

Lecturer's Note

**MICROPROCESSOR & MICROCONTROLLER
(MPMC)**

Module - II

Semester : 5th (ETC)

Banaja Mohapatra

Dept. Of Electronics & Telecom. Engg.

Instruction Set of 8085 μp :-

LXI H, 4050 - 3 byte - 3 m/c

MOV A, B - 1 byte - 1 m/c

MVI C, 00H - 2 byte - 2 m/c.

Instruction cycle consists of m/c cycles.

MOV A, M - 1 byte + read opr

T-state — No. of $\frac{\text{m/c}}{2 \text{ m/c}}$ cycles required.

$4 \text{ m/c} - \frac{1}{2} \text{ T-state}$

All the instructions in 8085 are divided into five groups. These are -

- ① Data transfer instruction.
- ② Arithmetic instruction.
- ③ Logical instruction.
- ④ Branch group instruction.
- ⑤ Stack machine and I/O control instruction.

Data transfer instructions :-

i) MOV R1, R2

R2 : source

R1 : destination.

$\text{MOV } [R2] \rightarrow [R1]$ or $[R2] \rightarrow [R1]$

$[R1]$: content of register R1.

ex - MOV A, B

$[A] \leftarrow [B]$

Length : 1 byte

Addressing mode - Register

M/C : 1 cycle.

addressing mode.

T-state : 1

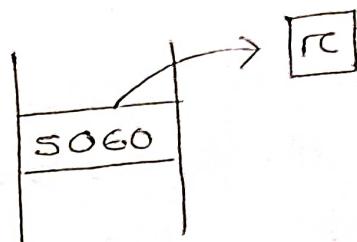
ii) Arithmetic instruction. Mov RC, M

M : Indirect memory having address at H,L pair.

$[[H-L]]$: Indirect memory.

$\approx [5060]$ when $H = 50$ $L = 60$

$[[H-L]] \rightarrow [RC]$



Length : 1 byte

M/C : 2 cycle.

T-state : $\frac{1}{2}$

Addressing mode : Register indirect addressing.

iii) Mov M,RC

$[RC] \rightarrow [[H-L]]$

L : 1 byte . T-state : $\frac{1}{2}$

M/C : 2 cycle A.M : Register indirect addressing

(iv) MVI RC,data(1 byte)

Data should be of 8 bit / 1 byte to equalise the RC.

$[RC] \leftarrow \text{data}$

ex - MVI B,02H

L : 2 byte

M/C - 2 cycles.

T-state - $\frac{1}{2}$

A.M - Direct addressing mode.

Immediate

(v) MVI M, data (1 byte)

$[H-L] \leftarrow$ data (8-bit)

L : 2 byte

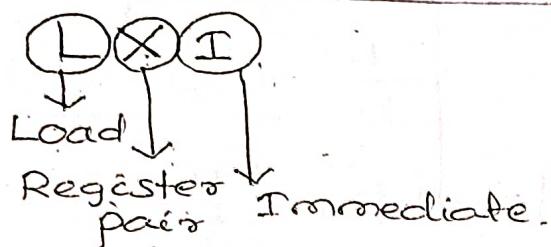
M/C : 3 cycle

T-state : (10)

A·M : Immediate / Reg. indirect

$[H-L]$ yoso

(vi) LXI R_p, data (16 bit) :-



ex — LXI D, 2060H

D E ← 20 60

$[R_p] \leftarrow$ 16 bit data

or,

$[R_h] \leftarrow$ 8MSB

$[R_l] \leftarrow$ 8LSB

L : 3 bytes.

M/C : 3

T-state : 10

A·M : Immediate addressing

(vi) LHD addr :-

LDA address :- (load acc. addr) read

L : 3

M/C : 4

T : 13

A·M - Direct address

$[addr] \rightarrow [A]$

ex - LDA 4050H

viii) STA addr :-

Store acc. addr. (write)

L : 3

M/C : 4

T-state : 13

A·M : Direct addressing

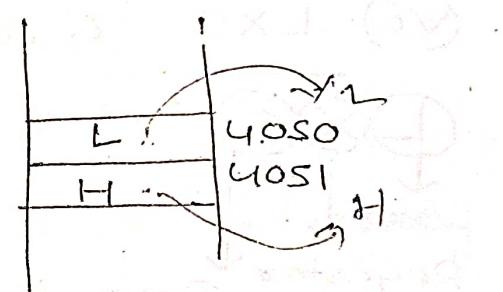
$[A] \rightarrow [addr]$

ix) LHLD addr :- (load HL direct addr)

✓ ex - LHLD 4050H
16 bit 8 bit

so L - 4050H
H - 4051H

H L
MSB LSB



$[addr] \rightarrow [L] \text{ or } [m_L]$

$[addr+1] \rightarrow [H] \text{ or } [m_H]$

L : 3

M/C : 5 (3+2 read opr)

T-state : 16

A·M - Direct addressing

x) SHLD addr - (store HL pair direct)

$[m_L] \rightarrow [addr]$

$[m_H] \rightarrow [addr+1]$

L : 3

M/C : 5

T-state : 16

A·M - Direct addressing

(xi) XCHG (exchange)

$[D-E] \leftrightarrow [H-L] \text{ or }$

$[D] \leftrightarrow [H]$

$[E] \leftrightarrow [L]$

A·M - Register addressing

L : 1

M/C : 1

T : 4

(xii) LDAX $\ddagger p$. (load acc. indirect)

ex - LDAX D.

$$[[\ddagger p]] \longrightarrow [A]$$

L:1

M/C:2 (read)

T: $\frac{1}{2}$
a.m - Reg. direct

(xiii) STAX $\ddagger p$

$$[A] \rightarrow [[\ddagger p]]$$

L:1

M/C:2 (write)

T: $\frac{1}{2}$
A.M : Reg. direct

Arithmatic instructions :-

(i) ADD R :-

$$[RC] + [A] \rightarrow [A]$$

(Add register pair to H-L pair)

L:1 byte

M/C:1 byte

T:4

A.M - Reg. addressing.

DAD RC

$$[H-L] \leftarrow [H-L] + [RC]$$

(ii) ADD M :-

$$[A] + [H-L] \rightarrow [A]$$

$$[H-L] \leftarrow [H-L] + [H-L]$$

L:1

M/C:2 byte

T:7

A.M : Reg. indirect.

L:3

M/C:3

T:10

Add - Reg. addr

(iii) ADD R (add with carry)

$$\checkmark [A] \leftarrow [A] + [RC] + [CS]$$

L:1

M/C:1

T:4

A.M - Reg. addressing.

(iv) ADC

$$\checkmark [A] \leftarrow [A] + [H-L] + [CS]$$

L: 1

M/C: 2

T: 3

A·M - Reg.indirect

(v) ADD data (8 bit) Ex: ADD 10H

$$8\text{ bit data} + [A] \rightarrow [A]$$

L: 2

M/C: 2

T: 3

A·M - Immediate

(vi) ACI data (8 bit)

$$[A] \leftarrow [A] + [8\text{ bit data}] + [CS]$$

L: 2

M/C: 2

T: 3

A·M - Immediate

(vii) SUB R

$$[A] - [R] \rightarrow [A]$$

L: 1

M/C: 1

T: 4

A·M - Reg.

(viii) SUB M

$$[A] - [H-L] \rightarrow [A]$$

L: 1

M/C: 2

T: 3

A·M: Reg.indirect

(x) SBB R (subtract with borrow)

$$[A] \leftarrow [A] - [R] - [CS]$$

L: 1

NYC: 1

T: 4

A.M - Reg. addressing.

xi) SBB M

$$[A] \leftarrow [A] - [H-L] - [CS]$$

L: 1

NYC: 2

T: 2

A.M - Reg. indirect.

xii) SUI data (8bit) (subtract immediate data from accumulator).

$$[A] \leftarrow [A] - \text{data}$$

L: 2

NYC: 2

T: 2

A.M - Immediate.

xiii) SBF data

$$[A] \leftarrow [A] - \text{data} - [CS]$$

L: 2

NYC: 2

T: 2

A.M - Immediate.

xiv) INR R

$$[R] + 1 \rightarrow [R]$$

L: 1

NYC: 1

T: 4

A.M - reg.

