CSE Learning Hub

Microprocessor & Interfacing
(3160712)

DEGREE
COMPUTER Engineering | SEM: 6
Notes By: Harsh Porwal

Data Transfer Group

1) MOV $R_{Destination}$, R_{Source},

The contents of register R_{Source} is copied into register $R_{Destination}$.

EX: **MOV A, B;** A ← B

Addressing Mode	Flags Affected	Cycles	T-State
Register	None	1 (OP)	4

2) MOV R, M

The contents of memory location pointed by HL pair id copied to register R.

EX: **MOV B, M;** B ← [HL] => B ← M

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	None	2 (OP+MR)	7

3) MOV M, R

The contents of register R is copied into the memory location pointed by HL.

EX: **MOV M, B;** [HL] ← B => M ← B

Addressing Mode	Flags Affected	Cycles	T-State
Register	None	2 (OP+MW)	7

4) MVI R, 8-bit data

The 8-bit data is immediately moved into the register specified in the instruction.

EX: **MVI C, 23H;** C ← 23H

Addressing Mode	Flags Affected	Cycles	T-State
Immediate	None	2 (OP+MR)	7

5) MVI M, 8-bit data

The 8-bit data is immediately moved into memory location pointed by HL pair.

EX: **MVI M, 23H;** [HL] ← 23H

Addressing Mode	Flags Affected	Cycles	T-State
Immediate	None	3 (OP+MR+MW)	10

6) LXI R_p, 16-bit data

The 16-bit data is immediately moved into the register specified in the instruction.

EX: **LXI B, 2300H;** B ← 23H, C ← 00H

Addressing Mode	Flags Affected	Cycles	T-State
Immediate	None	3 (OP+MR+MR)	10

7) LDA 16-bit data

The accumulator is loaded with the content of memory location given in instruction.

EX: **LDA 2300H;** A ← 2300H

Addressing Mode	Flags Affected	Cycles	T-State
Direct	None	4 (OP+MR+MR+MR)	13

8) STA 16-bit data

The accumulator is stored into the memory location given in instruction.

EX: **STA 2300H;** [2300] ← A

Addressing Mode	Flags Affected	Cycles	T-State
Direct	None	4 (OP+MR+MR+MW)	13

9) LHLD 16-bit data

The HL pair is loaded with the contents of the location pointed by the given address and address + 1.

EX: **LHLD 2300H;** L ← [2300], H ← [2301]

Addressing Mode	Flags Affected	Cycles	T-State
Direct	None	5 (OP+MR+MR+MR+MR)	16

10) SHLD 16-bit data

The HL pair is stored into the location pointed by the given address and address + 1.

EX: **SHLD 2000H;** [2000] ← L, [2001] ← H

Addressing Mode	Flags Affected	Cycles	T-State
Direct	None	5 (OP+MR+MR+MW+MW)	16

11) LDAX R_p

The accumulator is loaded with the contents of memory location pointed by value of the given register.

EX: **LDAX B;** A← [BC], if BC = 2000, A gets the value from location 2000.

A ← [2000]

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	None	2 (OP+MR)	7

12) STAX R_p

The accumulator is stored into the location pointed by value of register given in instruction.

EX: **STAX B;** [BC]← A, if BC = 2000, Location 2000 will gets the value from register A. [2000] ← A

Addressing Mode	Flags Affected	Cycles	T-State

Indirect	None	2 (OP+MW)	7

13) PCHL

The program counter gets the contents of the HL register pair.

This statement causes a branch in the sequence of the program.

EX: **PCHL;** PC ← HL

Addressing Mode	Flags Affected	Cycles	T-State
Register	None	1 (OP)	6

14) SPHL

The stack pointer gets the contents of the HL register pair.

This statement relocates the stack in the 64KB memory.

EX: **SPHL;** SP ← HL

Addressing Mode	Flags Affected	Cycles	T-State
Register	None	1 (OP)	6

15) XCHG

This instruction exchanges the content of HL pair and DE pair.

EX: **XCHG;** HL ⇔DE

Addressing Mode	Flags Affected	Cycles	T-State
Register	None	1 (OP)	4

16) XTHL

This instruction exchanges DE pair with the contents of location pointed by SP and SP+1.

EX: **XTHL;** HL ⇔[SP] and [SP+1], If [SP]=2000 then
L⇔[2000] and H⇔[2001]

Addressing Mode	Flags Affected	Cycles	T-State
Register	None	5 (OP+MR+MR+MW+MW)	16

Arithmetic Group

1) ADD R

This instruction adds the contents of register R with the accumulator, stores result in the accumulator.

EX: **ADD B;** A ← A+B

Addressing Mode	Flags Affected	Cycles	T-State
Register	All	1 (OP)	4

2) ADD M

This instruction adds the contents of memory location pointed by HL, with the accumulator, stores result in the accumulator.

EX: **ADD M;** A ← A+[HL]

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	All	2 (OP+MR)	7

3) ADI 8-bit data

This instruction adds the immediate data with the accumulator and result store in A.

EX: **ADI 25;** A ← A+25

Addressing Mode	Flags Affected	Cycles	T-State
Immediate	All	2 (OP+MR)	7

4) ADC R

This instruction adds the content of register R with the accumulator and also adds the carry flag, and store the result in the accumulator.

EX: **ADC B;** A ← A+B+Cy

Addressing Mode	Flags Affected	Cycles	T-State
Register	All	1 (OP)	4

5) ADC M

This instruction adds the content of memory location pointed by HL pair with the accumulator and also adds the carry flag, and store the result in the accumulator.

EX: **ADC M;** A ← A+[HL]+Cy

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	All	2 (OP+MR)	7

6) ACI 8-bit data

This instruction adds the immediate data with the accumulator and also adds the carry flag, and store the result in the accumulator.

EX: **ACI 25;** A ← A+25+Cy

Addressing Mode	Flags Affected	Cycles	T-State
Immediate	All	2 (OP+MR)	7

7) SUB R

This instruction subtracts the contents of register R with the accumulator, stores result in the accumulator.

EX: **SUB B;** A ← A-B

Addressing Mode	Flags Affected	Cycles	T-State
Register	All	1 (OP)	4

8) SUD M

This instruction subtracts the contents of memory location pointed by HL, with the accumulator, stores result in the accumulator.

EX: **ADD B;** A ← A-[HL]

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	All	2 (OP+MR)	7

9) SUI 8-bit data

This instruction subtracts the immediate data with the accumulator and result store in A.

EX: **ADI 25;** A ← A-25

Addressing Mode	Flags Affected	Cycles	T-State
Immediate	All	2 (OP+MR)	7

10) SBB R

This instruction subtracts the content of register R with the accumulator and also adds the carry flag during subtraction if a borrow is taken, and store the result in the accumulator.

EX: **ADC B;** A ← A-B+Cy

Addressing Mode	Flags Affected	Cycles	T-State
Register	All	1 (OP)	4

11) SBB M

This instruction subtracts the content of memory location pointed by HL pair with the accumulator and also adds the carry flag during subtraction if a borrow is taken, and store the result in the accumulator.

EX: **ADC M;** A ← A-[HL]+Cy

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	All	2 (OP+MR)	7

12) SBI 8-bit data

This instruction subtracts the immediate data with the accumulator and also adds the carry flag during subtraction if a borrow is taken, and store the result in the accumulator.

EX: **ACI 25;** A ← A-25+Cy

Addressing Mode	Flags Affected	Cycles	T-State
Immediate	All	2 (OP+MR)	7

13) INR R

This instruction increments the content of the specified register.

EX: **INR B;** B ← B+1

Addressing Mode	Flags Affected	Cycles	T-State
Register	All except carry	1 (OP)	4

14) INR M

This instruction increments the content of memory location pointed by HL.

EX: **INR M;** M ← M+1, [HL] ← [HL]+1

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	All except carry	1 (OP)	4

15) INX R_p

This instruction increments the content of the specified register pair.

EX: **INX B;** B ← BC+1, If [BC]=3000 then [BC] becomes 3001.

Addressing Mode	Flags Affected	Cycles	T-State
Register	None	1 (OP)	6

16) DCR R

This instruction decrements the content of the specified register.

EX: **DCR B;** B ← B-1

Addressing Mode	Flags Affected	Cycles	T-State
Register	All except carry	1 (OP)	4

17) DCR M

This instruction decrements the content of memory location pointed by HL pair.

EX: **DCR M;** M ← M-1, [HL] ← [HL]-1

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	All except carry	3 (OP+MR+MW)	10

18) DCX R_p

This instruction decrements the content of the specified register pair.

EX: **DCX B;** B ← BC-1, If [BC]=3001 then [BC] becomes 3000.

Addressing Mode	Flags Affected	Cycles	T-State
Register	None	1 (OP)	6

19) DAD R_p

This instruction adds the contents of the given register pair with HL pair.
The result is stored in HL pair.

