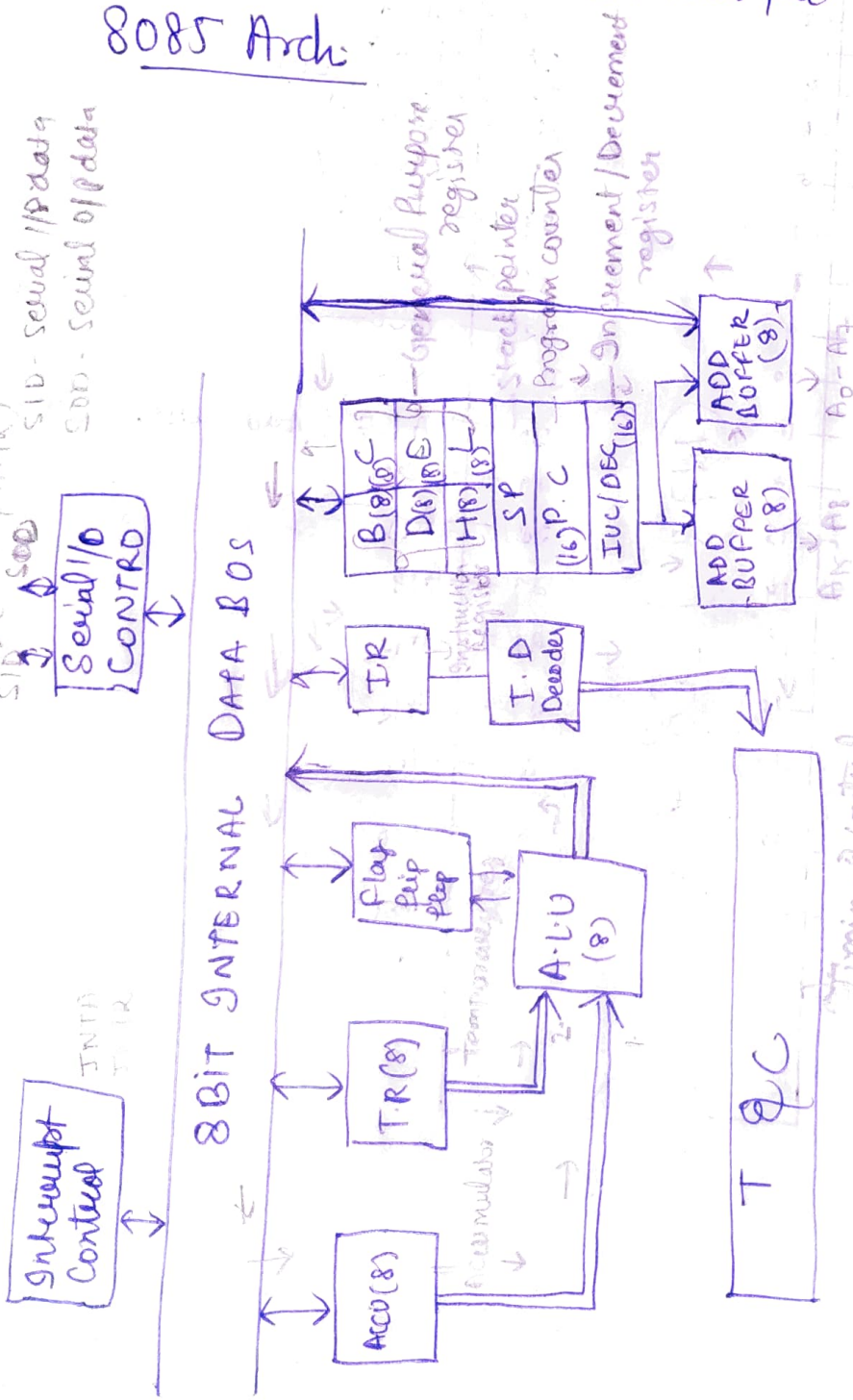
MPMC →

Unit I · Intro. to Microprocessor sys.
→ Arch. & PIN dig. of 8085
→ Timing dig.
→ Memory org.
→ Addressing modes
→ Interrupts.
→ Assembly language prog.

*. Microprocessor 8085 is a 8 bit micro. proc.
* Microprocessor 8086 is a 16 bit micro. proc. [∵ 4 bit = 1 Nibble]
[∵ 8 bit = 1 Byte]

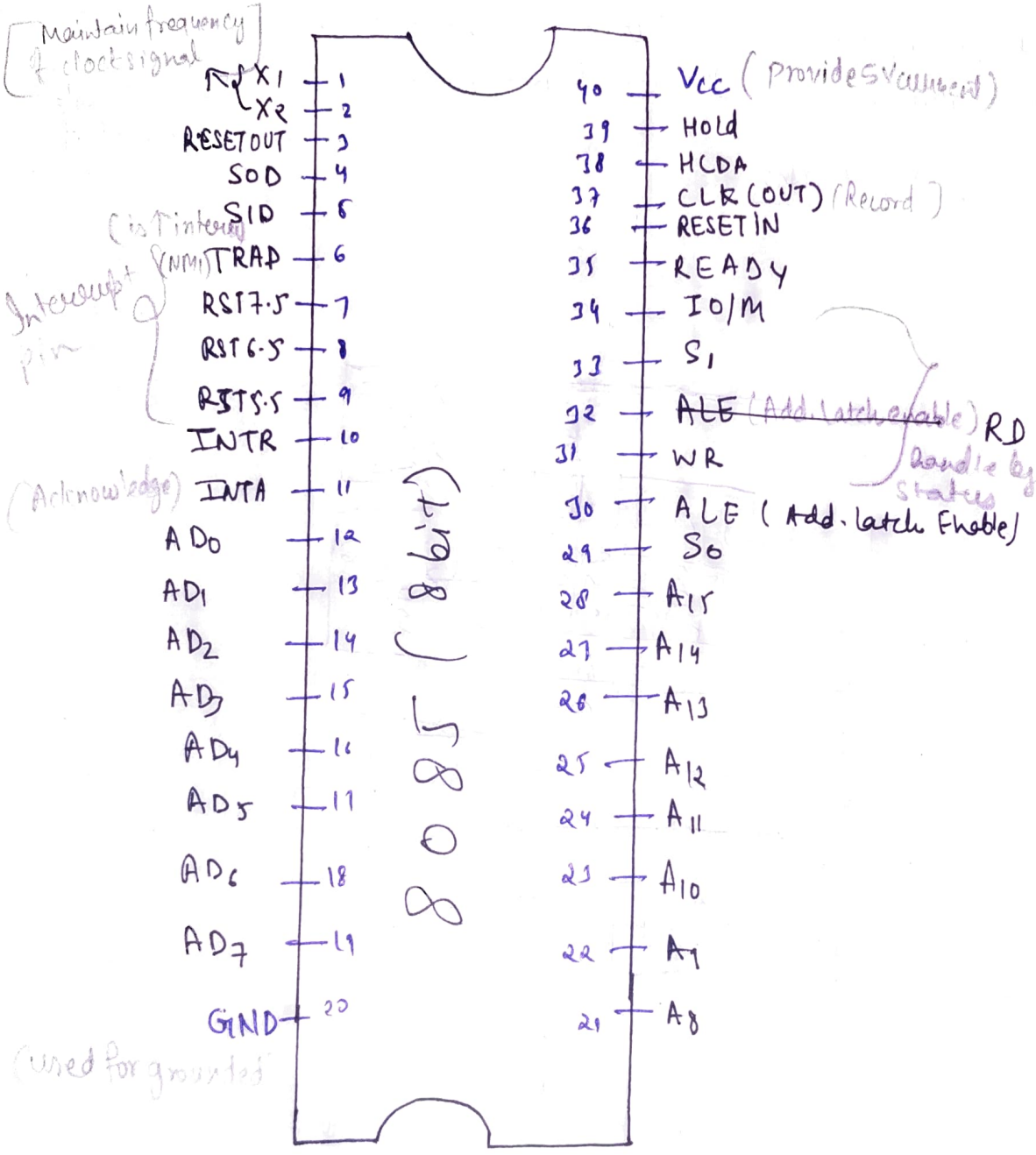
8085 Arch:



3 bus.
Address Bus → 16 Bit
Data Bus → 8 bit
Control Bus (data write or Read)

Pin Diagram of 8085.