INR M

$$[H-L] \leftarrow [H-L] + 1$$

L:1

M/C:3

T:10

A.M - Reg. indirect

DCR R

$$[R] \leftarrow [R] - 1$$

L:1

M/C:1

T:4

A.M - Reg.

DCR M

$$[H-L] \leftarrow [H-L] - 1$$

L:1

M/C:3

T:10

A.M - Reg. indirect

INX RP (16 bit)

$$[RP] \leftarrow [RP] + 1$$

L:1

M/C:1

T:6

A.M - Reg. addr.

DCX RP

$$[RP] \leftarrow [RP] - 1$$

L:1

M/C:1

T:6

A.M - Reg.

DAA (Decimal adjust acc.)

✓ This instruction is used for BCD operation.

After any BCD operation the result in the accum. can automatically adjusted to its BCD equivalent by using this instructions.

L:1

MVC:1

T:4

A.M - Implicit.

Logical Instructions of 8085 :-

① ANA X

$$[A] \leftarrow [A] \wedge [X] \quad \wedge - \text{conjunction}$$

MVI A,B0H

MVI C,1FH

ANA C

B0	1011	0000
1F	0001	1111
10	0001 0000	

L:1

MVC:1

T:4

Add. mode - register.

② ANA M

$$[A] \leftarrow [A] \wedge [H-L]$$

L:1

MVC:2

T:2

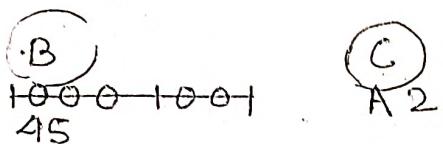
Address. - Reg. Indirect.

ORA R

LXI B, 45A2H

MVI A, 1SH.

ANA C.



1010 0010

0001 0101

$$\text{0000 } 0010 \cong 02.$$

ORA R, (Content of A OR with register)

$$[R] \vee [A] \rightarrow [A]$$

L=1

MVC=1

T:4

add-reg. addressing
mode

D	E	A
20	45	F6

ORA D

20 — 0010 0000

F6 — 1111 0110

$$\begin{array}{r} 10001 \\ + 0110 \\ \hline 10001 \end{array}$$

01000

ORA M

$$[A] \leftarrow [A] \vee [H-L]$$

MVC:2

L:1

T:7

A : reg. indirect

ORT data

$$[A] \leftarrow [A] \vee \text{data}$$

L: 2 bytes.

M/C: 2

T: 2

Ad: Reg. Imm. addressing

XRA R

$$[A] \leftarrow [A] \vee r$$

L: 1

M/C: 1

T: A

Ad: Reg. addressing

XRA M

$$[A] \leftarrow [A] \vee [H-L]$$

L: 1

M/C: 2

T: 2

Reg. indirect

XRF data

$$[A] \leftarrow [A] \vee \text{data}$$

L: 2

M/C: 2

T: 2

A-M - Imm. add.

CMA (Complement accumulator)

$$[A] \leftarrow \overline{[A]}$$

L: 1

M/C: 1

T: 4

A-M - Implicit addressing.

ORT data

$$[A] \leftarrow [A] \vee \text{data}$$

L: 2 byte.

M/C: 2

T: 2

Ad: Reg. Imm. addressing.

XRA R

$$[A] \leftarrow [A] \vee R$$

L: 1

M/C: 1

T: 1

Ad: Reg. addressing.

XRA M

$$[A] \leftarrow [A] \vee [H-L]$$

L: 1

M/C: 2

T: 2

Reg indirect.

XRT data

$$[A] \leftarrow [A] \vee \text{data}$$

L: 2

M/C: 2

T: 2

A.M - Imm. add.

CMA (Complement accumulator)

$$[A] \leftarrow \overline{[A]}$$

L: 1

M/C: 1

T: 4

A.M - Implicit addressing.

CMC (Complement carry)

$[CS] \leftarrow \overline{[CS]}$ (1 bit data).

L:1

MVC:1

T:4

A·M: no addressing mode is specified.

STC (set carry)

$[CS] \leftarrow 1$

L, M/C, T, A·M — do —

CMP rc.

This instruction is used to compare the content of register r with content of accumulator.

$[A] - [rc]$	$A > rc$	Z	C
		0	0
	$A < rc$	0	1
	$A = rc$	1	0

Here the content of rc is internally subtracted from accumulator content and the flag bits are changed according to the result as follows.

L, M/C, T — do —

AM: Reg. addressing.

CMP N1 :-

$[A] - [H-L]$	$A > M$	Z	C
	$A < M$	0	0
	$A = M$	0	1

L-1, M/C-2, T-?

Add. mod - Reg. ind.

CPI data

$\checkmark [A] = \text{data}$	$A > \text{data}$	Z	C
		0	0
		0	1
	$A = \text{data}$	1	0

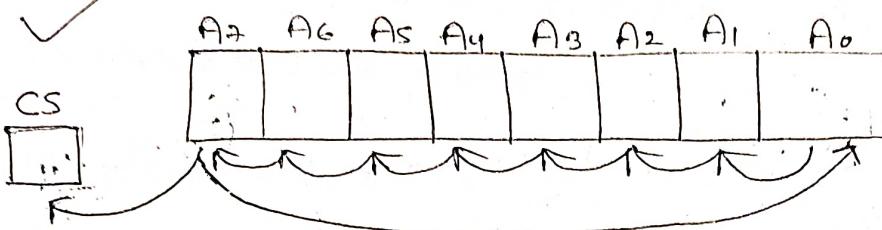
L: 2

NYC: 2

T: 7

AM: Imm. add.

RLC (Rotate acc. left)



0010 1010 $\xrightarrow{\text{RLC}}$ 01010100

$[A_{n+1}] \leftarrow [A_n]$ for $n = 0 \text{ to } 6$.

$[A_0] \leftarrow [A_7]$

$[A_7] \rightarrow [CS]$

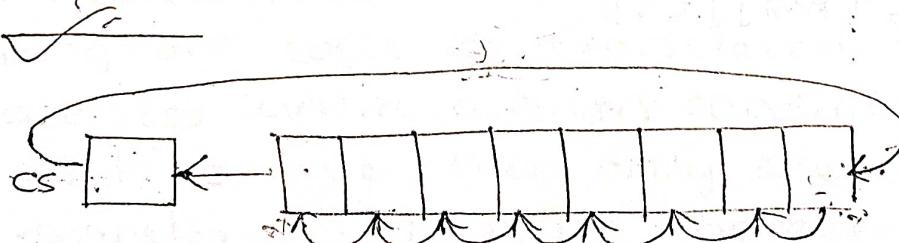
L: 1

NYC: 1

T: 4

address: Implicit

RAL (Rotate acc. left through carry)