EX: **DAD B;** HL ← HL+BC,

Addressing Mode	Flags Affected	Cycles	T-State
Register	Only Carry	3 (OP+BI+BI)	10

20) DAA

This instruction is used to get the answer in BCD form.
It adjusts the result of an addition operation to make the addition work like decimal addition.
It is implied addressing and works strictly on A register.
It checks the nibbles of A as follows:
If LN>9 or AC=1 then add 06H
If HN>0 or CY=1 then add 06H

Addressing Mode	Flags Affected	Cycles	T-State
Implied	All	1 (OP)	4

Logic Group

1) ANA R

Logically AND the contents of the specified register with accumulator, store result in accumulator.

EX: **ANA B;** A ← A AND B,

Addressing Mode	Flags Affected	Cycles	T-State
Register	All	1 (OP)	4

2) ANA M

Logically AND the contents of the memory location pointel by HL pair, store result in accumulator.

EX: **ANA M;** A ← A AND M,

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	All	2 (OP+MR)	7

3) ANI 8-bit data

Logically AND the immediate 8-bit data, store result in accumulator.

EX: **ANI 25;** A ← A AND 25,

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	All	2 (OP+MR)	7

4) ORA R

Logically OR the contents of the specified register with accumulator, store result in accumulator.

EX: **ORA B;** A ← A OR B,

Addressing Mode	Flags Affected	Cycles	T-State
Register	All	1 (OP)	4

5) ORA M

Logically OR the contents of the memory location pointel by HL pair, store result in accumulator.

EX: **ORA M;** A ← A OR M,

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	All	2 (OP+MR)	7

6) ORI 8-bit data

Logically OR the immediate 8-bit data, store result in accumulator.

EX: **ANI 25;** A ← A OR 25,

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	All	2 (OP+MR)	7

7) CMP R

Compares the contents of register R and accumulator.
This instruction performs A-R.
Important to remember that result of this comparisons is NOT stored in accumulator.

EX: **CMP B;** A ← A-B,

Addressing Mode	Flags Affected	Cycles	T-State
Register	All	1 (OP)	4

Conclusion	Zero Flag 'Z'	Carry Flag 'C'
A > B	0	0
A = B	1	0
A < B	0	1

8) CPI 8-bit data

Compare the immediate 8-bit data, store result in accumulator.

EX: **CPI 25;** A ← A - 25,

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	All	2 (OP+MR)	7

9) STC

Sets the carry flag

EX: Cy ← 1

Addressing Mode	Flags Affected	Cycles	T-State
Implied	Only Carry	1 (OP)	4

10) CMC

Complements the carry flag

EX: Cy ← Cy

Addressing Mode	Flags Affected	Cycles	T-State
Implied	Only Carry	1 (OP)	4

11) CMA

Complements the Accumulator

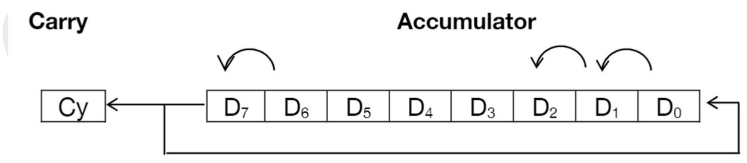
EX: A ← 1's Complement of A

Addressing Mode	Flags Affected	Cycles	T-State
Implied	None	1 (OP)	4

12) RLC

The content of accumulator is rotated left by 1.

The MSB goes to carry and LSB.

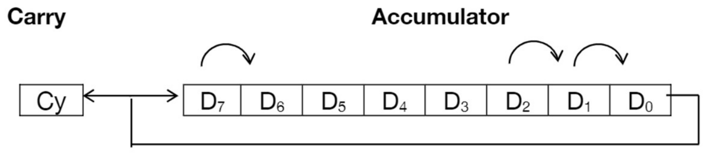


Addressing Mode	Flags Affected	Cycles	T-State
Implied	Carry	1 (OP)	4

13) RRC

The content of accumulator is rotated right by 1.

The LSB goes to carry and MSB.

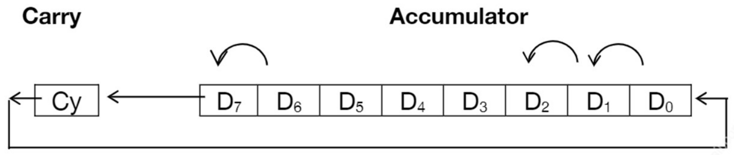


Addressing Mode	Flags Affected	Cycles	T-State
Implied	Carry	1 (OP)	4

14) RAL

The content of accumulator is rotated left by 1.

The MSB goes to carry and the Carry goes to LSB.

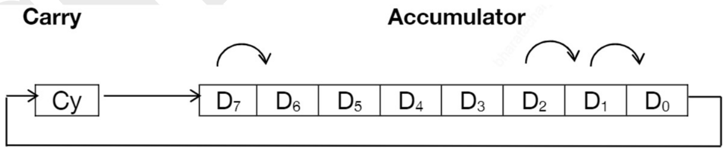


Addressing Mode	Flags Affected	Cycles	T-State
Implied	Carry	1 (OP)	4

15) RAR

The content of accumulator is rotated right by 1.

The LSB goes to carry and the Carry goes to MSB.



Addressing Mode	Flags Affected	Cycles	T-State
Implied	Carry	1 (OP)	4

Branch Control Group

1) JMP 16-bit address (Unconditional Jump)

Loads PC with the 16-bit address specified in the instructions.

EX: **JMP 2500;** PC ← 2500,

Addressing Mode	Flags Affected	Cycles	T-State
Immediate	None	3 (OP+MR+MR)	10

2) $J_{Condition}$ 16-bit address (Conditional Jump)

It is same as Unconditional JUMP except that the action takes place only if the condition is true.

EX: **JZ 2500;** PC ← 2500, if z = 1

Addressing Mode	Flags Affected	Cycles	T-State
Immediate	None	2/3	7/10

Condition	Description	True if:
NZ	No zero	Z = 0
Z	Zero	Z = 1
NC	No Carry	C = 0
C	Carry	C = 1
PO	Parity Odd	P = 0
PE	Parity Even	P = 1
P	Plus	S = 0
M	Minus	S = 1

3) CALL 16-bit address (Unconditional Call)

Loads PC with the 16-bit address specified in the instruction. Before doing it also pushes the current PC into stack.

EX: **JMP 2500;** SP ← SP-1
[SP] ← PC_H
SP ← SP-1
[SP] ← PC_L
PC ← 2500

Addressing Mode	Flags Affected	Cycles	T-State
Immediate	None	5 (OP+MR+MR+MW+MW)	18

4) CCondition 16-bit address (Conditional Call)

It is same as Unconditional call except that the action takes place only if the condition is true.

EX: **CNZ 2500;** If Z = then
SP ← SP-1
[SP] ← PC_H
SP ← SP-1
[SP] ← PC_L
PC ← 2500

Addressing Mode	Flags Affected	Cycles	T-State
Immediate	None	2/5	9/18

5) RET (Unconditional Return)

This instruction is written end of the sub-routine and enables the control to go back to main program. We enter a subroutine using CALL instruction in which we push the return address into the stack.
In RET instruction we do the opposite. We POP the return address from the stack into PC

EX: **RET;** $PC_L \leftarrow [SP]$
$SP \leftarrow SP + 1$
$PC_H \leftarrow [SP]$
$SP \leftarrow SP + 1$

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	None	3 (OP+MR+MR)	10

6) RCondition (Conditional Return)

It is same as Unconditional return except that the action takes place only if the condition is true.

EX: **RC;** If C = 1 then
$PC_L \leftarrow [SP]$
$SP \leftarrow SP + 1$
$PC_H \leftarrow [SP]$
$SP \leftarrow SP + 1$

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	None	1/3	6/12

7) RST$_n$ (Restart n)

It is very similar to CALL instruction. Here the branch address, instead of being specified directly in instruction, is calculated as (nx8). The new value of PC is (nx8). Value of n = 0,1,…,7

EX: **RST$_1$;** $SP \leftarrow SP - 1$
$[SP] \leftarrow PC_H$
$SP \leftarrow SP - 1$
$[SP] \leftarrow PC_L$
$PC \leftarrow 0008$ (1x8= 0008)

Addressing Mode	Flags Affected	Cycles	T-State
Indirect	None	3 (OP+MR+MR)	12

Stack I/O and Machine Control Group

- All stack operation which are PUSH and POP are compulsorily 16-bit operations. We can PUSH register pairs and not individual 8-bit register.
- PUSH and POP only support Register Addressing Mode. Hence if we want to push a number like 5000H we must first move into a register pair and then push it like
  EX: LXI B, 5000H
  PUSH B

1) PUSH R_P

It pushes the given register pair into the stack

EX: **PUSH B;** SP ← SP -1
[SP]← B
SP ← SP -1
[SP]← C

Addressing Mode	Flags Affected	Cycles	T-State
Register	None	3 (OP+MR+MR)	12

2) POP R_P

It pops the top 2 elements from the stack into the given register pair. The lower byte comes out first as the higher byte was pushed in first and stack operates in LIFO manner.