Address Bus (A)
Data Bus (D)



* 8085 is 8bit arch.
* Addr bit size 16 bit
* Data bit size 8 bit

⇒ Multiplex

A (16)
$A_0 - A_{15}$
$A_0 - A_7$
$A_8 - A_{15}$
(works indivisually)

D (8)
$D_0 - D_7$
$D_0 - D_7$

combine with D

SID – Serial I/P Data
SOD – Serial O/P Data

⇒ 5 types of Interrupt in 8085

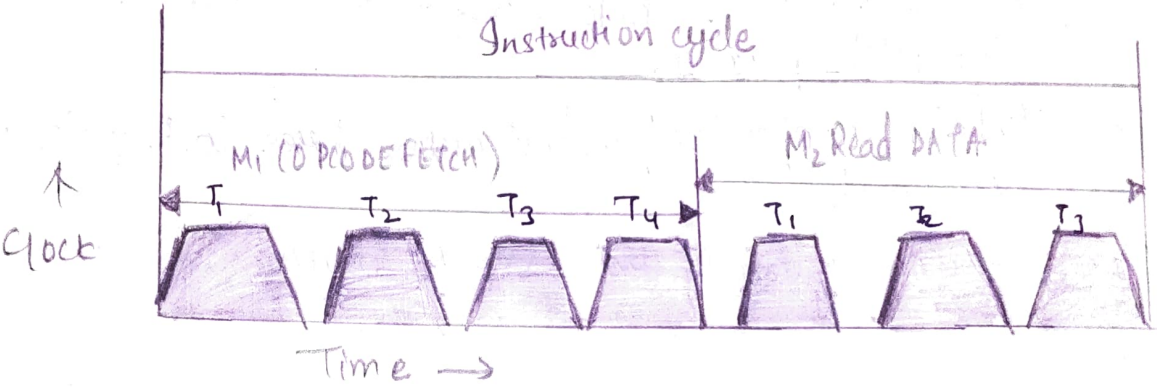
IO/M	S_0	S_1	Status
0	0	0	HALT
0	0	1	M-WRITE
0	1	0	M-READ
1	0	1	I/O WRITE
1	1	0	I/O-READ
1	1	1	Opcode Fetch
1	1	1	INT-ACK

INSTRUCTION CYCLE

⇒ A µp microprocessor fetches the instruction & executes it
⇒ The time taken for the execution of an instruction is called instruction cycle.
⇒ It consist of fetch cycle & Execute cycle.
⇒ READ and WRITE operations are the only communication b/w the processor & the other components

① Machine Cycle

⇒ ~~M.C is the sequence~~ Each READ or WRITE Operations of 8085 is referred to as Machine Cycle.
⇒ Each machine cycle contains a number of clock cycle. Clock cycle also referred to as T-states



Opcode fetch & READ Machine cycle

Machine cycle is the sequence of steps in performing the various operation like op-code fetch, memory read, memory write, I/O Read, I/O Write etc.

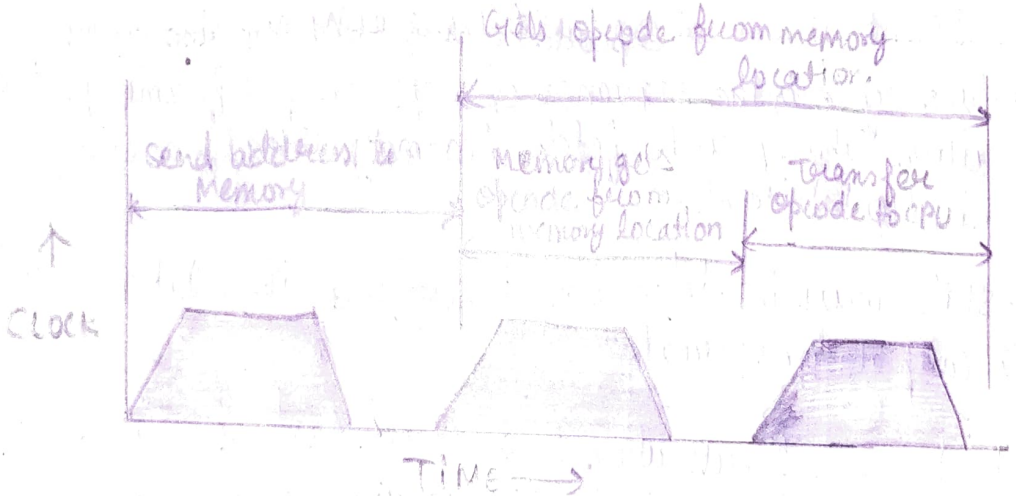
T-state may be defined as the ~~define~~ time taken by the clock to complete one period.

To perform a particular task by programmer on a computer, a programmer writes a set of instruction (d program & data are stored in the memory. The microprocessor fetches one instruction from memory at one time & executes it. It fetches and executes all the instructions of the progs. from the memory one by one to produce the final results.

Instruction cycle = Fetch cycle + Execute cycle

FETCH OPERATION

⇒ Instruction may be one byte, two byte or three byte long.
⇒ The first of any type of instruction is the opcode
⇒ The second or third Byte of an instruction may be the data or the address.
⇒ The program counter keeps the address of the next instruction to be executed.
⇒ Program counter which is the address of the memory location where op code of the instruction is stored is sent to the memory. The memory places opcode of the instruction on the data bus so as to transfer it to the CPU. This is Old Fetch operation.
⇒ To fetch an opcode of an instruction three clock cycles are required.
⇒ The clock cycle for which the CPU waits is old wait cycle



Fetch cycle.

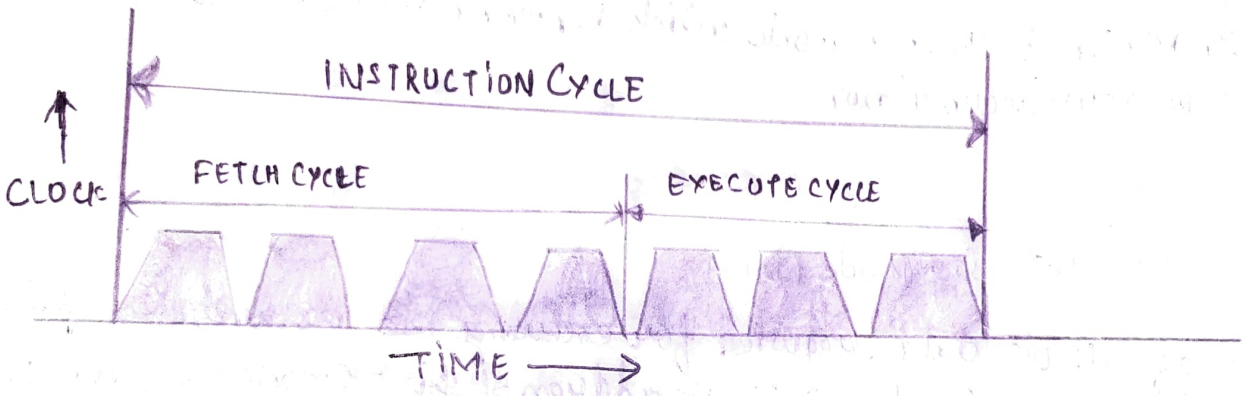
EXECUTE OPERATION

⇒ The opcode fetched goes to the data register & then it goes to instruction register.

⇒ From instruction register it goes to instruction decoder, which decodes the instruction & after decoding the instruction, the task specified in the instruction is carried out This is called Execute operation

⇒ In one byte instruction, the operand is a general purpose register & opcode & operand are specified by one byte.

⇒ In two byte or three byte instructions which contain data or address which is still lying in the memory, the CPU perform the Read operation to get the desired data, after receiving the data it performs the execute operation.



:- Languages of instructions of Microprocessor

1.) Machine language instruction
2.) Hexcode language instruction
3.) Assembly language or Mnemonics.

	Mnemonics (Assembly language)	Hexacode language (opcode)	Machine language
1.	MVIA	3E H	0011 1110
2.	LXIH	21 H	0010 0001
3.	STA	32H	0011 0010

:- ADDRESSING MODES of 8085.

⇒ The method by which address of source of data is given in a instruction is c/d addresing modes.

⇒ The method by which address of destination is given in the instruction is called addressing mode of destination.

There are 5 types of addresing modes of 8085:-

① Immediate Addressing Mode (IAM)

→ If 8/16 bit data required for executing the instruction is given along with the instruction then such instruction are c/d immediate addressing mode.

→ In these instruction the last alphabet of mnemonic will be generally 'I'

Eg- (i) MVI A, 35H
(ii) LXI B, 9000H

(Data is present in instruction itself → no need to call data from memory)

In MVIA is the mnemonic which represents move immediate data into ~~acc~~ accumulator.