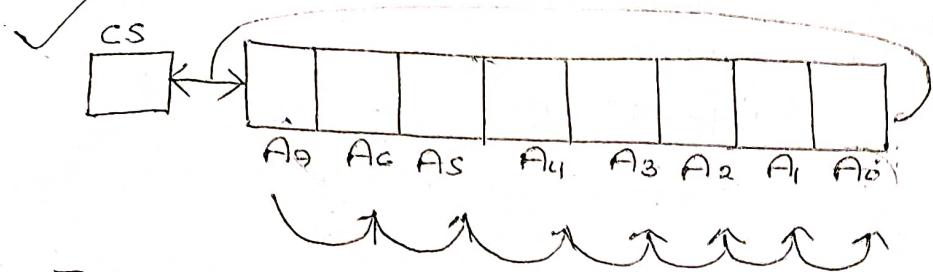


$[A_{n+1}] \leftarrow [A_n]$

$[CS] \leftarrow [A_7]$

$[A_0] \leftarrow [CS]$

RRC (Rotate right without carry) :-



$$[A_{n-1}] \leftarrow [A_n] \quad n = 1 \text{ to } 7$$

$$[A_7] \leftarrow [A_0]$$

$$[CS] \leftarrow [A_0]$$

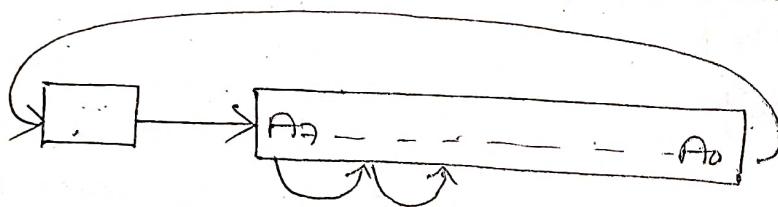
L : 1

MC : 1

T : 4

A.M - Implicit

RAR (Rotate acc. right through carry)



$$[A_{n-1}] \leftarrow [A_n]$$

$$[CS] \rightarrow [A_9]$$

$$[A_0] \rightarrow [A_{n-1}] [CS]$$

BRANCH GROUP INSTRUCTIONS

19/08/09

This group of instructions are used for abnormal termination of program flow.

JUMP

The jump instruction/operation may be conditional/unconditional.

Unconditional jump :-

① ~~JMP~~ addr. :-

In unconditional jump the program control is transferred to the address specified by the 16 bit operand.

$$[PC] \leftarrow \text{addr.}$$

L : 3

M/C : 3

T : 10

AM - Immediate. This instruction specifies the address of the target location.

Conditional jump :-

In conditional jumping the control of the program will be transferred to the specified address under a given condition i.e. If the cond' is true, then only the control will be transferred, otherwise the execution will be normal.

The conditions are given according to the status flag - (except auxiliary carry).

The conditional jump statements are -

- ① JC addr (carry), CS = 1.
 - ② JNC " (no carry), CS = 0
 - ③ JZ " (zero), Z = 1.
 - ④ JNZ " (nonzero), Z = 0.
 - ⑤ JP " (plus), S = 0.
 - ⑥ JM " (minus), S = 1.
 - ⑦ JPE " (parity even), P = 1.
 - ⑧ JPO " (parity odd), P = 0
- } Conditions.

L : 3

M/C : 3

T : 10/7

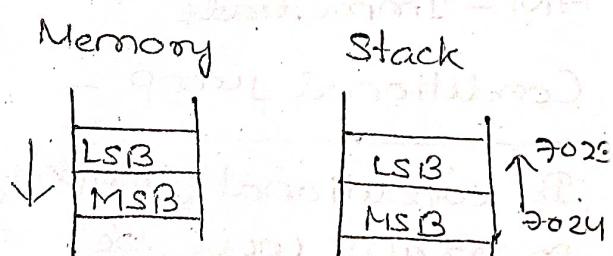
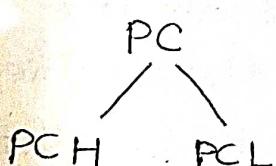
(10 for true - jumping
7 for false - not jumping)

Add mode : Immediate.

CALL :- A call instruction is used to call a subroutine or subprogram. A call instruction may be conditional / unconditional.

Unconditional call :-

① CALL addr :-



$$[PCH] \rightarrow [[SP]-1]$$

$$[PCL] \rightarrow [[SP]-2]$$

$$[SP]-2 \rightarrow [SP]$$

$$[PC] \leftarrow \text{addr}$$

L : 3 , M/C : 5 (write) T : 18 A: Immediate
2 byte (6, 3, 3, ...)

Conditional call :-

In case of conditional call a condition is specified over which the call operation will be held. If the cond' is true, then the processor will call the subroutine, otherwise the subroutine is not called. The conditions are specified according to the status flag bits (except AC).

- (i) CC addr — CS = 1
- (2) CNC addr — CS = 0
- (3) CZ " — Z = 1
- (4) CNZ " — Z = 0
- (5) CP " — S = 0
- (6) CM " — S = 1
- (7) CPE " — P = 1
- (8) CPO " — P = 0

RET :- (Return)

A ret instruction is always associated with a call instruction. This instruction is used to transfer the control from subprogram to main program and that's why each subprogram or subroutine must end with a return statement.

$$[PCL] \leftarrow [[SP]]$$

$$[PCH] \leftarrow [[SP]+1]$$

$$[SP] \leftarrow [SP]+2$$

L : 1

M/C : 3 bytes including self + return part

T : 10

A : Implicit . Reg. addr .

A return statement may be conditional / unconditional.

Unconditional ret — RET

Conditional ret — Conditions are:

- ① RC — CS = 1
- ② RNC — CS = 0
- ③ RZ — Z = 1
- ④ RNZ — Z = 0
- ⑤ RP — S = 0
- ⑥ RM ← S = 1
- ⑦ RPE — P = 1
- ⑧ PPO — P = 0

RST n ($n = 0, \dots, 7$) (Restart)

This is a software interrupt provided by 8085 for the programmers to interrupt the processor during program execution.

The interrupts are classified as —

Vectorized / nonvectorized. In case of a vector interrupt a specific call location is assigned by the processor where the interrupt service subroutine (ISS) is present for the corresponding interrupt.

For a nonvector interrupt there is no specific call location for ISS.

In 8085, INTR is the only nonvectorized interrupt.

The call location for software interrupt is as specified —

Interrupt

Call location

RST0	0000H
RST1	0008H
RST2	0010H
RST3	0018H
RST4	0020H
RSTS	0028H
RST6	0030H
RST7	0038H

Call location for interrupt of nth pin is 8n

PCHL (Jump to the addr. specified by HL) :-

$$[PC] \leftarrow [HL]$$

or

$$[PCH] \leftarrow [H]$$

$$[PCL] \leftarrow [L]$$

L : 1 M.C : 1

T : 6 A.M : Reg. addressing

Slack I/O & machine control instructions :-

I/O instructions -

IN <Port address> 8 bit.

ex - IN 80H

$$[A] \leftarrow [addr]$$

L : 2

M/C : 3

T : 10

A.M : Direct

OUT <port address> :-

$[A] \rightarrow [addr]$

Stack operations :-

(i) PUSH π_p :-

ex - PUSH D

$[[SP]-1] \leftarrow [D] \text{ or } [\alpha_h]$

$[[SP]-2] \leftarrow [E] \text{ or } [\alpha]$

$[SP] \leftarrow [SP]-2$

LSB E	9033
MSB D	9034
	9035

L : 1

MVC : 3

T : 12 (6, 3, 3)

A : reg. addr.

(ii) PUSH PSW :-

$[A] \swarrow \searrow [F]$

$[[SP]-1] \leftarrow [A]$

$[[SP]-2] \leftarrow [F]$

(iii) POP π_p :-

$[\alpha_l] \leftarrow [[SP]]$

$[\alpha_h] \leftarrow [[SP]+1]$

$[SP] \leftarrow [SP]+2$

C	9033
B	9034
	9035

Push : Before After

Pop : After Before.

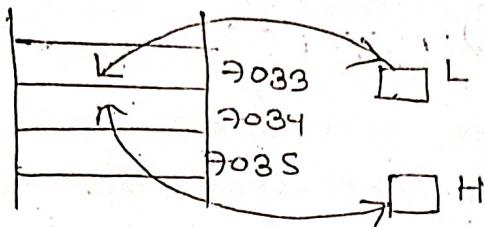
(iv) POP PSW :-

$$[F] \leftarrow [[SP]]$$

$$[A] \leftarrow [[SP]+1]$$

$$[SP] \leftarrow [SP]+2$$

XTHL (^{Content of} Exchange stack top with HL)



$$[L] \leftrightarrow [[SP]]$$

$$[H] \leftrightarrow [[SP]+1]$$

L : 1

M/C : 5 (2 read, 2 write +1)

T - 16

Add - Reg.

