

These are various formats specifying the operands or data or they indicate how data is accessed for an instruction.

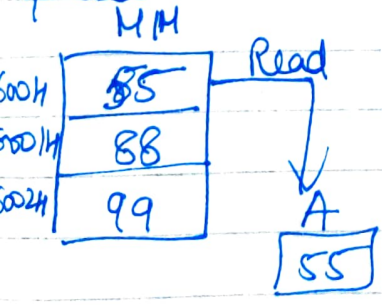
There are 5 types of addressing modes:

1] Immediate addressing mode.
2] Direct addressing mode
3] Indirect addressing mode
4] Register addressing mode
5] Implicit " "

Immediate adressing mode In this type, data is present in the instruction itself.

eg. MVI B, 77H
↓
Data

Direct Addressing Mode Address of the data is present in the instruction.

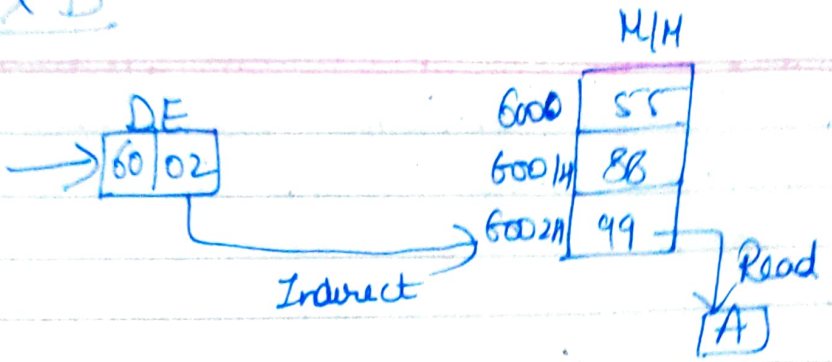


LDA 6000H

Indirect Addressing mode. Address of the data is present as content another register pair

LDA X D.
↑ operated on Register pair (DE)

LDA x D.



4) Register Addressing Mode is Data transfer b/w registers.

MOV A, D

5] Implicit Addressing modes No operands or data is present in instruction but some kind of operation is performed by the instruction.

eg. CMA :- complement Accumulator.
Operation may be performed in a single register

Timing Diagram

Pictorial representation of execution of instruction.

⇒ It is the pictorial representation of execution of an instruction with the help of various control & status signals.

Instruction cycle → Machine cycle → T states.

{F, MR, MW, IOR, IOW}

T states It is one subdivision of an operation performed in one clock Period. Subdivision and internal states synchronized with system clock.

$f_{CLK} = 3MHz$, $T = \frac{1}{f_{CLK}} = .33 usec.$

2) Machine cycle It is time required to access either memory or I/O. It is also equivalent to transfer a data byte to memory or I/O

1 Machine cycle may contain 3-6 T-states.

3) Instruction Cycle It is a time required to complete the execution of instruction.

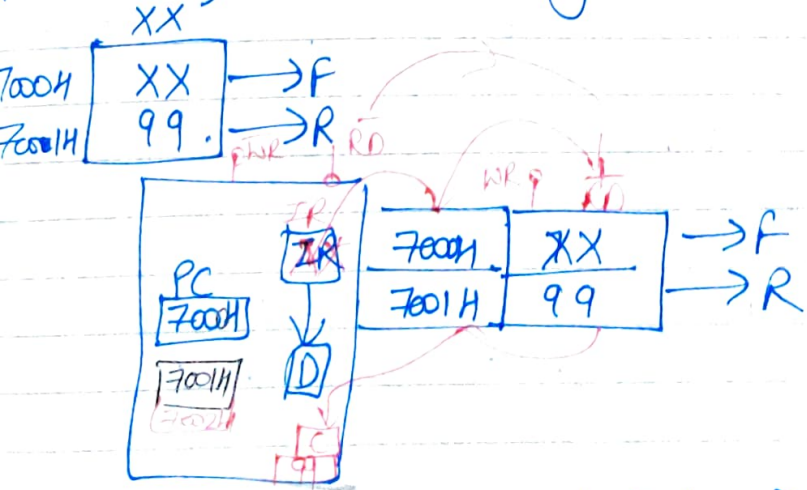
1 Instruction cycle may contain 1-5 machine cycle

The maximum no. of T states possible for execution of an instruction is 18.