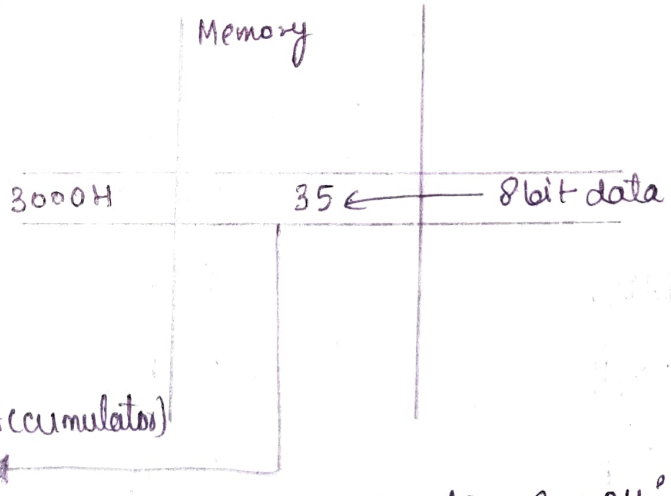
35H ——→ [A]

② Direct Addressing Mode (DAM)

→ If 8/16 bit data required for executing the instruction is present in memory location & 16 bit address of this memory location is given along with the instruction, then such instruction are called Direct addresing mode instruction (In this mode the address of the operand is given in the instruction itself)

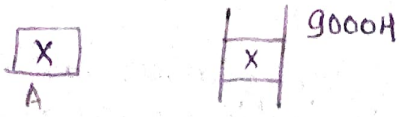
Eg: (i) LDA 3000H

Memory add. = 3000H



With this instruction 8 bit data of memory location 3000H is transferred to accumulator directly, so its addresing mode is direct

(ii) STA 9000H



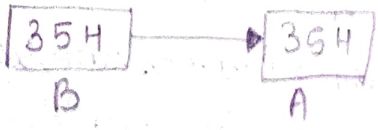
This instruction stores the content of accumulator at the given memory location directly.

(3) Register Data Addressing mode

→ If 8/16 bit data required for executing the instruction is present in Register / Register pair & the name of the Register / Register pair is given along with the instruction, then it is called Register direct addressing mode

(ii) MOV A, B

(Two registers are present in the instruction where the data moves)



(The operand is in general purpose register)

(iii) MOV H, B

(4) Register indirect addressing mode

→ If 8/16 bit data required for executing the instruction is present in memory location the 16 bit address of this memory location is present in register pair & the name of register pair is given alongwith the instruction then it is cld as 'Register indirect addressing mode'

(the address of operand is specified by a register pair)

eg. LDAX B, STAX B

(5) Implict Addressing Mode

→ If the source of data as well as address of destination of result is fixed, then there is no need to give any operand alongwith the instruction, cld 'Implict addressing mode'

eg (i) CMA (Compliment accumulator)

(Add. of source of data as well as add. of destination of result is fixed, then there is no need to give any operand along with instruction)

(ii) STC (Set carry flag)

8 bit data : 00000000 ---- 11111111
in hexadecimal representation :- 00H ---- FFH

16 Bit data : 0000000000000000 ---- 1111111111111111
in hexadecimal representation :- 0000H ---- FFFFH

:- Types of 8085 instructions.

⇒ To execute any instruction, the programmer has to translate each mnemonic into equivalent two digit hexa code.

(*) Opcode or operand.

⇒ The first part of the instructions specifies the task to be performed by the computer & is called operation code & opcode.

⇒ The second part of the instruction is the data to be operated and is called operand.

(1.) One/Single Byte instruction (SBI).

⇒ Any instruction of microprocessor in which number is not given as operand in the instruction can be completely expressed by one byte of code or/& two digit hexa code.

(i) MOV A, B

⇒ With this instruction the contents of register B is transferred to register A.

⇒ In this instruction opcode is 78H whose binary equivalent is $(01111000)_2$

First two bits 01 represent the Move operation, next three bits 111 represents registers 'A' & last three bits are the code of Register 'B'.

(ii) ADD, C

(iii) LDAX B

(2.) Two/Double Byte Instruction (DBI):-

⇒ If 8 bit no. is given as operand in the instruction then to express such instruction completely. we need two 8 bit codes.

(a) The first byte will be instruction op-code.

(b) The second byte will be 8 bit numbers given in the inst.

CPI 24H

First Byte — Second Byte.

③ Three / Triple Byte Instruction (TBI)

→ If 16 bit no. is given as operand in the instruction, & for expressing that we need three 8 bit code

(a) The first byte will be 8 bit Instruction op code.
(b) The second byte will be 8 LSB's of 16 bit number.
(c) The third byte will be 8 MSB's of 16 bit number.

eg. LXI H, 1536H

The instruction is stored as (21H, 36H, 15H)
21H is opcode of LXI H
36H is 8 LSB's of 16 bit no.
15H is 8 MSB's of 16 bit no.

* INSTRUCTION SET OF 8085-

1. Data transfer instruction
2. Arithmetic Instruction
3. Logical instruction
4. Stack instruction
5. Branching instruction
6. Machine control instructions
7. I/P & O/P instruction

Symbol	Representation	Symbol	Representation
□	Register	M	Memory location
[X]	Regi. with data X	Rd	8 bit Destination Register
##	Memory	Rs	8 bit Source Register
## 2000H	Memory with Add. 2000H	Rp	Register Pair
X 2000H	X showing contents of Memory location 2000H	Cf	Carry flag

ADDRESS	X

Register Pair giving the address of Memory having Data x

R — 8 bit Register

⟷ Exchange

16 — 16 Bit Bus

8 — 8 bit Bus.

R_H — higher 8 Bit Register of Register pair

R_L — lower 8 bit Register of Register Pair.

1.) Data transfer Instructions

⇒

The instructions which are used to transfer 8/16 bit Data from source to destination are c/d data transfer instru. The previous data of source is not lost, but the previous data of dest. get lost.

eg (i) Mov Rd, Rs — Source

destination →

(Move data of source register Rs into destination reg. Rd)

Diff. combination of Rd & Rs are

Rd ÷ A B C D E H L

Rs ÷ A B C D E H L

(ii) MVI, Rd

data → data

Rd

Q. Transfer 8 bit number 27H into accumulator

Sol

MVI A, 27H

27H → 27H

A

MVI (Move immediate data)

Addressing mode of this instruction is immediate & it is double byte instruction

(eg)(iii) LXI Rp data (16) Bit

load register pair

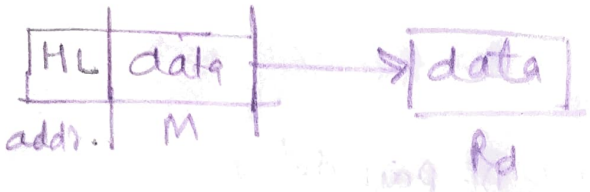
data → [data] Rp

Rp → B (BC pair), D (DE pair), H (HL pair)

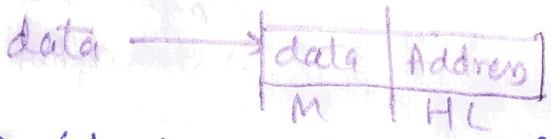
(eg)(iv) MOV M, Rs

Note:- In any instruction if 'M' is present then the address of the memory location will be always given by HL pair

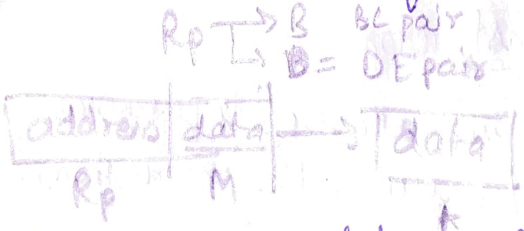
(v) MOV Rd, M (Move memory ~~register~~ data into destination register Rd, the address of memory will be given by HL pair)



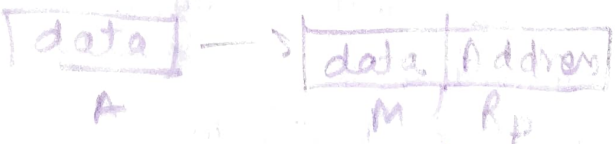
(vi) MVI M, data (Move immediate data into memory)



(vii) LDAX Rp (load accumulator from Register pair Rp data)

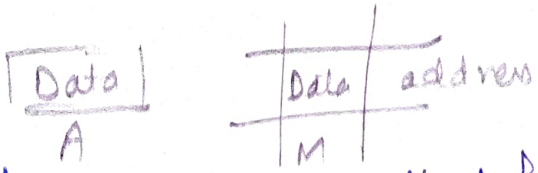


(viii) STAX Rp (Store accumulator at Register pair Rp add.)