SPHL :- (Move content of H-L pair to stack
pointer)

$$[SP] \leftarrow \{ [H-L] \}$$

L : 1

M/C : 1

T : 4

Add : Reg.

Machine control instructions

① HLT : End of the execution.

② EI : Enable interrupt.

③ This instruction enables all the interrupts except TRAP.

- ③ DI - Disable interrupt except TRAP.
- ④ STM - Set interrupt mask.
- ⑤ RIM - Read interrupt mask.
- ⑥ NOP - No operation.

Instruction set of 8086 :-

① Data transfer instructions :-

① Mov mem/reg , 8 bit/16 bit data :

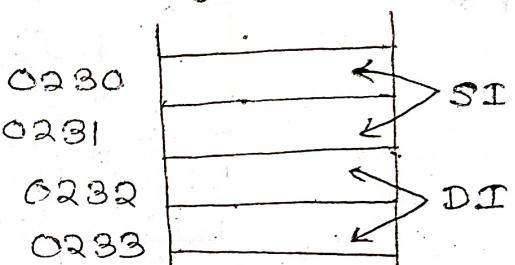
Mov mem/reg , reg/mem .

Mov reg/reg

② LDS reg , mem - This instruction loads a word (16 bit) from the specified mem. loc into the specified register. Furthermore, it also loads a word from the next 2 memory locations into DS registers.

i.e. LDS SI , [0230]

↓
(load data segment)



③ LES reg(16 bit) , mem - This construction will load the 16 bit reg. with 1st 2 memory content & also loads the extra segment reg. with the next 2 memory contents.

④ LEA reg, mem :- (load effective addr.)

The offset of the mem. is calculated & loaded to the 16 bit reg.

⑤ LAHF :- (load AH reg. with content of flag). The lower 8 bits of flag reg. are loaded to reg. AH.

⑥ SAHF :- (store AH reg. to flag)

Content of AH reg. is stored in the lower 8 bit flag.

⑦ XCHG reg :- Exchange content of reg with AX.
i.e. XCHG BX.

⑧ XCHG reg ,reg/mem :-

⑨ PUSH reg(16bit)/memory
PUSH seg reg.

PUSHF — Push the content of flag reg.

POP reg(16bit)/memory

POP seg reg.

POPF.

⑩ IN AL/AX , 8 bit port addl .

IN AL/AX ,DX — Holds the address of I/O.

OUT 8 bit port add , AL/AX

OUT DX , AL/AX .

⑪ XLAT or XLATB — (Translate)

When this instruction is used , content of [AL] and content of [BX] are added to give the desired offset address of look of table.

The content of memory location having the offset address AL+BX is move to AL. This instruction is used for code conversion..

Arithmatic Instructions:-

① ADD reg ,reg .

② ADD mem/reg , reg/mem

③ ADD mem/reg ; data(8/16).

④ SUB reg ,reg .

⑤ SUB mem/reg , reg/mem

⑥ sub mem/reg , data (8/16)

⑦ MUL mem/reg — The content of specified reg/mem if 8 bit length then multiplied with AL and the result is obtained in AX. If the content of specified mem/reg is 16 bit then it is multiplied with AX and the result is obtained in [DX][AX].
16MSB 16LSB.

⑧ IMUL mem/reg — This construction is used for multiplication of 8bit / 16 bit signed no. specified by mem/reg. with the content of AL/AX. The result is obtained in the same register as in case of MUL.

⑨ DIV mem/reg — Unsigned division.

$$[] \times [AL] \rightarrow [AX]$$

$$[] \times [AX] \rightarrow [AX][DX]$$

$$\frac{[AX]}{[]} \rightarrow [AL] \quad \frac{[DX][AX]}{[]} \rightarrow [AX]$$

⑩ IDIV mem/reg — Signed no. division.

⑪ INC reg/mem — Increment.

⑫ DEC reg/mem.

DAA (Decimal adjust accumulator) after

BCD addition — 8 bit construction and works on the content of AL.

DAS (Decimal adjust acc) after BCD SUB

Logical Instruction

- ① AND mem/reg , data(8/16)
- ② AND mem/reg , reg/mem
- ③ AND reg/reg .

Instead of AND , OR & XOR instructions can also be written .

- ④ NOT mem/reg - It's complement of the content of mem/reg.

TEST mem/reg , data(8/16)

TEST reg , mem/reg —

Performs logical AND operation of a specified operand with another operand , the operation is (8/16)bit according to the result only the flag bits are affected (S,Z,P) . The result is then discarded .

Rotate Instruction

RCL reg/mem , count/CL .

↳ (Rotate left with carry).

This instruction will rotate all the bits of operand (reg/mem) towards left through carry flag by the specified count .

ex - RCL BH , 2 .

RCL BH , CL .

RCR reg/mem , count/CL

Rotate count times towards right with carry .

ROR reg/mem, count / CL — Rotate right
not through carry.

ROL reg/mem, count / CL — Rotate left not
through carry.

Shift instruction :-

① SAL / SHL mem/reg, count / CL —
(Shift left by count times with LSB = 0)

1010 → 0100 → 1000 → 0000

② SAR mem/reg, count / CL —
(Shift right by count times with MSB = 1)

③ SHR mem/reg, count / CL —
(Shift right by count times with MSB = 0)

String manipulation instruction :- L.Q

A series of data bytes or words available in memory at consecutive locations to be referred collectively or individually, are known as byte strings or word strings.

① REP / RPT — This instruction is used as a prefix to other instructions. The instruction to which REP prefix is provided is executed repeatedly until the CX register becomes zero (at each iteration the CX value is decremented by 1).

② REP / REPZ — Repeat if equal. i.e. ($CX \neq 0$ or Z flag = 1)

③ REPNE / REPNZ — Repeat if not equal, not zero
i.e. $CX \neq 0$
 $ZF = 0$

④ ~~MOVSB / MOVSC~~ — (Moving string bytes or string words) stored in a set of consecutive

$[DS:SI]$, $[ES:DI]$
Source string destination string

memory locations to another destination.

The starting addr. of the source string is specified by $[DS:SI]$ and that of destination string is specified by $[ES:DI]$.

~~Quop~~ to copy a string of 100 bytes from a location 51020H to 62540H.

<u>Source</u>	<u>Dest^n</u>
Seg - 5000 (DS)	6000 (ES)
Off - 1020 (SI)	2540 (DI)

$CX = 64$ (100 decimal)

MOV AX, 5000

MOV DS, AX

MOV AX, 6000

MOV ES, AX

MOV SI, 1020

MOV DI, 2540

MOV CX, 64

CLD

REP MOVSB HLT

⑤ CMPSB / CMPSW - compare string bytes

Or string words. This instruction is used to compare 2 string bytes or words; length must be stored in CX. If the both the byte or word strings are equal, then zero flag is set. The source & destination string are specified by [DS:SI] & [ES:DI].

Q. WAP to compare 2 strings words of 100 bytes length and if both are equal increment the register BH. The string addresses are given as -

24500, 2A000

100 byte \Rightarrow 50 words.

[2000:4500] - source

CX = 32 in words

[2000 : A000] - dest

1 word = 2 bytes

MOV AX, 2000

MOV DS, AX

MOV ES, AX

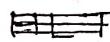
MOV SI, 4500

MOV DI, A000

MOV CX, 32

CLD

REP CMPSW



JNZ X

INC BH

X:HLT

} To increment.