eg. (CALL 16 bit Address → 3Bytes, 5 M/c's cycle - 18T states

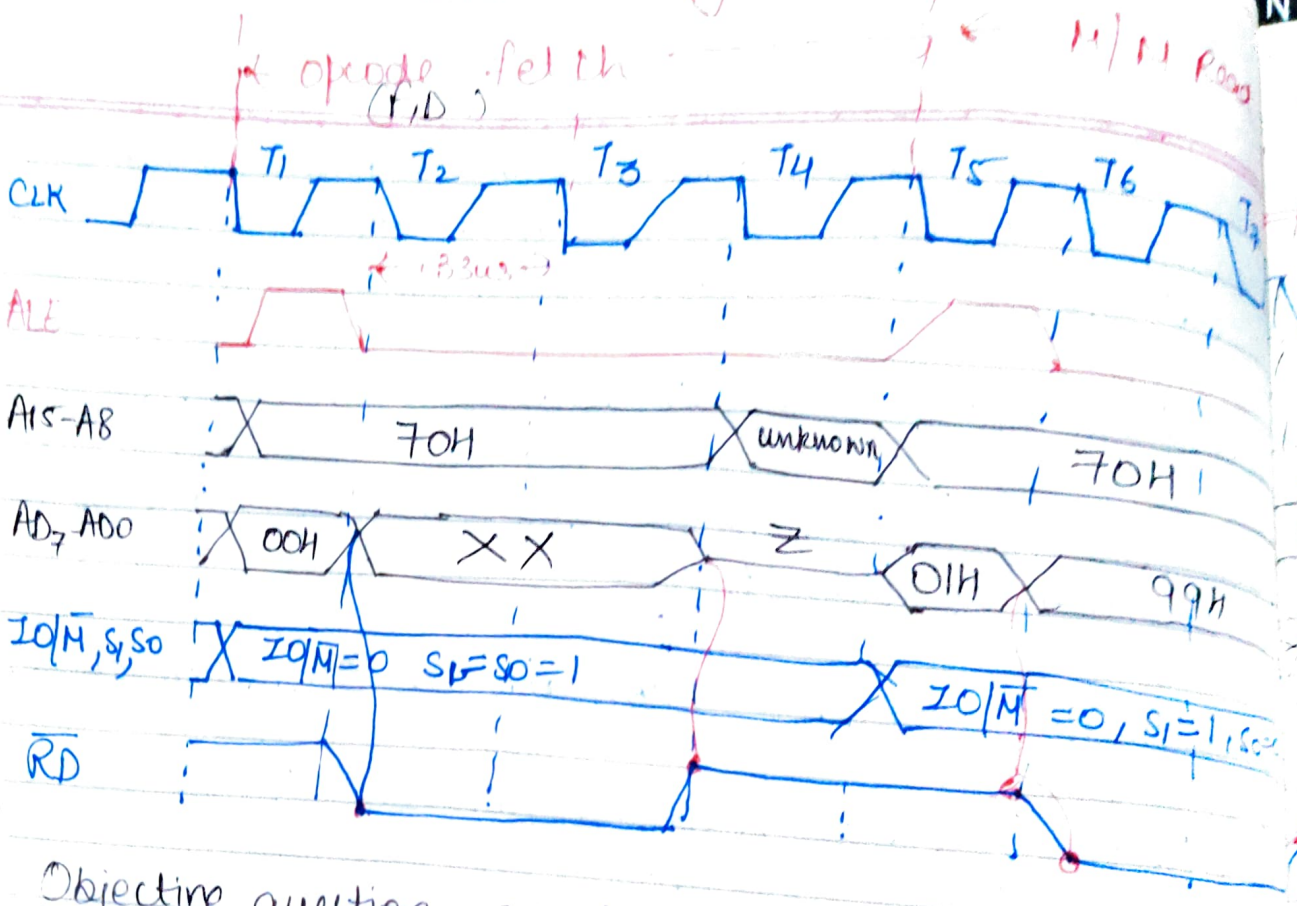
F → 4/8 6
MR →
MW →
I/O R →
I/O W →
} (3)

8050H
MVI C, 99H → 2Bytes Immediate.



MOV A, B
Requires only 4 T states

ALE-1 → 16 lines → address
→0 → A15-A8 → "
AD7-AD0 → Data



Objective question on timing diagram.

Min. T states — 3 M/C
Max T states — 6 M/C
2 M/C.

⇒ The above timing diagram is drawn by considering the instruction MVI C, 99H present at 7000H & operating frequency of 3MHz.

⇒ For any instruction to be executed, the first m/c cycle is opcode fetch which may be of 4 or 6 T states

⇒ T1 : ALE is high indicating all 16 lines acting as address lines

⇒ T2 : A8 to A15 contains higher byte of address
AD0 to AD7 " Lower byte " "

⇒ $\overline{IO/M}$ → $IO/\overline{M} = 0$ indicating memory operation

$S_1 = S_0 = 1$ for fetch.

⇒ $\overline{RD} = 1$ is inactive as there is no data Bus available

T2

⇒ As ALE becomes low, AD_0-AD_7 will acts as a data bus.

⇒ When Read signal is activated, $\overline{RD} = 0$, opcode from memory location is accessed on to data bus

T3

Opcode of data bus is accessed into Instruction Register.

