OBJECTIVES:

1. study of microprocessors: 8085, 8086, Intel → 1976, 1978
   first processor ⇒ 4004 (Intel)
2. study of assembly level language of 8085
3. study of interfacing peripheral devices with 8085

Books: R. Gaonkar & N. Senthil Kumar

PROCESSOR: It is a multi-purpose, clock-driven logical device which takes input & give desired output.

# MICROPROCESSOR	MICROCONTROLLER
→ ALU, CU, Register array	→ ALU, Register Array, CU, Memory, I/O.
→ Based on Von-Neumann architecture	→ Based on Harvard architecture.
→ Data can be stored anywhere.	→ Data is stored in separate space.

* 8085 MICROPROCESSOR *

→ By intel in 1976.

FEATURES:

1. It is an 8-bit processor → Processing capability of ALU / clock cycle.
2. It has 8-bit data bus.
   * Range of data ⇒ 00H - FFH
3. It has 16-bit address bus.
   Range ⇒ 0000H - FFFFH
   ∴ Capacity = $2^{16} = 2^6 \times 2^{10}$
   = 64 KB

5. It has clock freq. of 3.0 MHz.

∴ Clock Period = $T = \frac{1}{f} \approx 0.33$ Msec

6. It supports 5 hardware & 8 software interrupts.

7. It has 40 pin DIP IC.

→ Interrupt: A request generated by external device, to the microprocessor to perform a specific task, which is known as interrupt service routine (ISR).

* Hardware Interrupts: (Vectored Interrupts)

(a) TRAP (RST 4.5)
(b) RST 7.5
(c) RST 6.5
(d) RST 5.5
(e) RST INTR

↑ dec. in priority

* Software Interrupts:

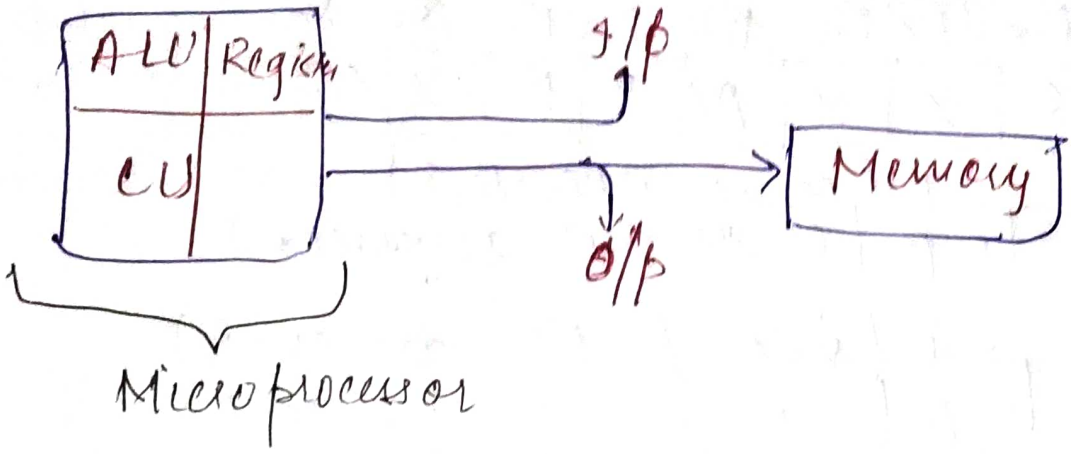
RST 0
RST 1
:
RST 7

8. It supports DMA operation.

MICRO-PROCESSOR: Multipurpose, clock driven, register based, digital IC that accepts binary data as input, process it acc. to instruction stored in its memory & provides o/p.

→ It is programmable device, performs logical & arithmetic operation.

→ Early processors → Intel 8008, i.e., → 8 bi



⇒ HCMOS: High speed Complementary Metal-Oxide Semi-Conducto
↳ Used in current tech.

8085 MICRO. P.

→ 8 bit → Intel (1977) → NMOS
→ +5 V power supply
→ supplied in a 40-pin DIP package.
→ Address up to 64K locations (16 add. line)
→ Word size = 8 bit
→ Freq. ⇒ 3.05 MHz

Bus Structure: 8 bit CPU communicates using:
→ 16-bit address bus
→ 8-bit data bus
→ a control bus

THE FLAG REGISTER : Indicate result of condition tests

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	X	AC	X	P	X	CY

CY → Carry → 1 (carry generated)
↳ 0 (carry not generated)

P → Parity → 1 (no. of 1's are even)
↳ 0 (" 0 " " " odd)

AC → Auxiliary carry → 1 (If carry generated from lower to higher nibble)
↳ 0 (vice-versa)

Z → Zero → 1 (Result = 0)
↳ 0 (Result ≠ 0)

S → Sign bit → 1 (-ve)
↳ 0 (+ve)

X → Could be anything (कोई महत्व नहीं)

* 1 1 E
9 5
B 1 9
-1 6
B 3 = (10110011)$_2$ (Question में Flag Register पूछा Replies)

(Check parity from 8-bit only)

S	Z	X	AC	X	P	X	CY
=0	=0		=1		=0		=0

* Flag Register से Information :

1 0 → -ve / ≠0
1 1 → Carry gen
0 1 → even parity
1 0 → No carry generated

ADDRESSING MODES of 8085:

Addressing mode give info. about effectiveness of operands.