SCASB / SCASW (Scan string byte/word)

This instruction is used to scan a string of bytes or words for an operand byte or word specified in the register AL or AX. The string is pointed by [ES:DI] register pair.

The length of string is stored in CX whenever a match is found to the specified operand in the string execution stops and zero flag is set.

LCDSB / LCDSW :- (Load string byte/word)

This instruction loads the AL or AX register by the content of string pointed by [DS:SI] register pair. The SI is modified automatically depending on the direction flag i.e. incremented/decremented by 1 or 2 according to the byte or word operation. (string ~~dest~~^{start}, reg ~~src~~^{source})

STOSB / STOSW :- (Store string byte/words)

Store from AX or AL to string pointed by [ES:DI]. (String - ~~source~~^{start}, reg ~~dest~~^{source}).

LOOP disp :- Jump to the specified level until CX becomes zero.

LOOPZ / LOOPE disp :-

Cond. for exec CX ≠ 0, Z = 1

LOOPNZ / LOOPNE disp :-

Cond. for exec CX ≠ 0, Z = 0.

Flag manipulation instruction :-

CLC - clear carry

STC - set "

CLD - clear direction

STD - set "

CLI - clear interrupt

STI - set "

CMC - complement carry

PROCESS CONTROL INSTRUCTION :-

→ HLT

→ NOP - no operation

→ ESC - escape

Makes the bus free for external devices

→ WAIT - Related to TEST pin & executes wait instruction.

→ LOCK - It locks the system bus.

BRANCH INSTRUCTION :-

Near

far

Intersegment.

JA 8 bit :- (jump above)

CY=0

JNBE 8 bit :- jump if not below or equal.

JB disp :- Jump if below.

CY = 1

JNAE :- jump if not above or equal

JC disp :- (Jump with carry)

JAE 8bit :- jump if above or equal

CY = 0, Z = 1

JNB 8bit :-

JNC 8bit

JBE 8bit - jump if below or equal.

CY = 1, Z = 1

JNA 8bit

JCXZ disp :- jump if CX = 0

JE disp :- jump if equal.

Z = 1

JNAB disp -

JZ disp -

JG disp - jump if greater = JA

JGE = JAE disp

disp

JL disp = JB disp

JLE disp = JBE disp

JNC disp = JNA disp,

JNE disp - Z = 0

JNZ

JUMP add

Classification of control transfer instruction

The control transfer instruction are of 3 types which are classified according to the specified operand and the distance of jump.

These are -

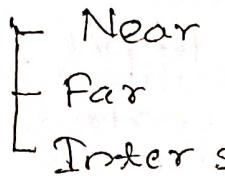
- ① Intra segment Relative Near Jump
- ② Intra segment relative far jump.
- ③ Inter segment direct jump.

- ① If the specified operand is 8 bit signed no. then the control will jump +127 in forward and a maximum -127 bytes in reverse direction. Here the transfer of control within a range of 256 bytes in the ~~current~~ segment. Hence it is referred as intra segment relative near jump.
- ② If a 16 bit offset is specified in the operand then the control can be transferred to any memory location inside the current segment in a range of 64KB. Hence it is referred as intra segment relative far jump.
- ③ If the specified operand is 32 bit (16 bit seg : 16 bit offset) then the control can be transferred to any

memory location inside any segment in the total memory map of 1MB. Referred as inter segment direct jump.

JMP mem/reg :- The 16bit offset is present inside the memory or register indirectly to which the control is to be transferred.

CALL add



CALL reg/mem :- call a reg or mem.

RET :- Return from subroutine to main prog.

IRET :- (Return from ISS)

RET 16bit disp :-

Interrupt Instructions :-

INT n - ($n = 0, 1, 2 \dots 255$)

INTO - (Interrupt on overflow)

Difference betw. 8085 and 8086

8085

8086

→ 8 bit

→ 16 bit

→ Address capacity =
64K(2^{16})

→ 1M(2^{20})

→ Uniprocessor

→ Multiprocessor

→ 3-5 MHz

→ 5/8/10 MHz

→ N-Mos

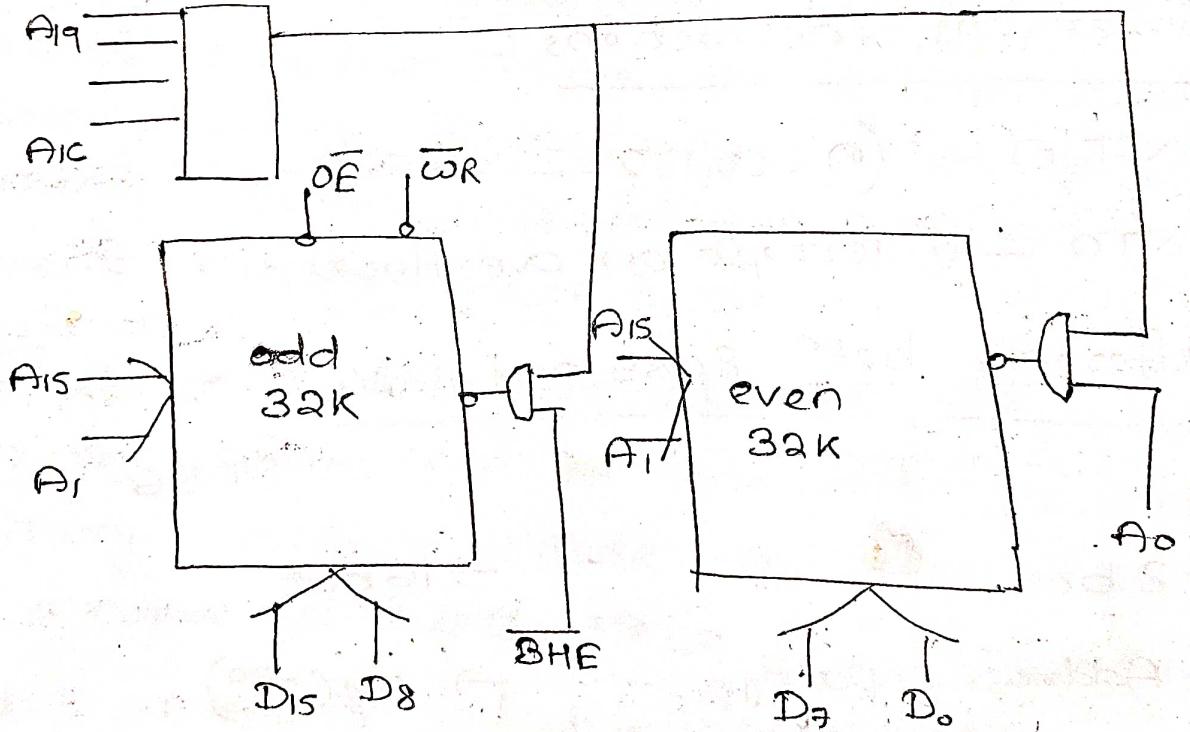
→ HMOS

→ No segmentation

→ Segmentation

- No pipelining → Pipelining
- 1 funcn unit. → 2 functional unit (BIU, EU)
- No queue.
- Queue.
- Physical add. → Segment & offset are used as logical add.
- 8 bit flag. → 16 bit flag.
- No control flags. → Control Flags
- S/W interrupt - 8 → S/W interrupt - 256
H/cn " - 5 H/w " - 2

Assemblers Directives :-



Assembler Directives :-

An assembler is a program used to convert an assembly language program into the equivalent machine code modules which may further be converted to executable codes.

The assembler decides the address of each level and substitutes the values for each of the constants and variable. Then it forms the machine code for the mnemonics and data in the assembly language program.

While doing these things the assembler may findout syntax errors, logical errors or programming errors are not found out by the assembler.

For completing all these tasks an assembler needs some hints from the programmer i.e. required storage for a particular constant or variable, logical names of the segments, types of different routines and modules, end of file etc.