EX: **POP B;** C ← [SP]
SP ← SP +1
B ← [SP]
SP ← SP +1

Addressing Mode	Flags Affected	Cycles	T-State
Register	None	3 (OP+MW+MW)	10

- ***NOTE:*** *PSW can only be used in PUSH and POP instruction.*

3) PUSH PSW

It pushes the PSW (Program Status Word) into the stack.
The PSW is the combination of the accumulator and Flag register, accumulator being higher byte.

EX: **PUSH PSW;** SP ← SP -1
[SP]← A
SP ← SP -1
[SP]← F

Addressing Mode	Flags Affected	Cycles	T-State
Register	None	3 (OP+MR+MR)	12

4) POP PSW

It pops the top 2 elements from the stack into the PSW.

EX: **POP PSW;** F ← [SP]
SP ← SP +1
A ← [SP]
SP ← SP +1

Addressing Mode	Flags Affected	Cycles	T-State
Register	None	3 (OP+MW+MW)	10

I/O Instructions:

1) IN 8-bit I/O Port Address

This instruction is used to read data from I/O port, whose address is given in instruction. This data can be read into Accumulator only.

EX: **IN 80;** A ← $[80]_{I/O}$

Addressing Mode	Flags Affected	Cycles	T-State
Direct	None	3 (IO+MR+IOR)	10

2) OUT 8-bit I/O Port Address

This instruction is used to send data from accumulator to I/O port, whose address is given in instruction. This data can be sent into Accumulator only.

EX: **OUT 80;** $[80]_{I/O}$ ← A

Addressing Mode	Flags Affected	Cycles	T-State
Direct	None	3 (IO+MR+IOW)	10

Machine Control Instructions:

1) SIM (Set Interrupt Mask)

This instruction is used to set the interrupt masking pattern for the µP.
The appropriate bit pattern is loaded into accumulator a then this instruction is executed.
It is basically used to mask and unmask the interrupt except TRAP and INTR.
It is also used to send bit out through the serial out pin SID.

EX: **SIM;** A ← µP accepts the masking pattern through the accumulator.

Addressing Mode	Flags Affected	Cycles	T-State
Implied	None	1	4

2) RIM (Reset Interrupt Mask)

This instruction is used to read the interrupt masking pattern for the µP.
After executing this instruction, the µP loads the bit pattern into accumulator.
It is also used to receive a bit out through serial in pin SID.

EX: **RIM;** A ← µP loads the masking pattern into the accumulator.

Addressing Mode	Flags Affected	Cycles	T-State
Implied	None	1	4

3) EI (Enable Interrupt)

This instruction is used to enable the interrupts in the µP.
This instruction effects all the interrupts except TRAP.

EX: **EI;** $INTE_{F/F}$ ← 1

Addressing Mode	Flags Affected	Cycles	T-State
Implied	None	1	4

4) DI (Disable Interrupt)

This instruction is used to disable the interrupts in the µP.
This instruction effects all the interrupts except TRAP.

EX: **EI;** $INTE_{F/F}$ ← 0

Addressing Mode	Flags Affected	Cycles	T-State
Implied	None	1	4

5) NOP (No Operation)

This instruction performs no operation, but consumes times of the µP.
It is simplest method of causing a software delay.

EX: **NOP;** Nothing

Addressing Mode	Flags Affected	Cycles	T-State
Implied	None	1	4

6) HLT (Halt)

This instruction signifies the end of the program.
It causes µP to stop fetching any further instruction, hence program execution is stopped.

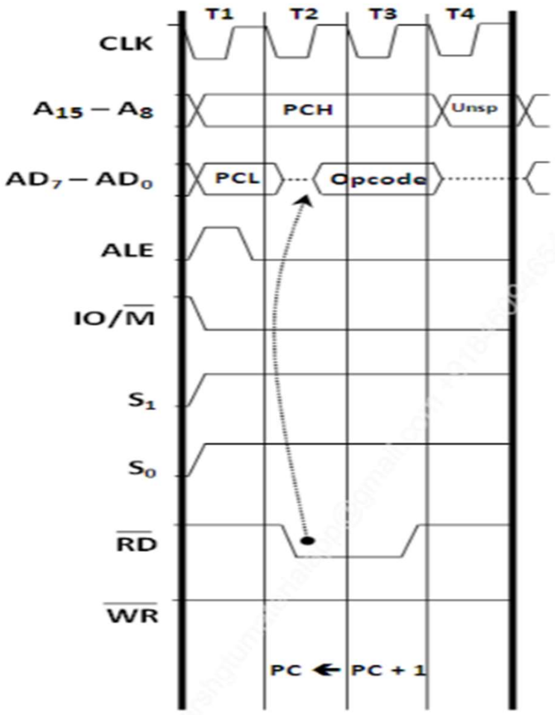
EX: **HLT;** Halt Flip-Flop ← 1

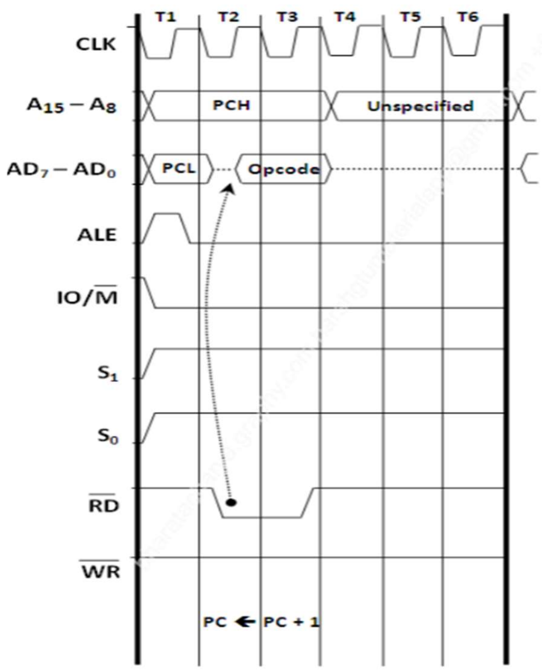
Addressing Mode	Flags Affected	Cycles	T-State
Implied	None	1+1T	5

Machine Cycles:

Name	IO/M	RD	WR	S1	S0	INTA	T-State
Opcode Fetch	0	0	1	1	1	1	4 OR 6
Memory Read	0	0	1	1	0	1	3
Memory Write	0	1	0	0	1	1	3
IO Read	1	0	1	1	0	1	3
IO Write	1	1	0	0	1	1	3
INT Acknowledge	1	1	1	1	1	0	3 OR 6
Bus Idle	0	1	1	0	0	1	3

Timing Diagram:





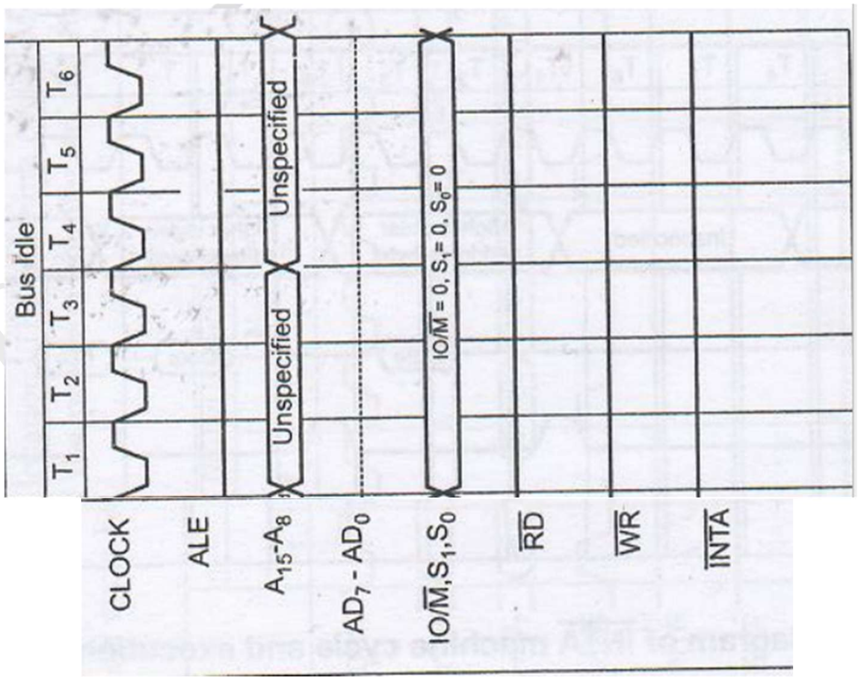
Instructions that use a 6T Opcode Fetch

PCHL
SPHL
INX
DCX
PUSH
CALL (all types)
RC (Conditional RET)
RSTn

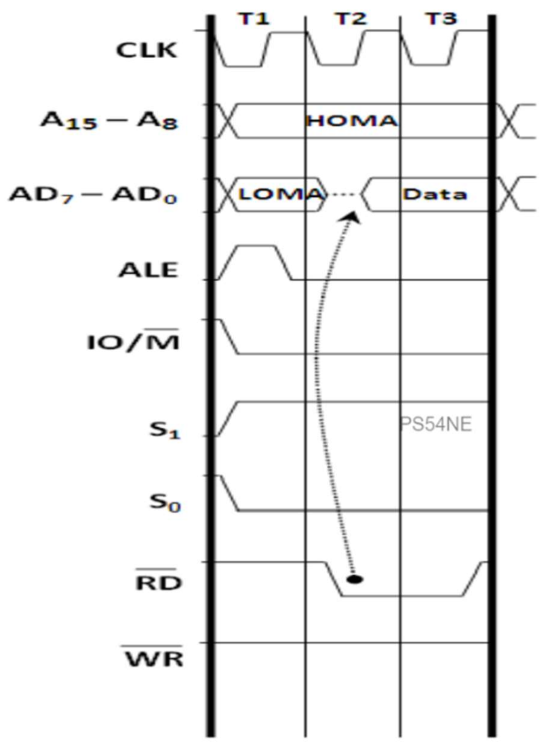
You will of course, understand this later as you learn timing diagrams of instructions.