T4

Opcode is decoded & execution may also be completed in some instructions like MOV A, B But, the instruction considered a memory read operation is required.

Memory Read, T5

⇒ ALE is high to point next memory location.

T6, T7

T6 & T7 are similar to fetch except that $S_1 = 1$, $S_0 = 0$ for read operation.

Conclusion

1) Length of instruction = 2 Bytes
2) No. of machine cycle = 2 Fetch / Read.
3) T states = 7 → 4 + 3
4) Total execution time.
   ⇒ Clock period X No. of T states X Count value. (no of times instruction is executed.)
   ⇒ $\frac{1}{f_{CLK}} \times T\text{states} \times C.V$
   $= \frac{1}{3 \times 10^6} \times 7 \times 1 = 2.3\,\mu s$

T state

	Min	Max
Fetch	4	6
M/c	3	6
Instruction	4	18

Imp

MV 1 C, 99H → 28,2

Note

The no. of m/c cycles required for execution of an instruction may or maynot be equal to length of instruction.

⇒ ALE is high in the * first ~~st~~ T state of a m/c cycle.

Instruction Set Classification

Instructions

Type of operation

Inst
1. Data Transfer/Copy Inst
2. Arithmatic Inst.
3. Logical "
4. Branching "
5. Machine Control Inst.

Length of Instruction

1. 1 Byte / Word Inst.
2. 2 " " "
3. 3 " " "

IN 30H → access data from input device whose address is 30H.

* OUT 40H → Output the data from 40H of I/O to Acc.

Flags are only affected for Arithmatic & Logical Inst.

1) Data Transfer / Copy Inst.

In this group, data present in source register is transferred to destination. It may be in between-

1. Reg – Reg — MOV A, B
2. Reg ⟺ M/M — MOV C, M / MOV M, D
3. Reg ⟺ I/O — IN 30H / Out 40H
4. Reg ⇐ Data — MVI E, 25H.

Note (1) flags are not affected for data transfer operations.

MOV A, B

A	B
56	78
78	

71

Q MVI A, 00H ← Data Transfer operation

a) flags are not affected.

b) zero flag is one.

(2)

⇒ The content of source is uncharged after the execution of instruction in almost all data transfer instructions.

Arithmatic Instruction

Addition / subtraction, increment / decrement.

(addition, subtraction: one type format)

Without carry

ADD R	ADD M	ADI 8 bit data
ADC R	ADC "	ACI "
SUB "	SUB "	SUI "
SBB "	SBB "	SBI "

(ADC, SBB: with carry)

⇒ Addition, Substraction, And, OR, Compare EXOR.
⇒ Increment, Decrement, Complement, Rotate

ANA	R	ANA	M	ANI	8bit data
ORA	"	ORA	"	ORI	"
CMP	"	CMP	"	CPI	"
XRA	"	XRA	"	XRI	"

Addition	ADD B
Subtraction	SUB M
Increment	INR B
Decrement	DCR H.

INR R	INR M	INX Rp
DCR "	DCR "	DCX "

Logical Instruction

AND	ANAC
OR	ORI 00H
EX-OR	XRA M
Compare	CMP D
Complement	CMA
Rotate	RRC

Note Flags are affected or modified only for arithmati & logical operations

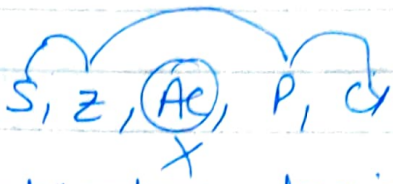
4) Branching Instructions

In this group control of the program is transferred fr one location to another conditionally or uncondit

⇒ Conditional instructions depend on status of flag (AC) → Auxilary carry. affected for previous

operation.

eg. JUMP
CALL
RETURN RET



depends on previous operation.

(8) { S=0 / =1 | Z=0 / 1 | P=0 / =1 | CY=0 / =1

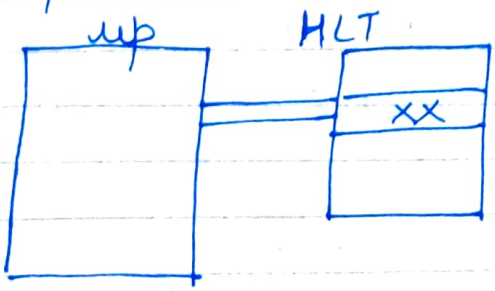
+1 unconditional
Total = 9.

JUMP → JZ 16 bit address ← 3 Bytes
(Jump if zero, then goes to 16 bit address)

Z=1
JZ 5000H Z=1? ✓ True.
True, Jump to 5000H
false. / & Next Instruction.

5] Machine Control Instruction

These are used for machine control operations internal to the processor.



eg. HLT → Halt
DI → Disable Interrupts