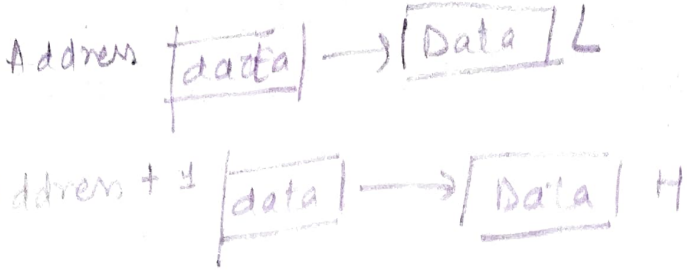


(ix) LDA address (load accumulator from address data)

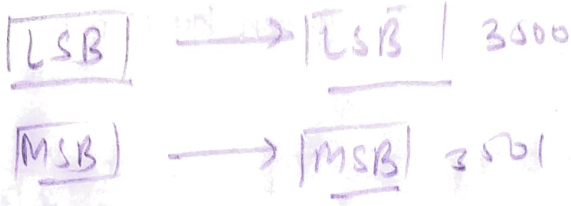
(x) STA Address (Store accumulator at given address)



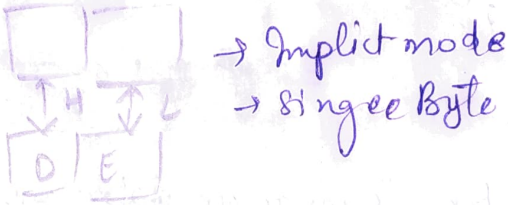
(xi) LHLD address (Load HL pair direct from address data)



(xii) SHLD address - Store HL pair data at address



(xiii) XCHG (Exchange HL & DE pair data)



(2.) Arithmetic Instructions

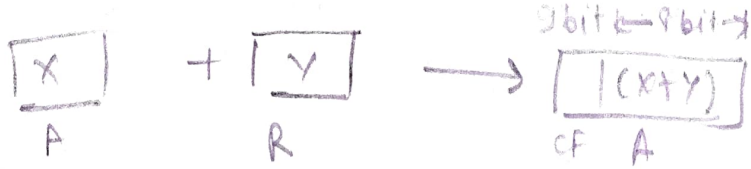
→ The instruction which are used to perform various arithmetic operations are c/d arithmetic instructions.

(i) 8 bit addition without carry
(ii) 16 bit addition without carry
(iii) 8 bit BCD addition
(iv) 8 bit Sub. with/ without borrow
(v) Increment 8/16 bit data
(vi) Decrement 8/16 bit data
(vii) Comparison of two 8 bit numbers

Note :- Microprocessor will execute arithmetic instruction in ALU & status of result is copied into status flags

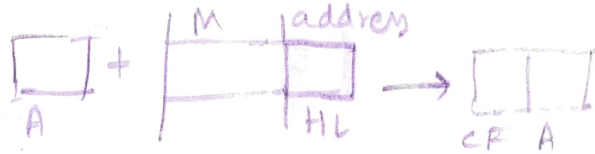
⇒ In all arithmetic Ins. the first data is taken from accumulator & result is stored back to accumulator

(i) ADD R (ADD register R Data) (A ← A + R)

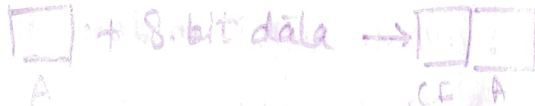


:- All flags affected after the execution of ADD R

(ii) ADD M (ADD memory data)
→ Register indirect addressing mode
→ All flags are affected
(A ← A + M)

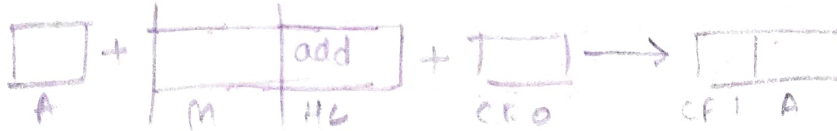


(iii) ADI data (Add immediate data)
→ Immediate addressing mode
→ All flags are affected
(A ← A + 8 bit data)



(iv) ADC R (Add with carry register R data)
→ Register direct addressing mode
→ All flags are affected
(A ← A + R + CF)

(v) ADC M (Add with carry memory data)
→ Register addressing mode
(A ← A + M + CF)



(vi) ACI 8 bit data (Add with carry immediate data)

→ Immediate addressing mode (A ← 8 bit data + CF)

(vii) DAD R_p (Double addition register pair Rp data)

→ Register direct addressing mode

→ Only carry flag is affected

Rp can be B (BC pair)

Rp can be D (DE pair)

Rp can be H (HL pair)

(viii) DAA (Decimal Adjust Accumulator)

→ Implict addressing mode

:- If 4 LSB's are greater than 9 then 6 is added to 4 LSB's of accumulator.

:- If 4 MSB's are greater than 9 or CF = 1 then 6 is added to 4 MSB's of accumulator.

(ix) SUB R (Subtract Register 'R' Data')

→ Register direct addressing mode (A ← A − R)

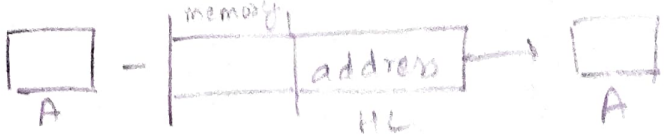
When microprocessor perform the ~~function~~ subtraction X − Y = Z then if

(i) X > Y then result Z will be either +ve or zero. The no. Z will be unsigned binary number & CF = 0

(ii) X < Y then result Z is the signed binary no. in 2's complement form and CF = 1

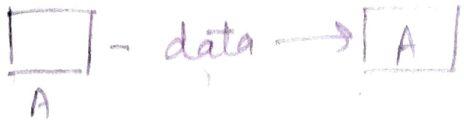
(x) SUB M (Subtract Memory Data) (A ← A - M)

→ Register indirect addressing mode



(xi) SUI data (Subtract Immediate Data)

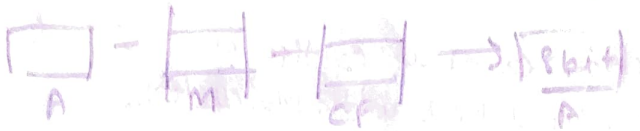
→ Immediate addressing mode (A ← A - 8 bit data)



(xii) SBB R (Subtract with borrow Register R Data)

→ Register direct addressing mode (A ← A - R - CY)

(xiii) SBB M (Subtract with borrow memory data)

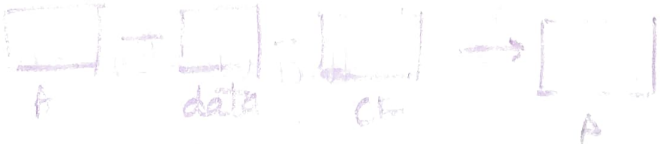


(A ← A - R - CY)

(xiv) SBI data

→ Subtract with borrow immediate data

→ Immediate addressing mode (A ← A - 8 bit data - 1)



(xv) CMP R (Compare Register R Data)

→ Register direct addressing mode (The instruction compares the data of accumulator & Register R.

a.) If A > R then CF = 0, Z = 0

b.) If A = R then CF = 0, Z = 1

c.) If A < R then CF = 1, Z = 0

(xvi) CMP M (Compare Memory Data)

a) If A > M then CF=0, Z=0

b) If A = M then CF=0, Z=1

c) If A < M then CF=1, Z=0

(xvii) CPI Data

→ Compare Immediate data.

The instruction compares the contents of accumulator & 8 bit data given along with instruction.

a) If A > data then CF=0, Z=0

b) If A = data then CF=0, Z=1

c) If A < data then CF=1, Z=0

(xix) INR M (Increment memory data) (M ← M+1)

→ Except carry flag, all flags are affected

(xx) INX Rp (Increment Register Pair Rp Data)

→ No flags are affected (Rp ← Rp+1)

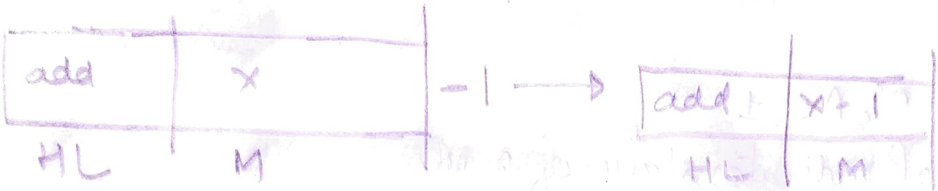
Rp can be B, D, H

B (BC pair)
D (DE pair)
H (HL pair)

(xxi) DCR RC Decrement Register R Data)

→ Except carry flag all flag are affected (R ← R - 1)

(xxii) DCR M (Decrement Memory Data) (M ← M-1)



→ Except carry flag, all flags are affected) (

(xxiii) DCX Rp (Decrement Register pair Rp data)

(Rp ← Rp+1)

(3) Logical Instruction

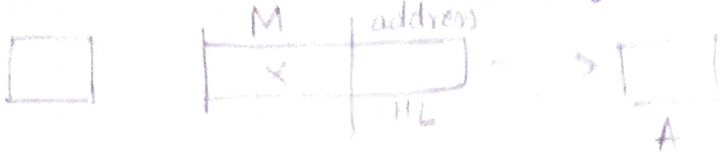
⇒ The instructions which are used to perform logical operations in microprocessor are cld as logical instructions.