OPCODE Fetch:
(4 – T State)

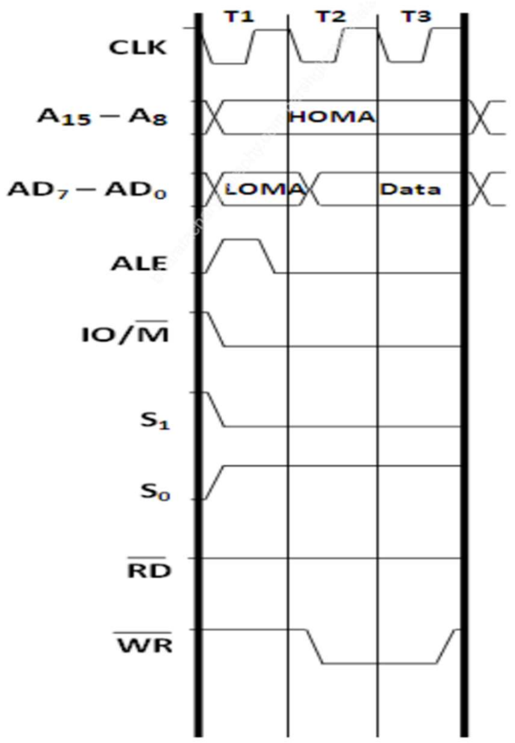
OPCODE Fetch:
(6 – T State)



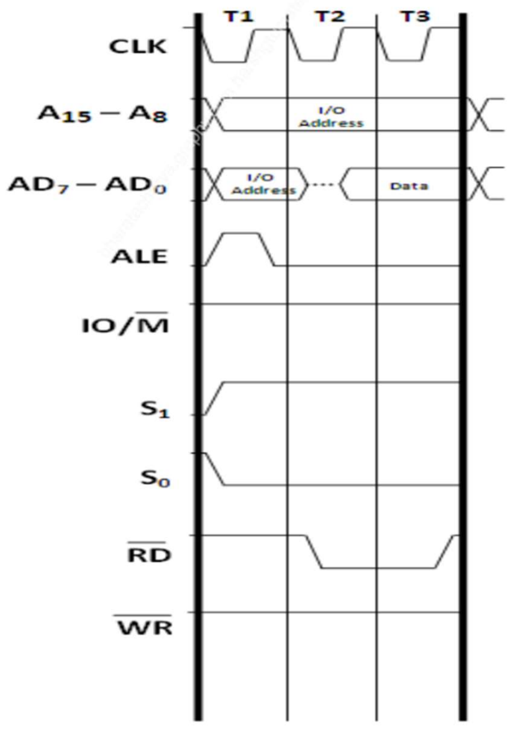
BUS Idle:



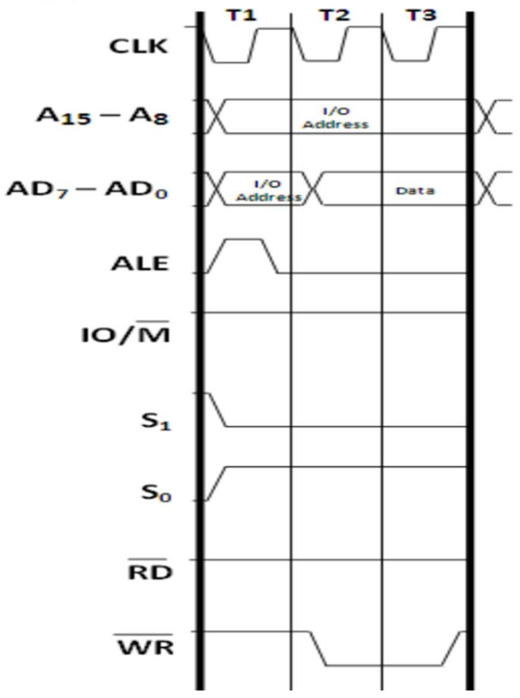
Memory Read:



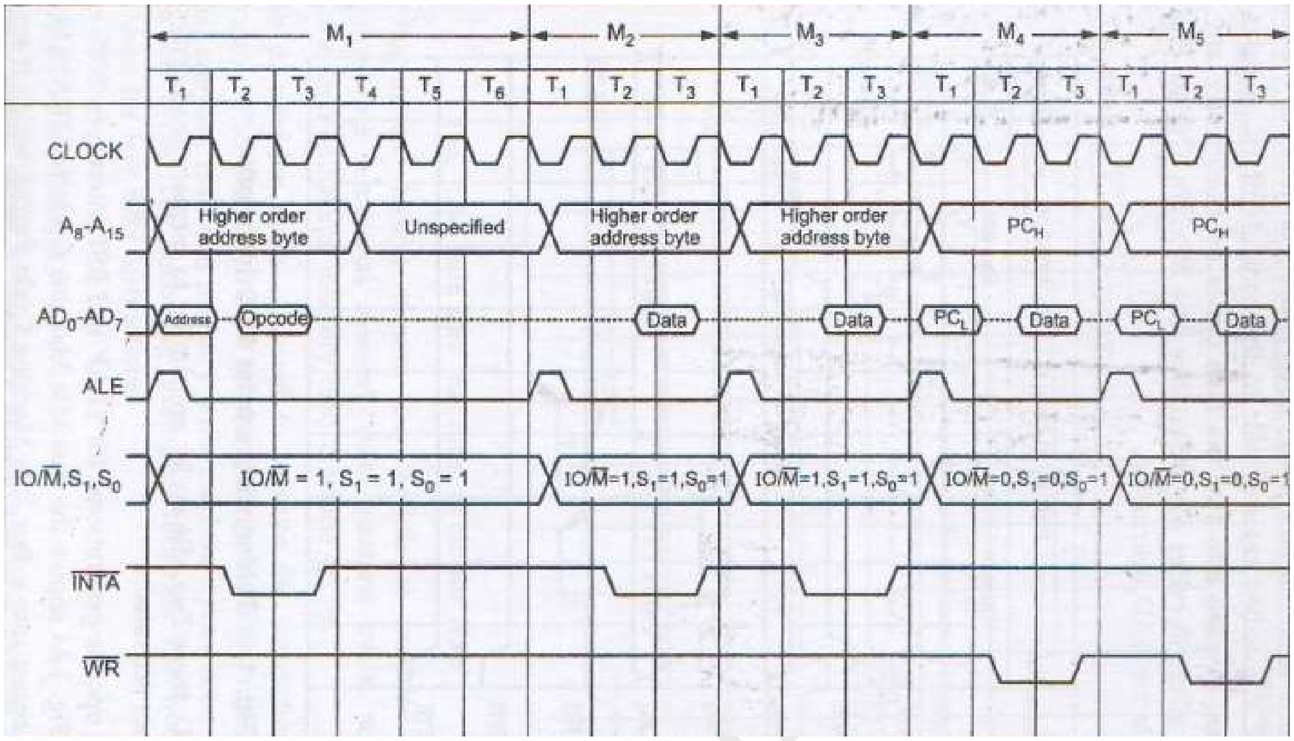
Memory Write



I/O Read:

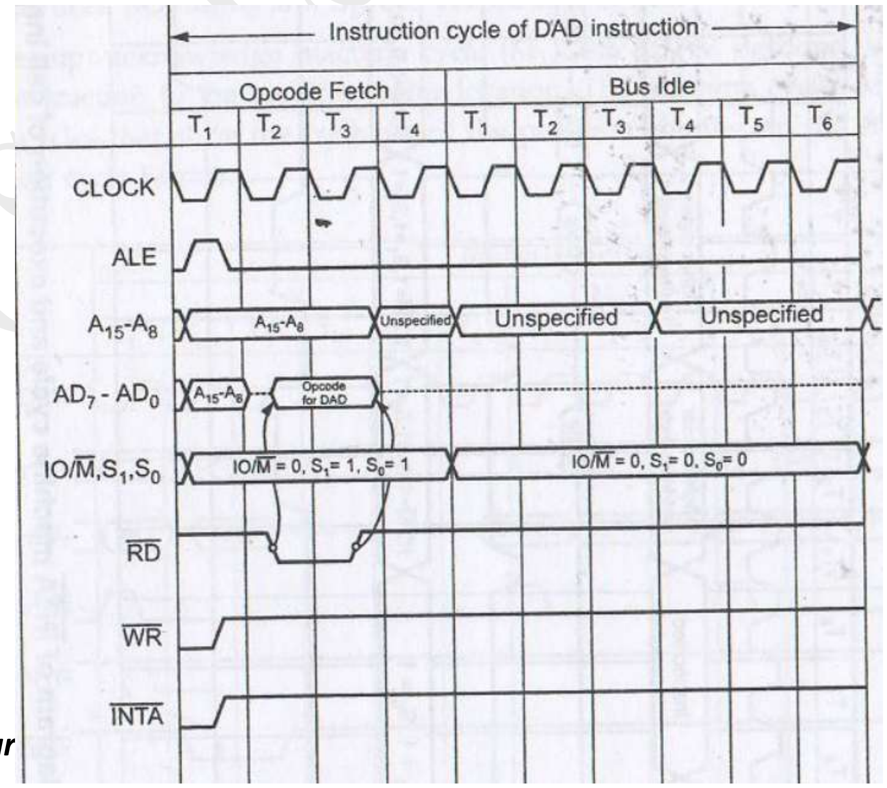


I/O Write:



CALL Timing Diagram:

DAD Timing Diagram:



Programs

(1) Add 2 8-bit numbers stored at 2000H and 2001 H. Store Sum at 2002H and carry at 2003H.

Without using M

MVI C, OOH	Assume Carry = 0
LDA 2000H	A = 1 number
MOV B, A	B = 1" number
LDA 2001H	A = 2nd number
ADD B	A = Sum
JNC SKIP	Check for carry
INR C	Increment C register if Carry Flag = 1
SKIP: STA 2002H	Store Sum at 2002
MOV A, C	A = Carry from C Register
STA 2003H	Store Carry at 2003H
HLT	End of program

Using M

MVI C, OOH ;	Assume Carry = 0
LXI H, 2000H	HL = 2000H; M = 1 number
MOV A, M	A= 1' number
INX H	HL = 2001H; M = 2' number
ADD M	A = sum
JNC SKIP	Check for carry
INR C	Increment C register if Carry Flag = 1
SKIP: INX H	HL= 2002H
MOV M, A	Store Sum at 2002H
INX H	HL = 2003H
MOV M, C	Store carry at 2003H
HLT	End of Program

(2) Transfer a block of 10 bytes from 2000H to 3000H.

LXI B, 2000H	Source pointer at 2000H
LXI D, 3000H	Destination pointer at 3000H
MVI L, 0AH	Count of 10d = OAH
<u>BACK</u>: LDAX B	A gets Source data
STAX D	A stored at Destination
INX B	Increment Source pointer
INX D	Increment Destination pointer
DCR L	Decrement Count register
JNZ <u>BACK</u>	Loop if count is not zero
HLT	End of program

(3) Perform inverted block transfer of 10 bytes from 2000H to 3000H.

LXI B, 2000H	Source pointer at 2000H
LXI D, 3009H	Destination pointer at 3009H
MVI L, 0AH	Count of 10d = OAH
<u>BACK</u>: LDAX B	A gets Source data
STAX D	A stored at Destination
INX B	Increment Source pointer
DCX D	Decrement Destination pointer
DCR L	Decrement Count register
JNZ <u>BACK</u>	Loop if count is not zero
HLT	End of program

(4) Exchange two blocks of 10 bytes from 2000H to 3000H.

LXI B, 2000H	Pointer to 1st Block at 2000H
LXI D, 3000H	Pointer to 2nd Block at 3000H
MVI L, 0AH	Count of 10d = OAH
BACK: LDAX B	A = data from Block 1
MOV H, A	H gets data from A
LDAX D	A = data from Block 2
STAX B	1 block gets q
MOV A, H	A gets data from H
STAX D	2 block gets P
INX B	Increment Source Pointer
INX D	Increment Destination Pointer
DCR L	Decrement Count Register
JNZ BACK	Loop if count is not zero
HLT	End of program

(5) Add a series of ten 8-bit number stored from 2000H store result at 200AH and 200BH.

LXI H, 2000H	HL = 2000H: M = 1 number
MVI C, OAH	Count
MVI A, OOH	Sum
MVI B, OOH	Carry
BACK: ADD M	Add current number
JNC SKIP	Check for carry
INR B	Increment B register if Carry Flag = 1
SKIP: INX H	Point to next number
DCR C	Decrement count
JNZ BACK	Loop
MOV M, A	Store Sum
INX H	Point to next location
MOV M, B	Store Carry
HLT	End of program

(6) Find the highest in a series of ten 8-bit numbers are stored from 2000H. Store result at 200AH.

LXI H, 2000H	HL = 2000H: M = 1 number
MVI C, OAH	Count
MVI A, OOH	Highest number
BACK: CMP M	Compare current number
JNC SKIP	Check for carry
MOV A, M	If carry is 1 then A ← M
SKIP: INX H	Point to next number
DCR C	Decrement count
JNZ BACK	Loop
MOV M, A	Store Result
HLT	End of program

(7) Find the number of even and odd numbers in a series of ten 8-bit number stored from 2000H. Store result at 200Ah and 200BH.

LXI H, 2000H	HL = 2000H; M = 1' number
MVI C, OAH	Count
MVI D, OOH	Even count
MVI B, OOH	Odd count
BACK: MOV A, M	Get number into A
RRC	Check LSB
JC ODD	If carry, its Odd
INR D	Increment Even count
JMP SKIP	Move ahead
ODD: INR B	Increment Odd count
SKIP: INX H	Point to next number
DCR C	Decrement count
JNZ BACK	Loop
MOV M, D	Store Even count
INX H	Move to next location
MOV M, B	Store Odd count
HLT	End of program

(8) Find the number of Ones in an 8-bit number stored at 2000H store result at 2001H.

LDA 2000H	A = given number
MVI B, OOH	Number of 1's
MVI C, 08H	Loop count
BACK: RRC	Rotate right
JNC SKIP	If no carry, move ahead
INR B	Increment ls count
SKIP: DCR C	Decrement loop count
JNZ BACK	Loop
MOV A, B	A + 1 s count from B
STA 2001 H	Store result
HLT	End of Program

(9) Write a program to multiply 2 8-bit numbers stored at 2000H and 2001H. Store the result at 2002H and 2003H.

LXI H, 0000H	Result will be in HL, initialized to 0
LXI D, 0000H	DE initialized to 0
LDA 2000H	Take 1 Operand
ADI 00H	For Zero Check
JZ Store	For Zero Check
MOV E, A	DE = 1 operand
LDA 2001H	Take 2nd operand
ADI 00H	For zero Check
JZ Store	For zero check
MOV C, A	C = 2 operand
Back: DAD D	Add the 1st operand to HL, HL=HL+DE
DCR C	Decrement the 2 operand
JNZ Back	Loop till C becomes 0
Store: SHLD 200H	Store the result
HLT	End of the program

(9) Write a program to divide 2 8-bit numbers stored at 2000H and 2001H. Store the result at 2002H (Q) and 2003H (R).

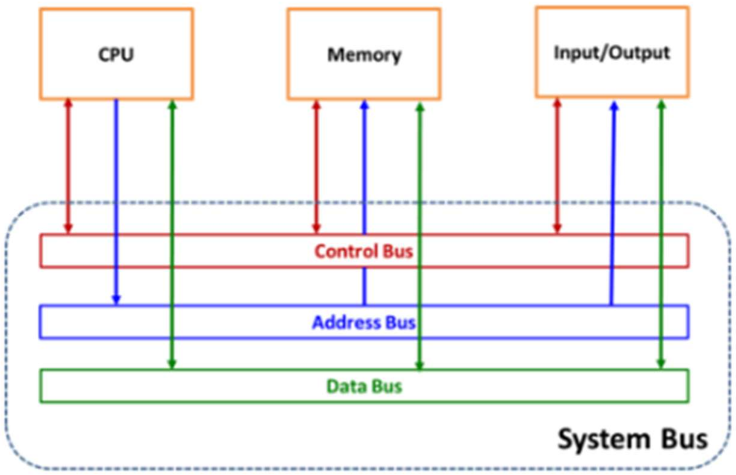
LDA 2001H	Take divisor in A
ADI 00H	For zero check
JZ Exit	If zero, simply exit the program
MOC C, A	C = divisor
MVI E, 00H	E = 0 (This will be quotient)
LDA 2000H	A = Dividend
BACK: CMP C	Compare A and C by doing A-C
JC Next	If A<C then move out of loop
SUB C	A ← A-C
INR E	Increment quotient
JMP BACK	Loop Back
Next: STA 2003H	Store reminder from A to 2003H
MOV A, E	Move quotient from E to A
STA 2002H	Store the quotient from A to 2002H
HLT	End of the program

WINTER: 2021

Q.1

(1) Explain System bus of 8085 Microprocessor.