Instruction: | opcode | operand |

↑ what func. should be performed.

opcode - 1 byte
operand - 1 or 2 byte in 8085

→ Length of instruction is from 1 byte to 3 byte
→ operand contains address (2 bytes), data (1 byte) or blank.

* Types of ADDRESSING MODES:

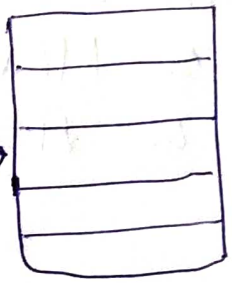
1. Implicit or Implied:
operand is defined in instruction itself.
eg: CMA

2. Immediate addressing mode:
operand is given with the instruction.
eg: MVI A, 8-bit data

3. Direct Addressing mode:

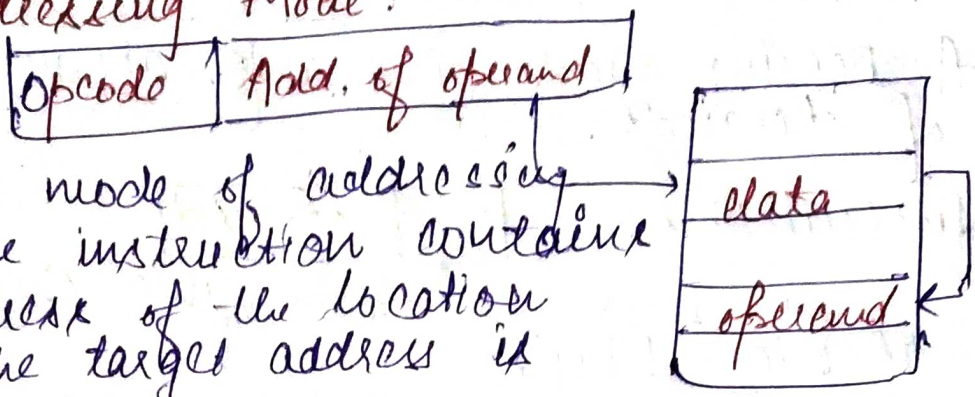
| opcode | add. of operand |

Here address is given of data with instruction present in memory.



eg: LDA, 16-bit add.
STA, 16-bit add.

4. Indirect Addressing Mode:



This is the mode of addressing where the instruction contains the address of the location where the target address is stored.

5. Register Addressing Mode:

In this addressing mode operand is placed in one of 8-bit or 16-bit general purpose register. The data is in the register that is specified by the instruction. Here one register reference is required to access the data.

Q. Find Instruction Length:

1. LDA, 05A6 ⟶ 3 byte
2. ADD B ⟶ 1 byte
3. MVI A, 05 (H) ⟶ 2 byte
4. CMP M ⟶ 1 byte
5. MOV A, B ⟶ 1 byte
6. ANI, 06 ⟶ 2 byte

TYPES of INSTRUCTIONS: HL → Memory pointer

1. Data Transfer Group:

Instruction	Operation	Length	Addressing	Flag
1. MOV Rd, Rs	Rd ← Rs	1 byte	Register	No
2. MOV Rd, M	Rd ← (HL)	1 byte	Indirect	No
3. MOV M, Rs	(HL) ← Rs	1 byte	Indirect	No
4. MVI R, 8-bit data	R ← 8-bit data	2 byte	Immediate	No
5. MVI M, 8-bit data	(HL) ← 8-bit data	2 byte	Immediate	No
6. LDA, 16-bit data	A ← 16-bit data	3-byte	Direct	No
7. STA, 16-bit data	Mem having 16-bit add. ← A	3-byte	Direct	No
8. LDAX, Rp	A ← (Rp)	1 byte	Indirect	No
9. STAX, Rp	(Rp) ← A	1 byte	Indirect	No
10. XCHG	DE ↔ HL	1 byte	Implied	No
11. LHLD, 16-bit add.	L ← (add.) H ← (add.+1)	3 byte	Direct	No
12. SHLD, 16-bit add.	(add.) ← L (add.+1) ← H	3 byte	Direct	No
13. LXI H, 16-bit add.	HL ← 16-bit data L ← Lower H ← Higher	3 byte	Immediate	No