(i) ANDing
(ii) ORing
(iii) Ex-ORing
(iv) 1's compliment
(v) Rotating the data towards left/right, with/without carry

(i) ANA R (AND Accumulator with Register R Data)

CF = 0 always, AC = 1 always
& Z, S, P will indicate status of result

(ii) ANA M (AND Accumulator with Memory Data)

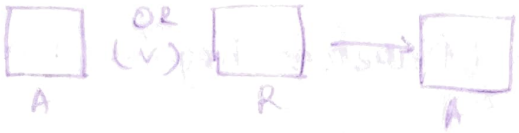


CF = 0, AC = 1
& Z, S, P indicate status of result.

(iii) ANI Data 8 bit (AND Accumulator with immediate data)

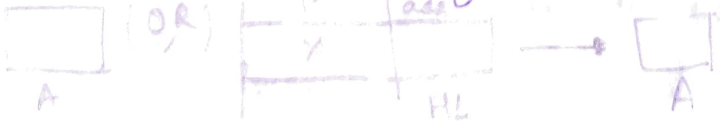
CF = 0, AC = 1
& Z, S, P indicates status of result

(iv) ORAR (OR-accumulator with Register R Data)



CF = 0, AC = 0
& Z, S, P indicates status of result.

(v) ORA M (OR-accumulator with memory data)

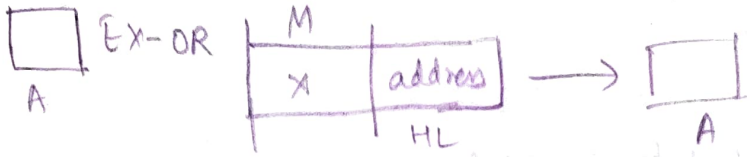


CF = 0, AC = 0
& Z, S, P indicates the status of result

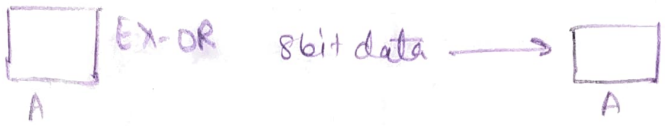
(vi) ORI Data (OR - Accumulator with Immediate data)

(vii) XRA R (Ex-OR accumulator with Register R Data)

(viii) XRA M (Ex-OR accumulator with memory data)



(ix) XRI data (EX-OR Immediate Data)



(x) CMA (Complement Accumulator)

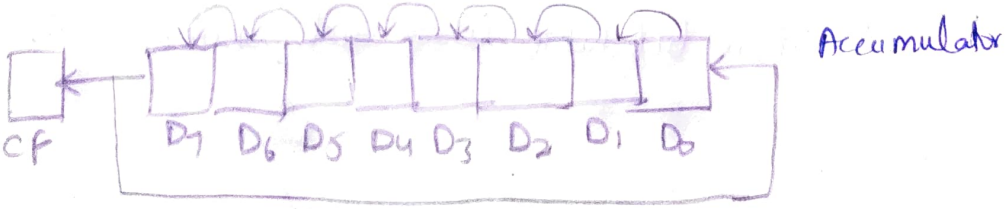
→ No flag are affected

(xi) STC (Set Carry Flag)

→ Only carry flag is affected.

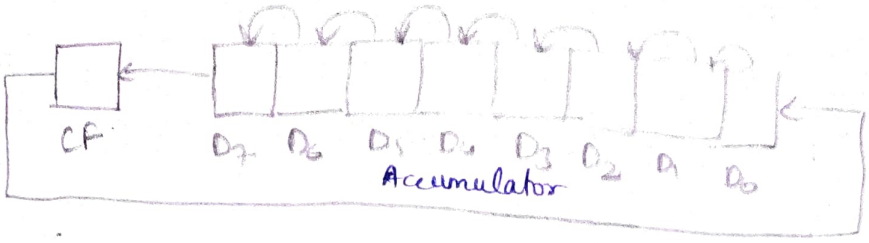
(xii) RLC (Rotate left Accumulator without carry)

→ Only carry flag is affected



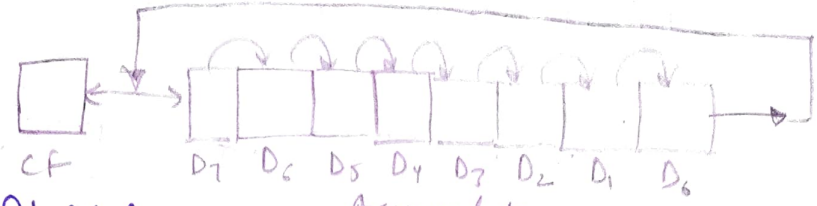
(xiii) RAL (Rotate Accumulator left through carry)

→ Only CF is affected



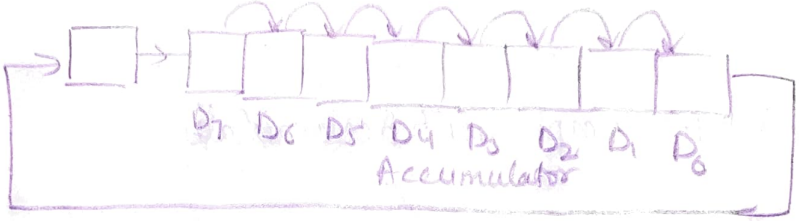
(xiv) RRC (Rotate Right Accumulator without carry)

→ only CF is affected



(xv) RAR (Rotate Right Accumulator through carry)

→ only CF is affected



(4) Stack Related Instruction

→ Stack memory is a part of Read-write memory which is used to save useful data of Register pairs in the form of stack of data.

→ It will be in the form of LIFO, FILO

→ Stack pointer is a 16 bit register.

→ It holds the address of last byte written onto the stack, this is also called as stack top.

(i) LXI SP, 16 bit data (Load Stack pointer with immediate data)

data → [data]
SP

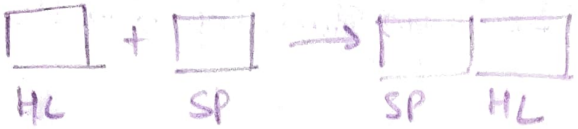
(ii) INX SP (Increment Stack pointer data by one)

→ No flag is affected

(iii) DCX SP (Decrement Stack pointer Data by one)

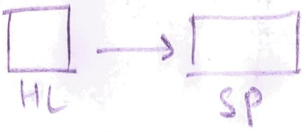
→ No flag is affected

(iv) Double Addition of Stack pointer - DAD SP



→ Only CF is affected

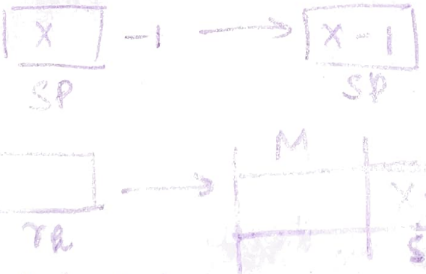
(v) SPHL (Transfer HL pair data into stack pointer)



→ No flag are affected

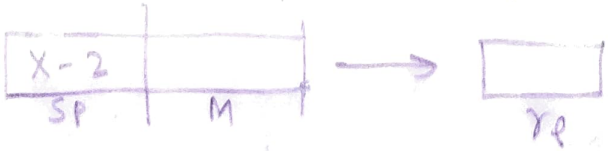
⇒ Savings the useful data from Register pair into stack memory.

(vi) PUSH RP (Push register pair RP data into Stack Memory)



(vii) POP RP (POP data into Register pair Rp)

Step 1 :-



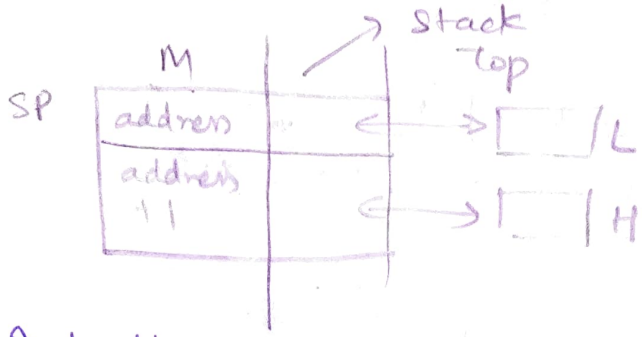
Step2 :-

Step3 :-

Step4 :-

The memory location where 16 bit address is present in SP is stack top.