- The 8085 microprocessor has a well-defined bus organization, which consists of three major components: Data, Address and Control bus.
- The bus organization is a critical aspect od the microprocessor as it is responsible for communication between the microprocessor and external memory and I/O devices.



- **<u>Data Bus:</u>**
  - It is a group od conducting wires which carries data only.
  - Data bus is bidirectional because data flow in both directions, from microprocessor to memory or I/O devices and from memory or I/O devices to microprocessor.
  - Length of this is 8-bit, ranging from 00H to FFH.

- When it is write operation, the processor will put the data on the bus, when it is read operation, the memory controller will get the data from specific memory block and put it into data bus.

➢ **Address Bus:**

- It is group of conducting wires which carries address only.
- Address bus is unidirectional because data flow in one direction, from microprocessor to memory or from microprocessor to I/O devices.
- Length of address bus is 16-bit. Ranging from 0000H to FFFFH.
- It can transfer maximum 16-bit address which means it can address 65, 536 different memory location.

➢ **Control Signal:**

- It is a group of conducting wires, which is used to generate timing and control all the associate peripherals, microprocessor uses control bus to process data.
- The control bus is a group of signals that are used to control the flow of data between the microprocessor and external memory and I/O devices.
- The control bus signals include read and write signals, chip select signals, and status signals.
- The read and write signals are used to control the direction of data transfer on the data bus.
- The chip select signals are used to select the external device that the microprocessor is communicating with.
- The status signals are used to indicate the status of the microprocessor or external device.

(2) Describe the functions of (1) ALE (2) TRAP (3) READY (4) HLDA

- **<u>READY PIN:</u>** The READY pin is an input pin used to synchronize the operation of the microprocessor with other slower devices in the system. When the READY pin is high, it indicates that the slower device is ready to receive or transmit data. The microprocessor will wait until the READY pin goes high before proceeding with its operations.

- **<u>ALE:</u>** ALE stands for Address Latch Enable and it is a control signal used to latch the address into external latch. The microprocessor uses this signal to indicate that the address on the address bus is stable and can be latched into an external latch for further use

- **<u>HLDA:</u>** HLDA stands for Hold Acknowledge, and it is an output signal used to acknowledge that the microprocessor has received the HOLD signal. When the microprocessor receives a HOLD signal, it responds with a HLDA signal to indicate that it has relinquished control of the system bus.

- **<u>TRAP:</u>** TRAP is a non-maskable interrupt, which means that it cannot be disabled. It has the highest priority among all the interrupts and is used for critical events that require immediate attention.
  The TRAP interrupt is used for handling critical situations such as power failure, system reset, or other hardware malfunctions. It is also used for implementing a software breakpoint to pause the program execution for debugging purposes.

(3) Explain 8085 Programming Model and Flag Register.

➢ **<u>8085 Programming Model:</u>**

- The 8085 microprocessors programming model consists of six registers, namely the accumulator (A), the program counter (PC), the stack pointer (SP), the flag register (FLAGS), the temporary register (register pair H and L), and the memory address register (MAR).
- The accumulator (A) is an 8-bit register that stores data and arithmetic operations are performed on this register.
- The program counter (PC) is a 16-bit register that holds the address of the next instruction to be executed.
- The stack pointer (SP) is also a 16-bit register those points to the top of the stack in memory.
- The flag register (FLAGS) is an 8-bit register that stores the status of the arithmetic and logic operations.
- The temporary register (register pair H and L) is a 16-bit register used for holding intermediate values during arithmetic and logical operations.
- The memory address register (MAR) is a 16-bit register that holds the address of the memory location to be accessed.

➢ Flag Register:

- The flag register in the 8085 microprocessor is a special register that holds the status of various operations performed by the microprocessor.
- It is a vital component of the 8085 microprocessor, as it provides information about the result of arithmetic and logical operations, and is used to control the flow of program execution.
- The flag register is 8-bit register, with each bit representing a specific flag.
- Flag register includes five flip-flops, which are set or reset after an operation according to the data conditions of the result in the accumulator and other registers.
- The microprocessor uses these flags to set and test data conditions.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	**Z**		**AC**		**P**		**CY**

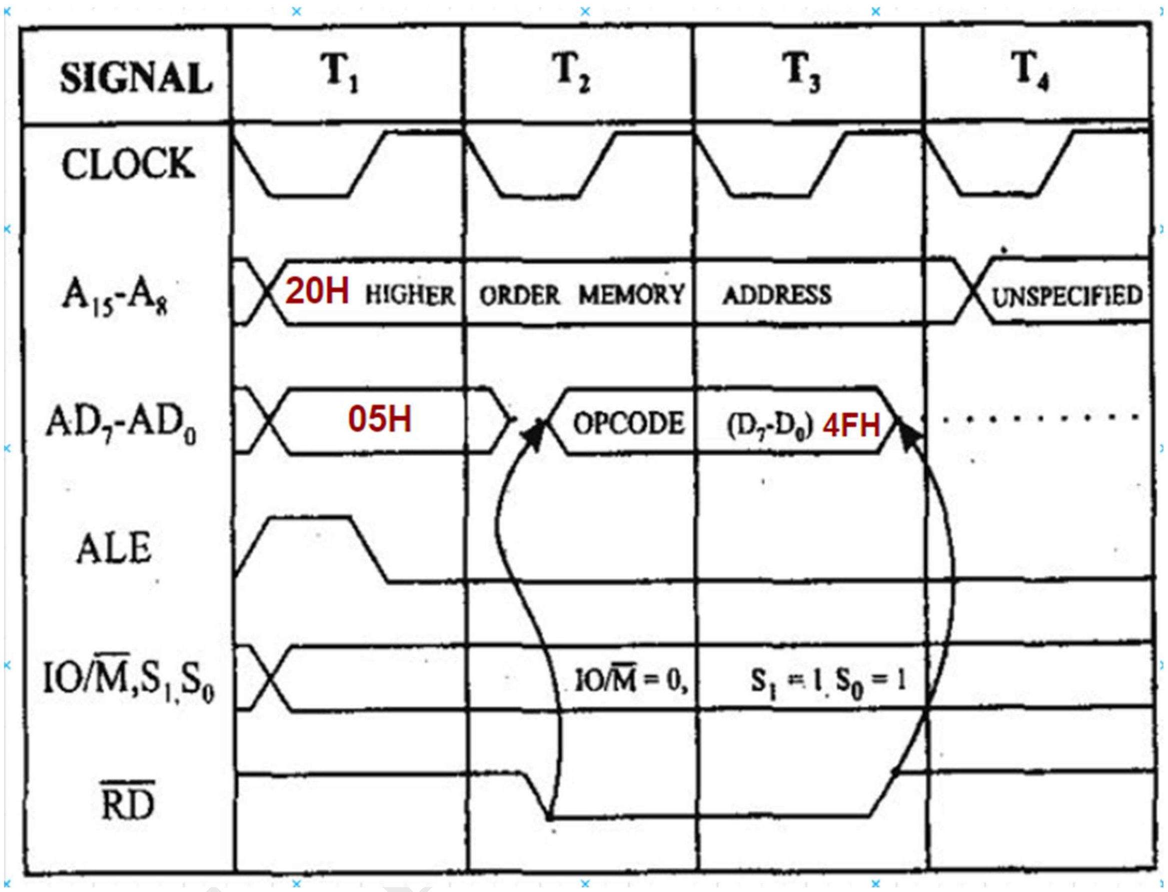
- **S (Sign) Flag:** Indicates the sign of the result of an arithmetic operation. If the result is negative, the flag is set (1), otherwise it is reset (0).
- **Z (Zero) Flag:** Indicates result of mathematical or logical operation is zero or not. If result is zero this flag will be set (1), otherwise reset (0).
- **AC (Auxiliary Carry) Flag:** Indicates if there is carry out from lower nubble to the upper nibble during arithmetic operation. This flag is used in binary-coded decimal (BCD) arithmetic operations.
    - If carry generated, (0) => If there is no carry generated.
- **P (Parity) Flag:** Parity is the number of 1's in a number. If the is even then that number is known as even parity number.
- If the number is odd then that number is known as an odd parity number. This flag indicates whether the current result is of even parity (set) (1) or of odd parity (reset) (0).
- **CY (Carry) Flag:** Indicates if there is carry out from most significant bit during an arithmetic operation. (1)=> last operation generate carry, (0)=> no carry from last operation

Q.2

(1) Differentiate 8085 Microprocessor with 8086 Microprocessor.