2. Arithmetic Group :

3/16 operation on 8-bit → Flag affect
OR " 16-bit → carry flag "

Instructions	Operation	Addressing	Length	Flag
1. ADD R	A ← A + R	Register	1 Byte	All
2. ADD M	A ← A + (HL)	Indirect	1 Byte	All
3. ADI, 8-bit data	A ← A + 8 bit data	Immediate	2 Bytes	All
4. ADC R	A ← A + R + carry	Register	1 Byte	All
5. ADC M	A ← A + (HL) + carry	Indirect	1 Byte	All
6. ACI, 8-bit data	A ← A + 8 bit data + carry	Immediate	2 Bytes	All
7. DAD Rp	HL ← HL + Rp	Register	1 Byte	carry
8. DAA (Decimal adjust accumulator)	A (BCD) ← A (Binary) (Binary Coded Decimal)	Implicit	1 Byte	All
9. SUB R	A ← A - R	Register	1 Byte	All
10. SUB M	A ← A - (HL)	Indirect	1 Byte	All
11. SUI 8-bit data	A ← A - 8-bit data	Immediate	2 Bytes	All
12. SBB R	A ← A - R - carry	Register	1 Byte	All
13. SBB M	A ← A - (HL) - carry	Indirect	1 Byte	All
14. SBI 8-bit data	A ← A - 8-bit data - carry	Immediate	2 Bytes	All
15. INR R	R ← R + 1	Register	1 Byte	All
16. INR M	(HL) ← (HL) + 1	Indirect	1 Byte	All

Instructions	Operation	Addressing	Length	Flag
17. INX, Rp	Rp ← Rp+1	Register	1 Byte	Carry
18. DCR R	R ← R-1	Register	1 Byte	AU
19. DCR M	M ← M-1	Indirect	1 Byte	AU
20. DCX Rp	Rp ← Rp-1	Register	1 Byte	Carry

3. LOGICAL Instruction: (+ → (AND))

(a) AND, OR, XOR
(b) Compare
(c) Complement
(d) Rotate

(a) Instruction	Operation	Addressing	Length	Fl
1. ANA B	A ← A + B	Register	1 Byte	
2. ANA M	A ← A + (HL)	Indirect	1 Byte	
3. ANI, 8-bit data	A ← A + 8-bit data	Immediate	2 Byte	
4. ORA C	A (OR) C = A	Indirect	1 byte	
5. ORA M	A (OR) (HL) = A	Indirect	1 byte	
6. ORI 8-bit data	A (OR) 8-bit = A	Immediate	2 byte	
7. XRA D	A (XOR) D = A	Indirect	1 byte	
8. XRA M	A (XOR) (HL) = A	Indirect	1 byte	
9. XRI 8-bit data	A (XOR) 8bit = A	Immediate	2 byte	

Q. C6 H , 49 H ⇒ AND, XOR → Flags ??

⇒ C6 → 1100 0110
49 → 0100 1001
XOR → 1000 1111
AND → 0100 0000

		AND	XOR
S		0	1
Z		0	0
P		0	0
HC	AC	1	0
	CY	0	0

(b.) Compare:

(i) CMP R/M

Eg: CMP B ⟶ (checks) A > B, A = B, A < B

Contents of flags will be updated but the source & destination will not be altered.

It will perform:

A - B

& store result in flag

For:

	CY	Z
A > B	0	0
A = B	0	1
A < B	1	0

Register 1-Byte (~~Indirect~~)

Eg: CMP M ⟶ Memory Pointer (HL)

(Rest same as CMP R) (Indirect) 1-Byte

(ii) CPI 8-bit

Eg: CPI 05

	CY	Z
A > 05	0	0
A = 05	0	1
A < 05	1	0

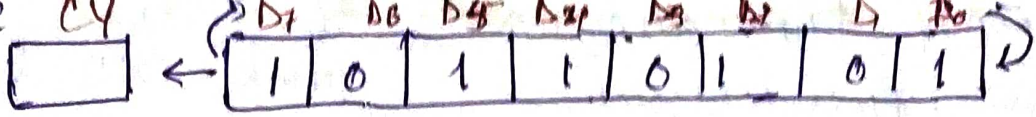
(Immediate) 2-Byte

(c) Rotate:

(i) RLC
(ii) RRC
(iii) RAL
(iv) RAR

} to rotate the content of accumulator by 1-bit to right or left.