(viii) XTHL (Exchange Stack Top with HL pair)

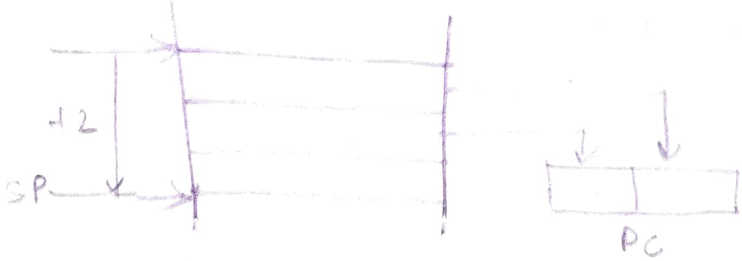


(5) Branching Instructions.

⇒ These instruction which are used to transfer new 16 bit address into PC are c/d as Branching Instructions

(i) PCHL (Transfer HL pair data into PC)

(ii) RET (Unconditional Return)



(6) Machine Control Instruction

(i) DIC (Disable interrupt)

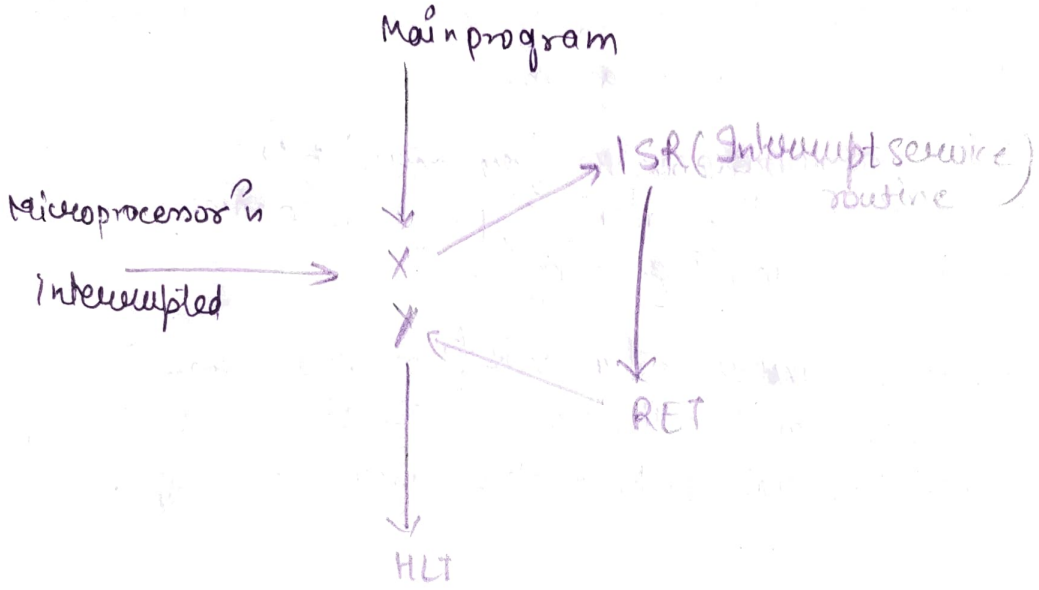
→ The interrupt enable flip-flop is reset & all the interrupts except the TRAP are disabled

→ This instruction is commonly used for giving critical time delay.

(ii) EI (Enable Interrupt)

→ The interrupt enable flip flop is set & interrupts are enabled

:- **Interrupts:**



There are two types of interrupts:-

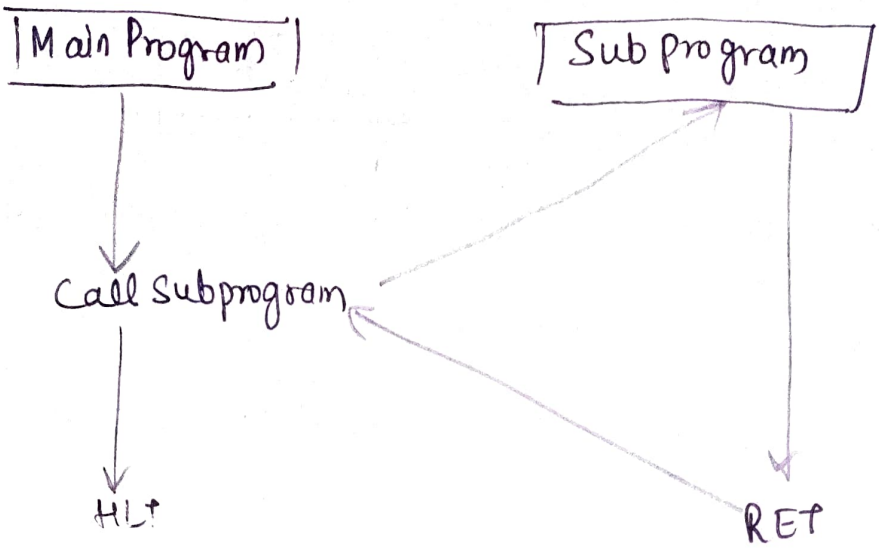
(i) Software interrupts.

→ If microprocessor is interrupted by giving ins^n in the main prog. then it is c/d as software interrupt.

The ins^n used for software interrupt are:

(1) CALL address (2) RSTn.

1.) Call address (16 Bit)



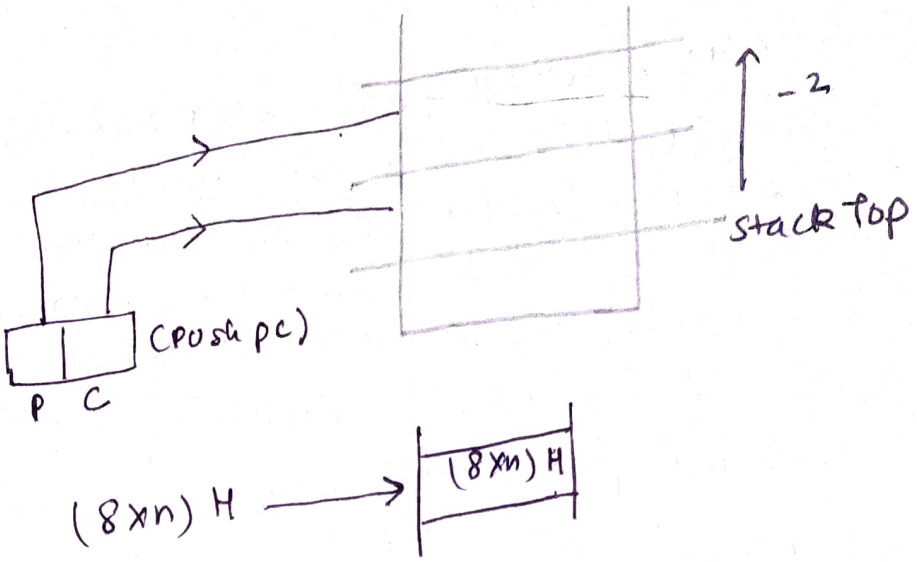
⇒ The program can be executed for number of times & when required to execute the subprogram, programmer has to give CALL instruction in the main program.

By CALL ins^n microprocessor transfers control from main prog. to subprog. Microprocessor saves the contents of program counter into stack memory by decrementing SP by 2.

By RET ins^n m.p. transfers control back from subprogram to main program. m.p. transfers the contents of stack top to program counter with this ins^n stack pointer is incremented by 2.

(2.) RSTn.

→ With this instruction, m.p. jumps to the address given by (8Xn)H, (n=0to7) n is the no. given with the RST ins.? m.p. saves the content of PC into stack memory by describing the stack top by 2. The RST instructions are equivalent to one byte CALL instruction to one of the eight memory location on page 0.



Ins^{n}	Opcode	address
RST 0	C7 H	0000 H
RST 1	CF H	0008 H
RST 2	D7 H	0010 H
RST 3	DF H	0018 H
RST 4	E7 H	0028
RST 5	EF H	0020 H
RST 6	F7 H	0038 H
RST 7	FF H	0030 H

⇒ RST ins^{n} similar to call because it first pushes the next line address to stack then ISR no. to PC

Call - 3 byte

RST - 1 byte.