8085 Microprocessor	8086 Microprocessor
This is an accumulator based on 8-bit microprocessor which process 8-bit of data at a time.	This is a register based on 16-bit microprocessor which process 16-bit of data at a time.
It has a data bus of 8-bit size.	It has a data bus of 16-bit size.
Size of address bus in this is 16-bit.	Size of address bus in this is 20-bit.
The maximum accessible memory capacity of this processor is 64 KB.	The maximum accessible memory capacity of this processor is 1 MB.
It has an 8-bit ALU.	It has a 16-bit ALU.
It has an on-chip oscillator of 3 MHZ.	It is available in three version with a clock speed of 5,8 and 10 MHz.
It consists of 6500 transistors in its structure.	It consists of 29000 transistors in its structure.
Supports only single mode of operation.	Support two modes of operation: Minimum and Maximum Mode.
Pipelining architecture is not supported in 8085.	Pipelining architecture is supported in 8086.
It is cheaper than 8086.	It is relatively expensive than 8085.

(2) Draw a timing diagram of MOV M, D instruction of 8085 microprocessor.



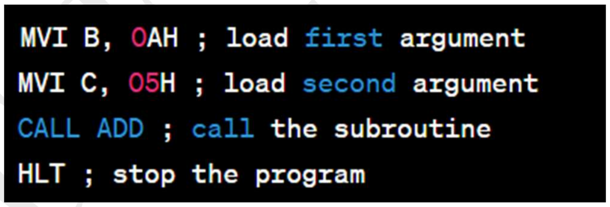
(3) Write an 8085-assembly language program to arrange the following numbers in ascending order: 29H, 47H, 06H, 03H, 17H. **<u>OR</u>**

Write an 8085-assembly language program to arrange the following numbers in descending order: 29H, 47H, 06H, 03H, 17H.

Q.3

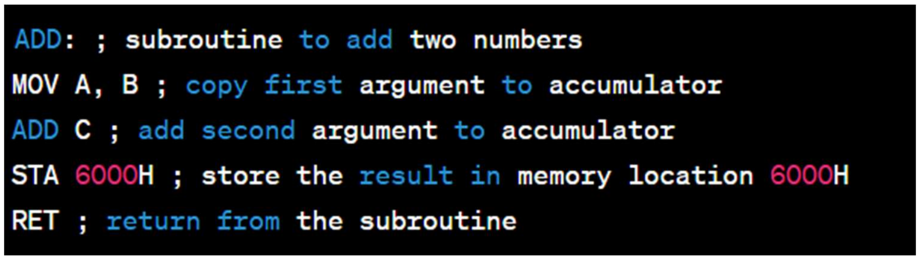
(1) Explain Subroutine with suitable example.

- Subroutine is a sequence of instructions that can be called multiple times from different parts of a program.
- The subroutine contains a set of instructions that perform a specific task, and when called from the main program, it returns the control to the main program after completing the task.
- Subroutines help in modularizing the program by breaking it into smaller modules, making it easier to read and maintain.
- 8085 microprocessor provides several instructions to implement subroutines. The most commonly used instructions are CALL, RET.
- The CALL instruction is used to call a subroutine from the main program. The CALL instruction stores the current address of the program counter on the stack and loads the address of the subroutine into the program counter.
  The syntax of the CALL instruction is: CALL address
  Where the address is the address of the first instruction of the subroutine.
- The RET instruction is used to return the control from the subroutine to the main program. The RET instruction pops the address from the stack and loads it into the program counter.
  The syntax of the RET instruction is: RET



```
MVI B, 0AH ; load first argument
MVI C, 05H ; load second argument
CALL ADD ; call the subroutine
HLT ; stop the program
```

<u>Subroutine:</u>



```
ADD: ; subroutine to add two numbers
MOV A, B ; copy first argument to accumulator
ADD C ; add second argument to accumulator
STA 6000H ; store the result in memory location 6000H
RET ; return from the subroutine
```

(2) Explain following instructions with no. of bytes, machine cycles and Tstates required for execution:

1. SHLD 2. RAL

- **SHLD:** The HL pair is stored into the location pointed by given address and address + 1.

Addressing Mode	Flag affected	Cycles	T-State	Bytes
Direct	None	5	16	3

- **RAL:** The contents of accumulator is rotated left by 1. The MSB goes to the carry and carry goes to LSB.

Addressing Mode	Flag affected	Cycles	T-State	Bytes
Implied	Carry	1	4	1

(3) Ten 8-bit values are stored from memory location 5000H onwards. Write an 8085-assembly language program to add POSITIVE values on addresses starts from 5100H and NEGATIVE values on addresses starts from 5200H.

Q.3

(1) What is Interrupt? List hardware interrupts in 8085.

- An interrupt is a signal that is sent to the processor to temporarily halt its current operation and switch to an interrupt handling routine. The interrupt handling routine is a small program that is executed to handle the interrupt event, and it typically saves the current state of the processor, performs the necessary actions to handle the interrupt, and then restores the saved state of the processor and resumes its previous operation.
- There are five hardware interrupts:
  - **TRAP:** It is a non-maskable interrupt, and its priority is the highest among all interrupts. It is used for critical error handling.
  - **RST 7.5:** It is a maskable interrupt and has the second-highest priority. It is triggered by an external signal on the INTR pin or by a software instruction.
  - **RST 6.5:** It is a maskable interrupt and has the third-highest priority. It is triggered by an external signal on the INTR pin or by a software instruction.
  - **RST 5.5:** It is a maskable interrupt and has the fourth-highest priority. It is triggered by an external signal on the INTR pin or by a software instruction.
  - **INTR:** It is a maskable interrupt and has the lowest priority among all interrupts. It is triggered by an external signal on the INTR pin or by a software instruction.

(2) Explain following instructions with no. of bytes, machine cycles and T-states required for execution:

1. CALL 2. CPI

➢ **Call:** The call instruction transfers the program sequence to the memory address given in the operand. Before transferring, the address of the next instruction after CALL is pushed onto the stack.
Ex. CALL 2000H

Addressing Mode	Flag affected	Cycles	T-State	Bytes
Immediate	None	6 (Unconditional) 7/17 (Conditional)	18	3

➢ **CPI:** The CPI (Compare Immediate) instruction is used to compare the contents of the accumulator (ACC) register with an 8-bit immediate data value. The immediate data is specified as the second byte of the instruction.

Addressing Mode	Flag affected	Cycles	T-State	Bytes
Immediate	None	2	7	2

(3) Ten 8-bit values are stored from memory location 3000H onwards. Write an 8085-assembly language program to find the largest value and stored it on the location 4000H.

➢

```
       LXI H, 3000H; load HL register pair with starting memory location
       MVI C, 0AH; set loop counter to 10 for 10 values
       MOV A, M; move first value to accumulator
       INX H; increment memory pointer
BACK:  CMP M; compare current value with accumulator
       JNC SKIP; jump if not carry, i.e., if accumulator is larger
       MOV A, M; move current value to accumulator
SKIP:  INX H; increment memory pointer
       DCR C; decrement loop counter
       JNZ BACK; jump if not zero, i.e., if there are more values to compare
       STA 4000H; store the largest value in memory location 4000H
       HLT; halt the program
```

Q.4

(1) **What will be the value in accumulator, for the given 8085 program below?**

```
MVI C,7FH
MVI B, 3EH
MOV A, B
RLC
RLC
ANI 7FH
HLT
```

➢

MVI C,7FH	**C =**	0	1	1	1		1	1	1	1
MVI B, 3EH	**B =**	0	0	1	1		1	1	1	0
MOV A, B	**A =**	0	0	1	1		1	1	1	0
RLC	**A =**	0	1	1	1		1	1	0	0
RLC	**A =**	1	1	1	1		1	0	0	1
ANI 7FH	**A =**	0	1	1	1		1	0	0	1
HLT										

➢ **A (Accumulator) = (0111 1001) => 79H**

(2) Consider the following 8085 assembly language instructions: LXI D, 1234H 04 2 NEXT: DCX D MOV A, E ORA D JNZ NEXT What amount of delay is generated if the crystal frequency is 4 MHz?

(3) Explain various addressing mode in 8085 microprocessors.

Special Notes:
The “I” in the instruction indicates “Immediate Addressing Mode”
Hence the number in the instruction must be DATA.
Hereafter, when you see a number in any instruction look for an “I”.
If “I” is present then the number is DATA else its an address.
The “X” in the instruction (LXI) indicates “Register Pair”.

➢ Immediate Addressing Mode:

- In this mode, the **data** is specified in the instruction itself.

EX.

MVI A, 35H; Move immediately the value 35 into Accumulator.
; A ←35H

LXI B, 4000H; Move immediately the value 4000 into register pair BC.
; BC ←4000H

Advantage:

Programmer can easily identify the operands.

Disadvantages:

Always more than one byte hence requires more space.
The µP requires 2 or 3 machine cycles to fetch instruction hence slow.