These type of hints are given to the assembler by using some predefined alphabetical strings called Assembler directives.

Assembler directives help the assembler to correctly understand the assembly lang. prog. to prepare codes.

DB (Define Byte)

The DB directive is used to reserve byte or bytes of memory locations in the available memory while preparing the executable file & this directive directs the assembler to allocate the specified no. of bytes to the datatype that may be a constant, variable, string etc.

ex - ROLL DB 01, 02, 03, 04.

This statement directs the assembler to reserve four memory locations for a list named ROLL and initialises them with the specified four values.

MESSAGE DB 'HELLO'

DW (Define word)

Reserve word or words. Capacity is 16 bit or 2 bytes. ex - 1024.

DQ (Define quad word)

ROLL DQ 1234 1111 1024 1111

↓ 4 words

DT (Define ten byte)

10 byte or 5 words.

ASSUME — It is used to assume the logical segment name.

ASSUME CS : CODE

" ES : EXTRA

END

It indicates end of a program.

ENDP

It indicates end of procedure / subroutine
Define byte or define program.

ENDS

(End of segments) — This directive marks the end of logical segments.

DATA SEGMENT

DATA ENDS

EVEN

Align on even memory address.

EQU (equate)

Used to assign a label with a value or symbol.

ex - LABEL EQU 4050

ADDITION EQU ADD.

Intel 8259 PIC

Programmable/priority interrupt controller

- The 8259 is a PIC designed to work with Intel 486 i.e. 8085, 8086 and 8088.
- Function of 8259 :- It can manage 8 interrupts according to the instructions written into the control register. This is equivalent to providing 8 interrupt pins in the processor instead of INTR pin.
 - (i) It vector 8 interrupt anywhere in the memory mapped however all 8 interrupts are spaced after interval of either 4 or 8 locations.
 - (ii) Resolve 8 levels of interrupt priorities in a variety of modes such as -
 - ① fully nested mode.
 - ② Automatic rotation.
 - ③ Specific rotation.
 - (iv) Make each interrupt request individually.
 - (v) Read the status of pending interrupt, inservice interrupt and mask interrupt.
 - (vi) It can setup to accept level triggered mode or the edge triggered mode.
 - (vii) It can be expanded to 64 priority level by cascading 8 additional 8259's.

Pin diagram of 8259 :- The data lines are used to send the control word to the control register.

A₀ - Used to select the control reg ($A_0 = 1$) .

① WR - This pin is used to write the control word in the control register.

② RD - Used to read the status of IRR, ISR and IMR.

IRR - Interrupt req. register.

ISR - Insertive register.

IMR - Interrupt mask register.

③ D₀ to D₇ - This is an 8 bit bidirectional data bus which is used to interface the 8085 with 8259 for transfer of data (to read the status register or to write the control word)

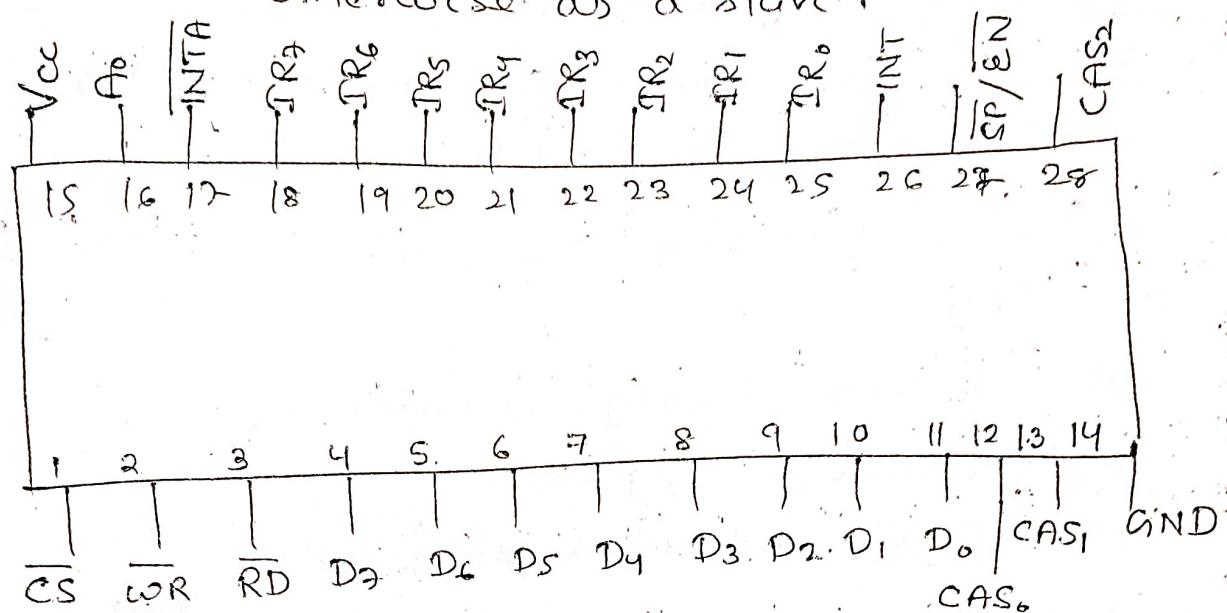
④ CAS₀ to CAS₂ - These are the cascade lines used to select slave 8259's (8 no.s)

⑤ INT ; INTA - These are 2 pins connected to INTR and INTA of 8085 to send the interrupt req. on behalf of the 8 I/O devices that are connected to IR₀ to IR₇.

⑥ IR₀ to IR₇ - These are 8 h/w interrupt pins, where 8 I/O devices can be connected and send interrupt requests simultaneously.

⑦ SP / EN - (Slave programmer or enabled buffer)

This pin is used to specify whether the 8259 is acting as a master or as a slave. A logic 1 indicates the 8259 is acting as a master otherwise as a slave.



In the funcⁿ block diag. of 8259 there are basically 3, 8 bit registers.

① IRR — This is an 8 bit register which keeps the status of the interrupts currently being requested through the pins IR₀ to IR₇. If a bit in IRR is 1, then it indicates the corresponding interrupt is requested.

ex - if PRR contains 83H then it indicates there is a request through IR₇, IR₁ and IR₀.

② ISR — It is an 8 bit register which keeps the status of that interrupt which is currently being in service.

from the 8 bit register which is used to make the h/w interrupt request pins (IR₀-IR₇) for ex - If D₀ = 1 in IMR , it indicates IR₀ pin is masked off .

Operation of 8259

- The req. from I/O devices is provided to IRR through the pins IR₀ to IR₇.
- The priority resolver will findout which interrupt carries the highest priority according to the selected priority mode.
- The ISR will get the highest prior interrupt from priority resolver and then on behalf of that the 8259 will send an interrupt req. through the INT pin through 8085 or 8086.
- After getting a request through INT, the 8085 will send an acknowledgement to 8259 on INTA pin.
- After getting INTA, the 8259 will generate an RST-n code and call the corresponding vector location to execute the ISS.

Priority modes —

- ① Automatic rotation mode — In automatic rotation the interrupt after being solved gets the lowest priority by 8259 .

$IR_7 \cdot IR_6$

0

1

IR_0

2

If we have a req. to IR_6 and IR_5 , IR_6 will be implemented first but it will get the lowest priority.

② Fully nested mode — In this mode all

the interrupt requests are assigned a priority in ascending order or in descending order. i.e. If IR_0 carries the highest priority & then IR_7 carries the lowest priority & the priority is fixed throughout the process.

The priority is fixed i.e. either ascending or descending order.

③ Specific rotation mode — ^{Special mask} _{Poll mode}

This mode is similar to automatic rotation mode except that the user can select any IR . For the lowest priority and can fix the all other priorities. In this mode we can randomly assign the priorities to any interrupt.