⇒ Hardware Interrupt *

① TRAP :- (RST 4.5) (↓↑)

→ +ve edge, level triggered hardware interrupt
→ if signal on TRAP pin has a leading edge & a sustained high level, the MP completes the current ins^n, pushes the program counter in the stack, & branches to location 0024.
→ In order to accept another Interrupt on the trap line, the prev. trap interrupt can be disabled by the falling edge. This avoids multiple interrupts from same devices.
→ Trap is used for Powerful failure & emergency shut off.
→ highest priority, can't be disabled.
→ Non - maskable interrupt. (NMI)

② RST 7.5 :-

→ +ve edged interrupt hardware Interrupt has 2nd priority
→ can be disabled
→ Maskable interrupt
→ 8085 completes current ins.^n, Push current PC contents in stack & branches to 003C H.
→ +ve edged also set an internal F/F
→ Automatically disabled

③ RST 6.5 :- (↓↑L)

→ level triggered hardware interrupt
→ can be disable so maskable
→ After completion of current inst., Push PC contents in state & branches to 0034 H
→ automatically disabled

④ RST 5,5:-

→ level triggered Hardware Interrupt.
→ Disabled, Maskable
→ After completion, Push PC contents in stack & branches to 002 CH.
→ Enable/Disable through SIM inst.

⑤ ~~Int~~ INTR →

→ level triggered (5>T), least priority.
→ can be disabled, Maskable.
→ after completion, Push the PC into stack, generates an interrupt acknowledge (INTA) low pulse on the control Bus.
→ this instr. must be provided with external hardware.

Enable/disable through SIM Inst for RST 6.5, 7.5, 5.5

:- Machine Control Inst.

① EI all interrupts enabled
→ Enable Interrupt

② DI Disable Interrupt
→ All disable except TRAP

③ SIM (Set Interrupt Mask)
→ Masking RST 7.5, 6.5, 5.5.

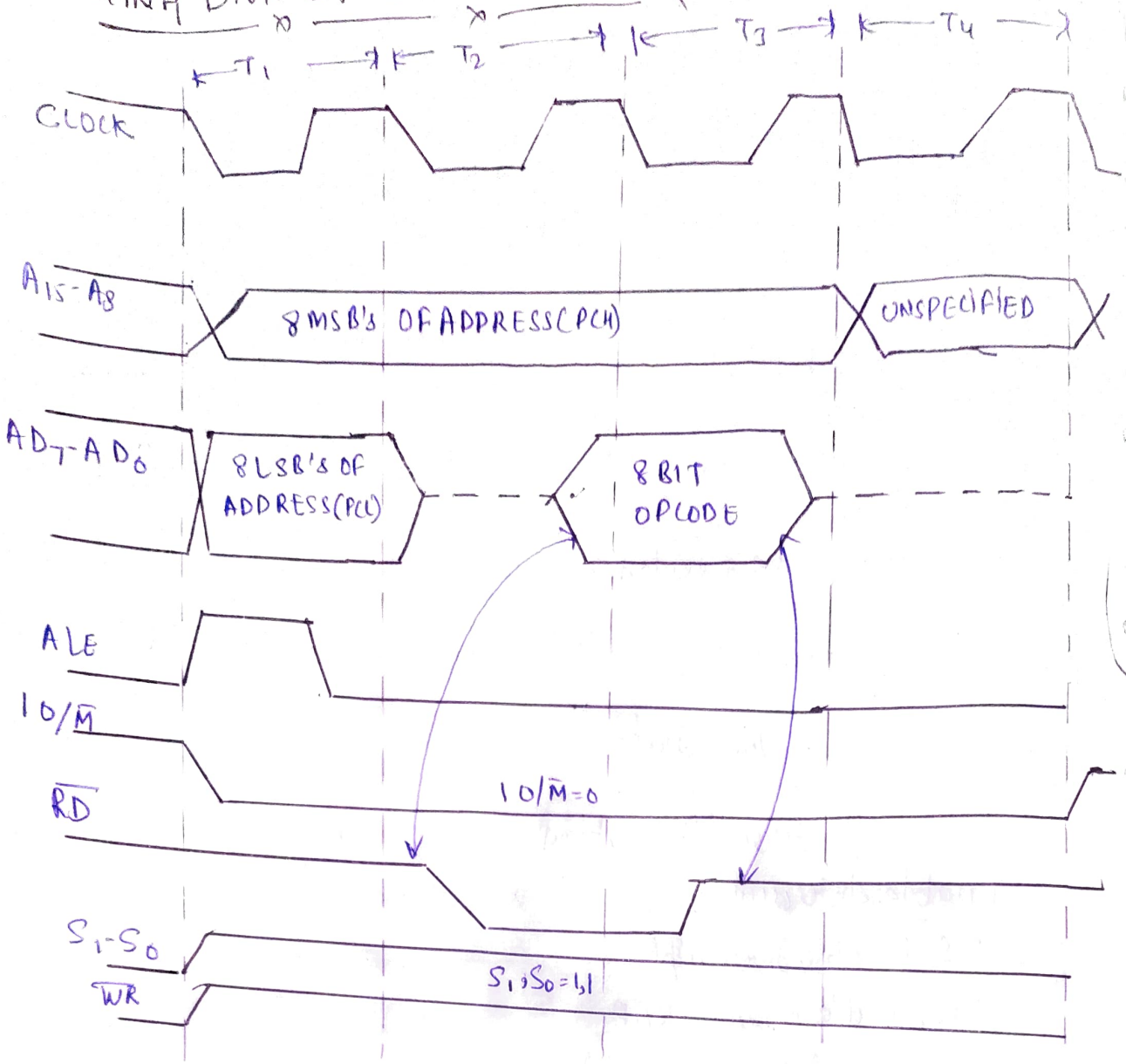
④ RIM (Read Interrupt Mask)
→ Read the status of various pending interrupts

⇒ Maskable → Interrupts which can be disabled
→ Non Maskable → " can't be disabled

→ Vectored Interrupts → When m.p. calculates the add. of ISR internally eg → RST 5.5 ⇒ 5.5 × 8 = 002C

→ Non vectored Interrupt → When add. is supplied in the form of RST opcode or call inst.

Timing Diag. of Opcode fetch Machine Cycle



Opcode fetch Machine Cycle:- The first operation in any instruction is ~~operat~~ opcode fetch. The m.p needs to get this machine code from the memory register where it is stored before the m.p. can begin to execute instruction. 8085 fetches the machine code using address & data ~~using~~ buses & the control signals.

(2) Control unit sends the control signal $\overline{RD}$. This is sent during T_2 (clock period)

(3) The byte from memory location is placed on data bus.

(4) The byte is placed in the instruction decoder of the m.p. & the task is carried out according the instruction.

⇒ 8085 uses first three states T_1 - T_3 to fetch the code & T_4 to decode & execute the opcode.

(i) T_1 Clock cycle :-

(1) M.P. will transfer 8 MSB's of address from PC (Program control) higher to 8 upper address pins $A_8 - A_{15}$

(2) The 8 LSBs of address is transferred from Pc lower to lower add. pins $AD_7 - AD_0$.

(3) To Latch 8 LSB's of add. $AD_7 - AD_0$ pins, m.p gives logic '1' to ALE pin

(4) M.P. also give $IO/\overline{M} = 0$ & $S_1, S_0 = 1, 1$

(ii) T_2 Clock cycle

(1) The 8 lower add pins $AD_7 - AD_0$ are tristated & m.p gives $\overline{RD} = 0$

(2) As $IO/\overline{M} = 0$, $\overline{RD} = 0$, $\overline{WR} = 1$, so $\overline{MRD} = 0$