(ii) RLC CY

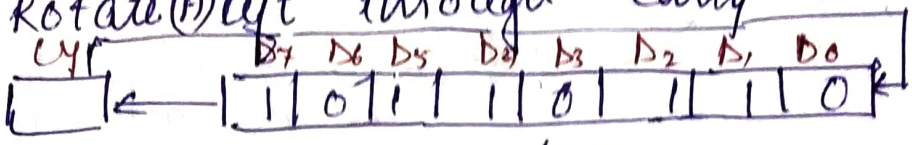


1 0 1 1 0 1 0 1 (After Rotate)

(ii) RRC: Just the opposite direction of RLC

[⇒ What's the application of rotate:
⇒ To extract LSB or MSB & check -ve / +ve or even / odd.]

(iii) RAL: Rotate (A) left through carry



⇒ Previous carry will move to D0.

(iv) RAR: JUST the opposite of RAR.

Addressing for all ⇒ Implicit
Size of all = 1 Byte

(d) Complement:

(i) CMA: Calc. 1's complement of A & result is also placed in A.
No flags are affected.

(ii) CMC: Complement carry flag
If CY=0
after CMC ⇒ CY=1 (& vice-versa)
For both: Implicit & 1 Byte

4. BRANCH INSTRUCTIONS: Instructions like jump, call & return

(a) Jump Instructions
(b) Call & Return Instructions
(c) Restart Instructions (RST0 - RST7) (For Interrupt

(a) Jump Instructions: Used conditionally (flag) or unconditionally

(i) JMP, 16-bit address: Jump unconditionally
i.e., 2002 JMP 2007,

Jump → won't come back even if there are instructions in between.

(ii) JC 16-bit address: Jump if CY = 1
(iii) JNC 16-bit address: Jump if CY = 0
(iv) JZ 16-bit " : " " Z = 1
(v) JNZ " " : " " Z = 0
(vi) JPE " " : " " P = 1
(vi) JPO " " : " " P = 0
(viii) JP " " : " " S = 0
(viii) JM " " : " " S = 1

→ Used to change the sequence of program execution.

(b) Call & Return Instructions: → Used to call a function within a main program

→ Next add. (After call) is stored in stack.

(i) CALL 16-bit add: Call unconditionally
(ii) CC " " : Call if CY = 1
(iii) CNC " " : " " CY = 0
(iv) CZ " " : " " Z = 1
(v) CNZ " " : " " Z = 0

(vi) CPE 16-bit addr : Call if P=1
(vii) CPO " " : " " P=0
(viii) CP " " : " " S=0
(ix) CN " " : " " S=1

Q. CALL N/S JUMP: Call can return but jump cannot.

(c) Return instruction:

(i) RET : Return unconditionally
(ii) RC : " if C=1
(iii) RNC : " " C=0
(iv) RZ : " " Z=1
(v) RNZ : " " Z=0
(vi) RPE : " " P=1
(vii) RPO : " " P=0
(viii) RP : " " S=0
(ix) RN : " " S=1

→ Data from the top of the stack are fetched into PC (Program counter).

Q MVI A, F2 → 1111 0010
RLC → 1110 0101
MOV B, A → B=1110 0101
RLC → 1100 1011
RLC → 1001 0111
+ 1110 0101
ADD B
HLT
1 0111 1100 ⇒ 7C

Q MVI A, 07 → 0000 0111
MVI B, 17 → 0001 0001
ADI 05 → 07+05 = 0C = 13
ANI F6 → 0000 1100
1111 0110
XRI 22 → 0000 0100
ADD B → 0010 0010
HLT
0010 0110
+ 0001 0001 = 0011 0111 = 37

TIMING DIAGRAM IN 8085

→ It represents the time taken by each instruction in a graphical format.
→ It is represented in T-states.

* INSTRUCTION that calculate the execution time:
* PARAMETER that calculate the execution time:

(a) Instruction cycle: Time req. to execute an instruction.
↓ Ins. cycle = 1-5 machine cycles

(b) Machine cycle: Time req. to complete 1 operation, i.e., accessing mem. or I/O.
↓ Machine cycle = 3-6 T-states

(c) T-state: Time corres. to one clock cycle, req. to calc. time.

⇒ Types of machine cycles:
1. Opcode fetch cycle (4T)
2. Memory read cycle (3T)
3. Memory write cycle (3T)
4. I/O read cycle (3T) (IN & OUT)
5. I/O write cycle (3T)

1. OPCODE FETCH CYCLE:

Instruction:
* MOV B, C ⇒ 0000 opcode
1. M/C cycle = OF cycle
∴ No. of M/C = 1
T-state = 4T

* MVI A, 05
→ 2000 3E (OF cycle)
→ 2001 05 (MR cycle)
∴ No. of M/C = 2
T-State = 07

* LDA 2005

=> 3000 3A (OF)
3001 05 (MR)
3002 20 (MR)
2005 12 (MR)

* STA 3000

∴ M/C = 4
T-state = 13T.

* SHLD 3000