$IR_7 \cdot IR_6 \cdot IR_5 \cdot IR_4 \cdot IR_3 \cdot IR_2 \cdot IR_1 \cdot IR_0$

3 6 4 5 2 1 7 0

3 5 4 7 2 1 6 0 — if req.

through this IR_4

Block diag

8257 - programmable DMA controller

DMA - Direct memory access betⁿ I/O & memory without using CPU.

Programmable DMA controller

- As data transfer betⁿ I/O and mem. through acc. is time consuming, DMA data transfer is referred.
- In DMA, data are directly transferred from I/O devices to memory or from memory to I/O devices.
- for DMA, data & add. comes under the control of peripheral devices which wants DMA data transfer.
- The CPU has to release control of buses for DMA operation on req. of I/O device.
- For DMA data transfer, I/O device must have its own reg. to store the byte count and the mem. add.
- It must be able to generate control signal req. for DMA data transfer. But such facilities are not available with I/O devices. Thus DMA controllers are used for the interfacing of I/O devices to the CPU for DMA data transfer.
- 8257 has 4 DMA channels.
 - Each channel has 2 reg.
 - (1) DMA addr. reg - 16 bit
 - (2) Terminal count reg - 16 bit.

8257 has 3 modes of operation.

- 1) Read
- 2) Write
- 3) Verify

2 MSB's of terminal count reg. used to identify the modes of operation.

DIS	D14
0	0 — verify
0	1 — write
1	0 — read

→ There are 4 DMA req. lines.

i.e. DRQ₀ — DRQ₃.

→ The I/O devices gives DMA req. to 8257 through DRQ₀ to DRQ₃.

→ \overline{DACK}_0 — \overline{DACK}_3 are DMA acknowledgement lines.

→ DRQ₀ — highest priority.

DRQ₃ — lowest priority.

→ HRQ — Connected bet' 8257 — 8085 μP. A hold req. is given by 8257 to 8085 to release the buses.

HLDA — 8085 — 8257

TC — Terminal count pin.

When the data transfer is over, there will be a high (1) in TC pin. Then req. from another channel is serviced.

→ 8257 is a 4 channel programmable DMA controller.

→ It is a 40 pin TC package. 4 I/O devices can be interfaced through this device.

→ It can perform read, write & verify operation.

- During read operation data are directly transferred from memory to I/O.
- During write operation data are transferred directly from I/O to memory.
- On receiving a request from an I/O device the 8257 generates a sequential mem. addr which allows the I/O device to read/write directly from or to the memory.
- Each channel on 8257 has 316 bit reg. associated with it.
 - (1) DMA addr reg.
 - (2) Terminal count reg.
- These reg. are initialised before a channel is enabled.
- Initially the DMA addr reg. is loaded with the addr. of 1st memory location to be accessed.
- During DMA operation it stores the next mem. location addr. to be accessed in the next DMA cycles.
- The 14 LSB of the terminal count reg. store no. of bytes to be transferred.
i.e. $2^{14} = 16\text{ KB}$.
- $2^{14} = 16\text{ KB}$ of data can be directly transfer to the mem. from the I/O device and
- The 2, MSB of TC reg indicates the operation which will be performed on that channel.

General operation of 8257 -

- An I/O sends req. for DMA transfer through one of the 4 DRQ lines.
- On receiving DMA req. for data transfer from I/O device 8257 sends the HOLD req. to I/F through HRQ line.
- The 8257 receives hold acknowledgement signal from I/F through HLDA line.
- After receiving hold acknowledgement from the I/F, 8257 sends to the I/O device a DMA acknowledgement through DACK line.
- The mem. add is sent out on the addr and data lines.
- The 8257 sends the 8 MSB's of the mem. addr over the data lines and 8 LSB over the addr lines.
- 8 MSB's of memory are latched into an external cache by using A DSTB signal.
- This signal is similar to ALE in 8085.
- For DMA read operation to control signal MEMR and I/O/w are issued by 8257.
- The MEMR signal enables the addr. memory for reading data from it.

- The I/O/w enables the I/O device to accept data for DMA write operation to control signal.
- MEMW and I/O/R are issued by DMA controller.
- The MEMW enables the addressed mem. for writing data into it.
- The I/O/R enables the I/O device to read the data from the I/O device.
- The byte count is decremented by 1 after the transfer of 1 byte of data.
- When the byte count becomes zero the terminal count goes high indicating that the data transfer using DMA is complete.
- The 4 DMA channels are programmed according to the priority assigned to them DRQ₀ is assigned to highest priority and DRQ₃ is assigned to lowest priority.

Pin configuration of 8257

- DRQ - DRQ₀-DRQ₃ are the DMA req. lines. An I/O device sends its DMA req. on 1 of these lines.
- DACK₀-DACK₃ - These are the DMA acknowledgement lines. The 8259 sends an acknowledgement signal through 1 of these lines informing an I/O device that it has been selected for DMA data transfer.

A₀-A₇ — These are the address lines which are used to carry the 8 MSB's of the 16 bit DMA address.

D₀-D₇ — These are the data lines during the DMA cycle. The 8257 sends the 8 MSB of the mem. addr through these lines at the beginning of DMA cycle. These 8 MSB's are latched and the databus is made available to handle the data transfer during the rest of DMA cycle.

AEN — Address enable

It is an active high o/p signal which is used to disable the address bus, data bus & control bus.

ADSTB — A high signal on this line latches the 8 MSB's of the address which are send on the D₀-D₇ line into an external latch.

CS — chip select signal. In the slave mode it is used. In the slave mode it is activated & in the master mode the CS is internally disable in order to prevent the 8257 from selecting itself.

Reset :- It clears the registers and disables DMA channels.

CLK :- This is connected to clock o/p of 8085 μp.

T_{O/P} — It is used as an o/p pin to write data into peripheral which requires DMA operation in read mode.

DOR — Used to access data from peripheral device whose DMA req. is being serviced in write mode.

ENRW — During a DMA write cycle this pin is used to write data into memory loc. which is selected by the addr. put out by 8257.

ENR — During DMA read cycle ,this pin is used to read data from mem. location which is selected by the address put out by 8257.