(iii) T_3 clock cycle

(1) 8085 places opcode in ins? register & disable the $\overline{RD}$ signal

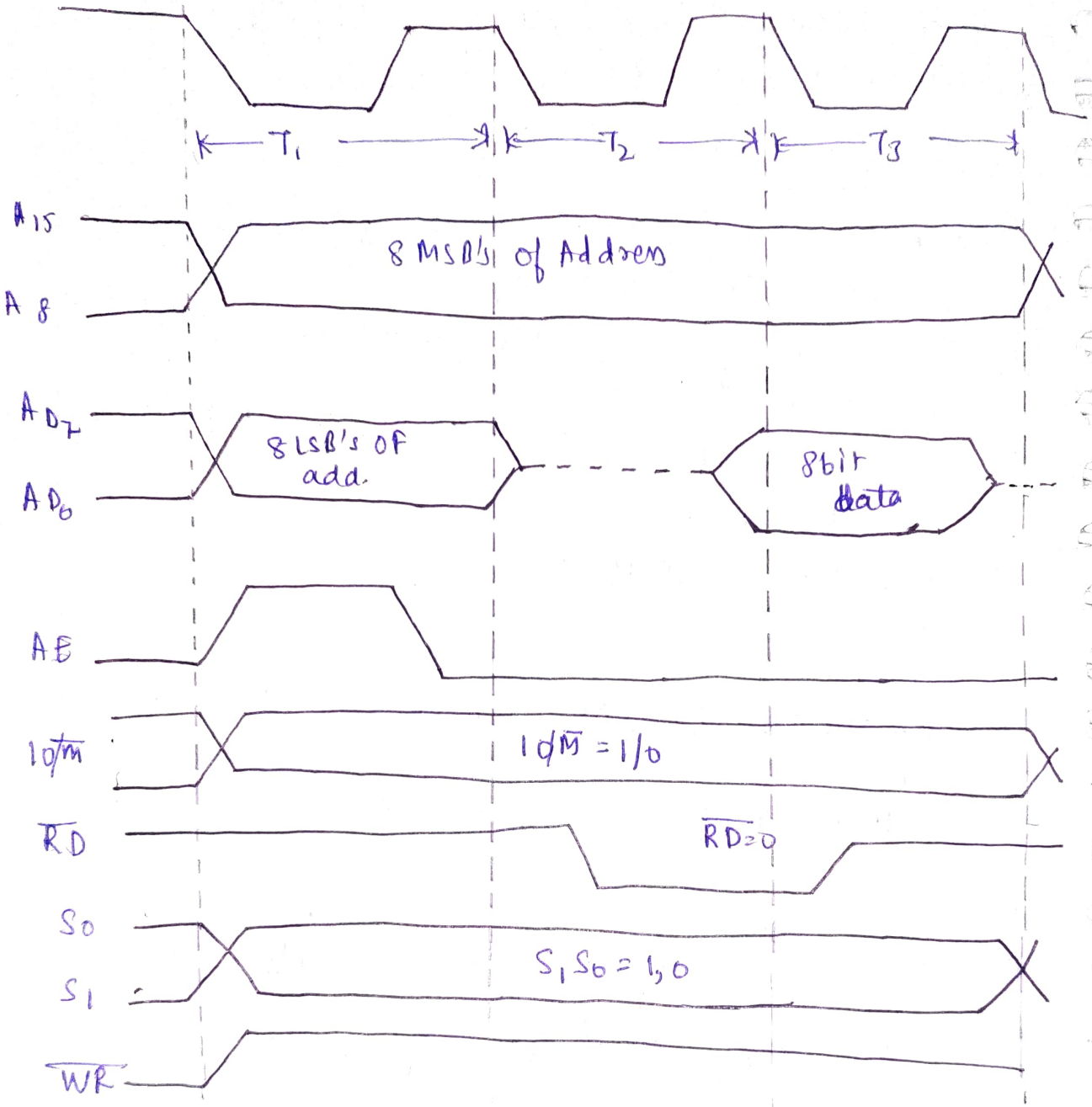
(2) The fetch cycle is completed in state T_3

(iv) T_4 Clock cycle

(1) 8085 decodes opcode & corresponding ~~inst~~ ins? is executed.

TIMING DIAGRAM OF MEMORY READ OR I/O READ

MACHINE CYCLE



⇒ Memory read or I/O read machine cycle requires 3 T states

T_1 Clock cycle

(1) M.p. will transfer 16 bit memory location address on A_8 - A_{15} & AD_0 - AD_7 pins.

(2) If m.p. is executing IN ins^n, then m.p will transfer 8 bit input port add. on A_8 - A_{15} as well as duplicated on AD_0 - AD_7 pins.

⇒ To latch 8 LSB's of address on AD_7 - AD_0 pins, m.p. gives logic 1 on ALE pin.

⇒ M.p. also gives $IO/\overline{M}$ = 1/0 resp. & S_1, S_0 = 1, 0

T_2 Clock cycle

(1) The eight data pins AD_0 - AD_7 are tristated & then m.p gives $\overline{RD}$ = 0, $IO/\overline{M}$ = 1/0, $\overline{RD}$ = 0, $\overline{WR}$ = 1 $\overline{IORD}/\overline{MRD}$ = 0.

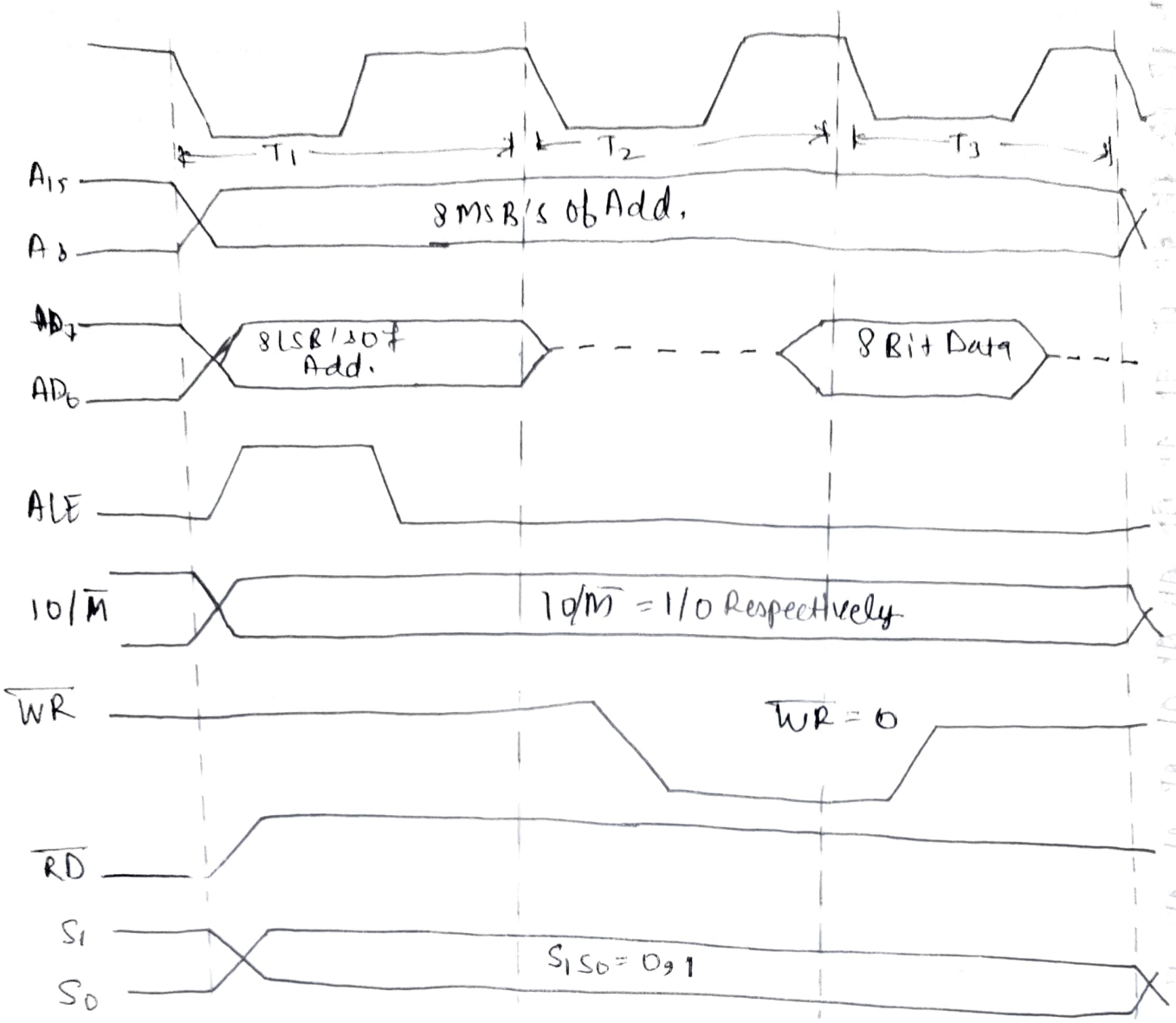
(2) 8 Bit data of selected i/p port is transferred to data pins AD_7 - AD_0

T_3 clock cycle

⇒ M.p. will transfer 8 bit data from AD_7 - AD_0 pins to any of its corresponding internal register

Timing Dig. of Memory Write OR I/O Write machine cycle

⇒ It will take 3T states



① T_1 clock cycle :-

⇒ If m.p. executing OUT ins? then m.p. gives port add. on A_{15}-A_8 as well as duplicate it on $AD_7 - AD_0$ pins.

⇒ M.p. will give logic '1' on ALE pin

② T_2 clock cycle

⇒ After transfer 8 bit data on $AD_7 - AD_0$ & the m.p gives $\overline{WR} = 0$ As $IO/\overline{M} = 1/O$, $\overline{WR} = 0$ so $\overline{IOWR}/\overline{MWR} = 0$

③ T_3 clock cycle

⇒ 8 bit data on lines is stored in selected o/p port or memory location.