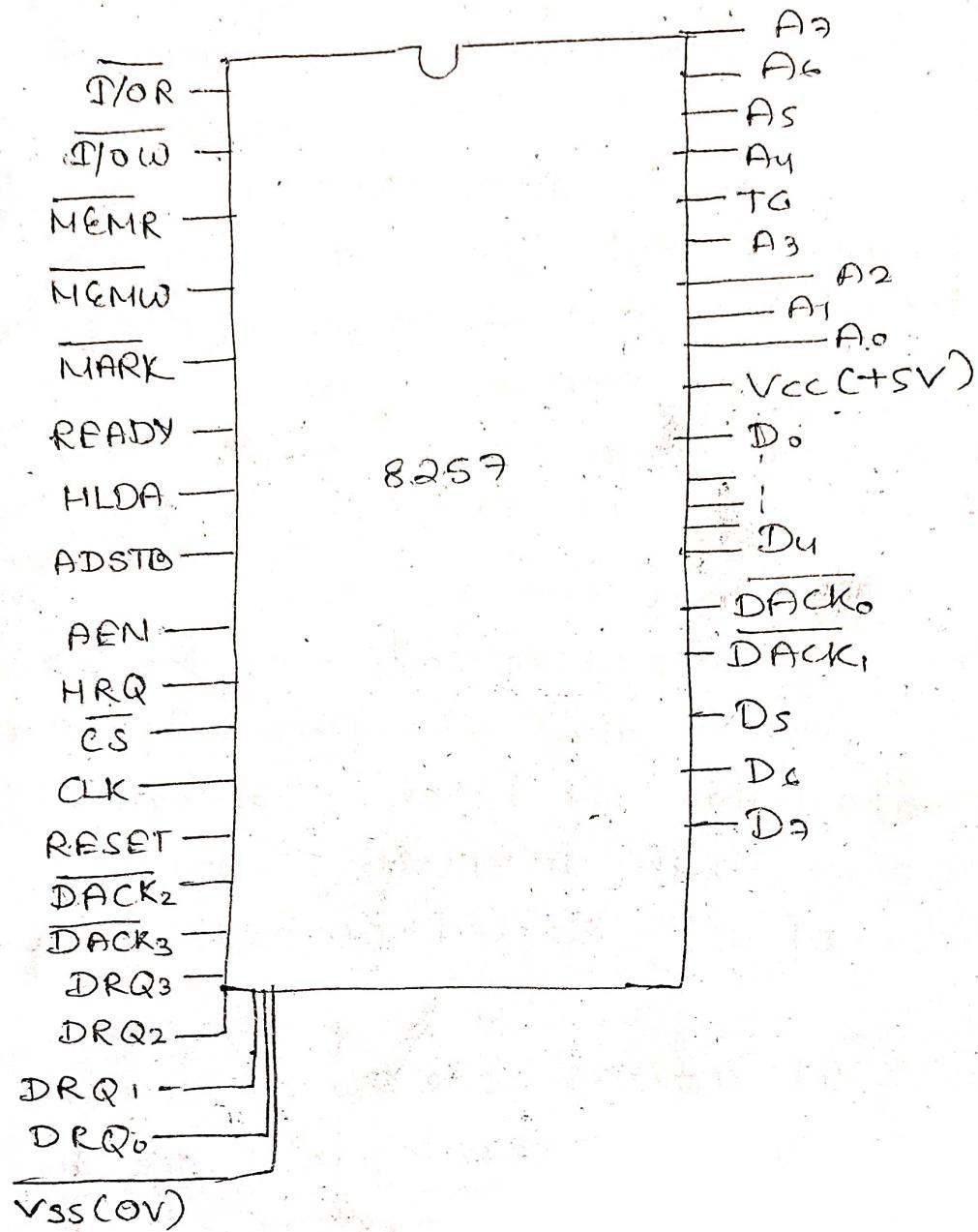
TC — (Terminal count)

A high on this line indicates the peripheral being serviced that the present DMA cycle is the last for the block. The terminal count op becomes high when the contents of the TC reg. of the selected channel is equal to zero.

MARK — It informs the I/O device being serviced that the current DMA cycle is the 128th cycle since the previous mark o/p.

HRQ — through this pin the 8257 request the control of the system buses . It is normally connected to the hold I/p of 8085 μp.

HLDA — The processor informs the 8257 through this pin that it has released the control of buses and the 8257 may take the control over the buses.



Stack of 8051

The stack in a microcontroller can be described as a set of memory locations in the read write memory where data are stored temporarily during prog. execution.

Data are stored and retrieved by using push and pop instruction.

The last memory location of the occupied portion of stack is known as stack top.

The stack pointer in 8051 is of 8 bit wide which means that it can take values from 00 to FF. When microcontroller is powered off stack pointer register contains value 07H.

i.e. The 1st memory location being used for stack is 08H.

In 8051 the RAM location 08H to 1FH total 24 bytes can be used for stack operation if in a program an user wants to access more than 24 bytes of stack we can change stack pointer to point RAM locations 30H to 3FH. This can be done with the help of Mov instruction by writing -

MOV SP, 8bit H,00

MOV R6, #FF11H

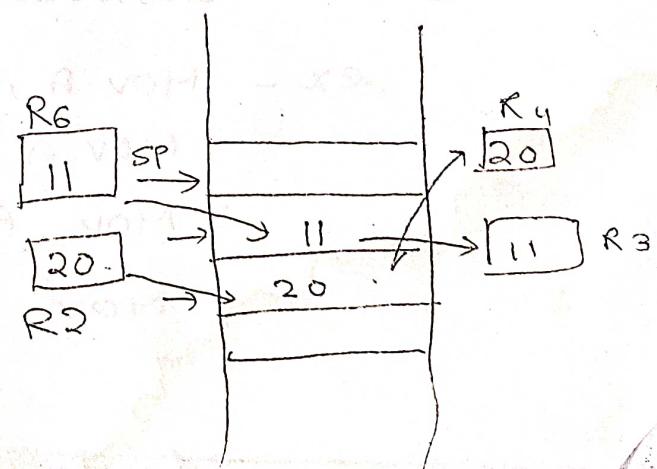
MOV R2, #20H

PUSH R6

PUSH R2

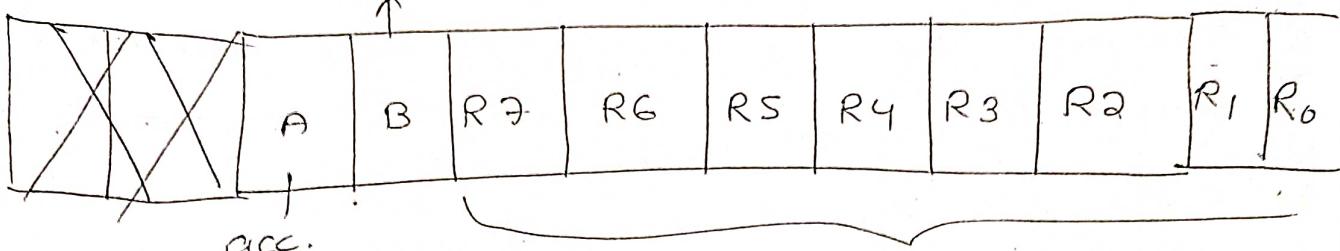
POP R4

POP R3



Registers in 8051

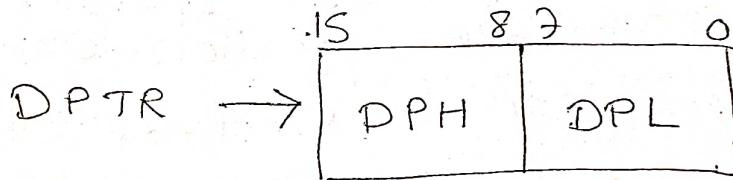
CPR & used for mult & div.



Reg. bank of 9/12/3

DPTR — 16 bit length
(data pointer)

DPTR in 8051 is the only user accessible 16 bit register and is used as a pointer which points to the data. So in external memory access the address of the memory to be accessed is indicated by DPTR register.



Instruction set of 8051 :-

Data transfer instructions :-

① MOV destination, source.

ex - MOV A, RI

MOV A, #25H

MOV A, @ RI — indirect address.

MOV A, PI

Arithmetic instructions -

ADD A, source

ADD A, #25

@ R0

P1

ADDC A, source — source + acc. + carry

SUB A, source

SUBB A, source — A - source - borrow/carry

DA A — decimal adjust acc.

MUL AB

DIV AB

Logical instruction

→ ANL operand1, operand2

↳ AND logical \ A \ RI
(bitwise)

→ ORL op1, op2

→ XRL " ", "

→ CPL operand \ A

→ CMPL

Compare instruction

CJNE

Rotate instruction

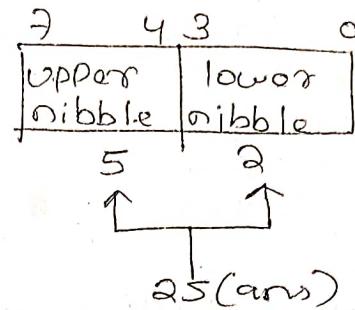
→ RL A

→ RR A

→ RLC A

→ RRC A

Swap A



Single bit instructions

SETB bit — Set the bit.

SETB P2.4

CLRB bit — clear bit

CPLB bit — Complement bit

JB bit, target — jump if bit = 1 to the target add.

JNB bit, target —

JBC bit, target — jump if bit = 1 and clear the target bit and jump to target.

ex — JBC 01, 30H