➢ Register Addressing Mode:

- In this mode, the **data** is specified in registers.

EX.

MOV B, C; Move the contents of C-register into B-register.
; B ←C

INR B ; Increment the content of B-register.
; B ← B+1

Advantage:

The µP requires only one machine cycle to fetch instructions.

Disadvantages:

Operands can not be easily identified.

➢ Direct Addressing Mode:

- In this mode, the address of the operand is specified in the instruction itself.

EX.

LDA 2000H; Load the Accumulator with contents of location 2000.
; A ← [2000]

STA 2000H; Store the content of the accumulator at location 2000.
; [2000] ←A

Advantage:

Programmer can easily identify the address of operands.

Disadvantages:

These are 3 bytes instructions hence 3 fetch cycles require.

➢ In-Direct Addressing Mode:

- In this mode, the address of the operand is specified in the register.
- Hence, the instruction indirectly points to the operands.
- Even the memory pointer 'M' can be used as it is pointed by HL pair.

EX.

STAX B; Store the content of the accumulator at the location pointed by BC pair.
; [BC] ← A

INR M ; Increments the contents of the location pointed by HL pair.
; [HL] ← [HL]+1

Advantage:

Address of the operand is not fixed and hence it can be used in loop.
Size of the instruction is small as compared to direct addressing mode.

Disadvantages:

Requires initialization of the register pair hence requires at least one more instruction.

Special Notes:
Remember, during programming when you want to access only 1 or 2 locations, use Direct addressing mode as it is simpler.
But when you want to access a series of locations, use Indirect addressing mode.
Initialize the first address in a register pair.
Thereafter increment/decrement that pair in a loop to access a series of locations.

➢ Implied Addressing Mode:

- In this mode, the operand is implied in the instruction.

EX.

STC ; Set the carry flag in flag register.

; Cy 🡐 1

CMC; Complement the carry flag in flag register.

; [HL] 🡐 [HL]+1

Advantage:

Instructions are generally only one byte.

Disadvantages:

Programmer can not easily identify the value of the operand.

Q.4

(1) **What will be the value in accumulator, for the given 8085 program below?**

```
MVI A, 07H
RLC
MOV B, A
RLC
RLC
RLC
ORA B
HLT
```

MVI A, 07H	**A =**	0	0	0	0		0	1	1	1
RLC	**A =**	0	0	0	0		1	1	1	0
MOV B, A	**B =**	0	0	0	0		1	1	1	0
RLC	**A =**	0	0	0	1		1	1	0	0
RLC	**A =**	0	0	1	1		1	0	0	0
RLC	**A =**	0	1	1	1		0	0	0	1
ORA B	**A =**	0	1	1	1		1	1	1	1
HLT										

➢ **A = (0111 1111) => 7FH**

(2) Write an 8085 assembly language program to convert a two-digit BCD number into its equivalent hexadecimal number

(3) Define the followings: Addressing mode, Machine Cycle, T-state, JC, CMP, RET, SBB, STC

➢ **JC:**

Addressing Mode	Flag affected	Cycles	T-State
Immediate	None	2	7

➢ **CMP:**

Addressing Mode	Flag affected	Cycles	T-State
Register	ALL	1	4

➢ **RET:**

Addressing Mode	Flag affected	Cycles	T-State
Indirect	None	3	10

➢ **SBB:**

Addressing Mode	Flag affected	Cycles	T-State
Register	All	1	4

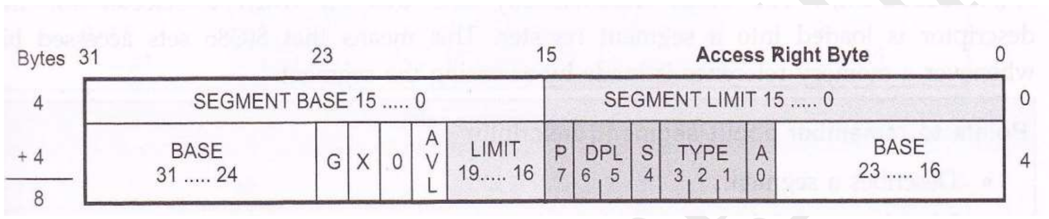
➢ **STC:**

Addressing Mode	Flag affected	Cycles	T-State
Implies	Carry	1	4

Q.5

(1) Explain format of descriptor in 80386 with diagram.

- In protected mode, Memory Management Unit (MU) uses the segment selector to access a descriptor for the desires segment in a table of descriptors in memory.
- Segment descriptor is a special structure which describes the segment.
- Exactly one segment descriptor must be defined for each segment of the memory.



- **<u>Base:</u>** It contains the 32-bit base address for a segment.
- **<u>Limit:</u>** It defines the size of the segment.
- **<u>Granularity Bit:</u>** It specifies the units with which the limit field is interpreted. When bit = 0 then limit is interpreted in unit of one byte, when bit = 1 limit is interpreted in units of 4 Kbytes.
- **<u>0 (Reserved By Intel):</u>** It neither can be defined nor can be used by user.
- **<u>AVL/U (User bit):</u>** This bit is completely undefined, and 80386 ignores it. This is available bit for user or operating system.
- **<u>P (Present Bit):</u>** If P = 1 the segment is loaded in physical memory.
- **<u>DPL (Descriptor Privilege Level):</u>** It is a 2-bit field defines the level of privilege associated with the memory space that the descriptor defines – DPL_0 is the most privileged whereases DPL_3 is the least privileged.
- **<u>S (System Bit):</u>** If the S = 1 then the segment is either a code or data, When S = 0 then the segment is system segment.
- **<u>A (Accessed Bit):</u>** 80389 sets accessed bit whenever a memory reference is made by accessing the segment.

(2) Draw internal block diagram of 80286.

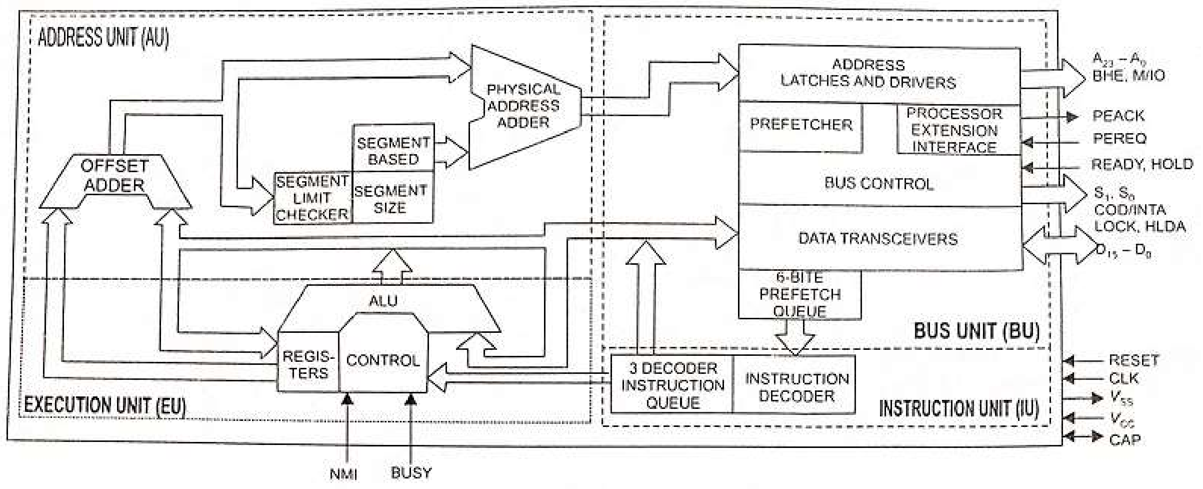


Fig. 11.15 *Internal block diagram of the 80286 processor*

➢ The Architecture of 80286 Microprocessor is an advanced, high-performance microprocessor with specially optimized capabilities for multi-user and multitasking systems. The 80286 has built-in memory protection that supports operating system and task isolation as well as program and data privacy

➢ **Address Unit (AU):**

- The address unit (AU) is used to determine the physical addresses of instructions and operands which are stored in memory.
- The AU computes the 20-bit physical address based on the contents of the segment register and 16-bit offset just like 8086.
- The address lines derived by AU can be used to address different peripheral devices such as memory and I/O devices.
- This physical address computed by the address unit is sent to the Bus Unit (BU) of the CPU.

- **Bus Unit (BU):**

  - The bus unit interfaces the 80286 with memory and I/O devices. This processor has a 16-bit data bus, a 24-bit address bus, and a control bus.
  - The bus unit is responsible for performing all external bus operations.
  - This unit consists of latches and drivers for the address bus, which transmit the physical address A19-A0. The A19-A0 facilitates all memory and I/O devices for read and write operations.
  - The bus unit is used to fetch instruction bytes from the memory. Generally, the instructions are fetched in advance and stored in a queue for faster execution of the instructions. This concept is known as instruction pipelining.

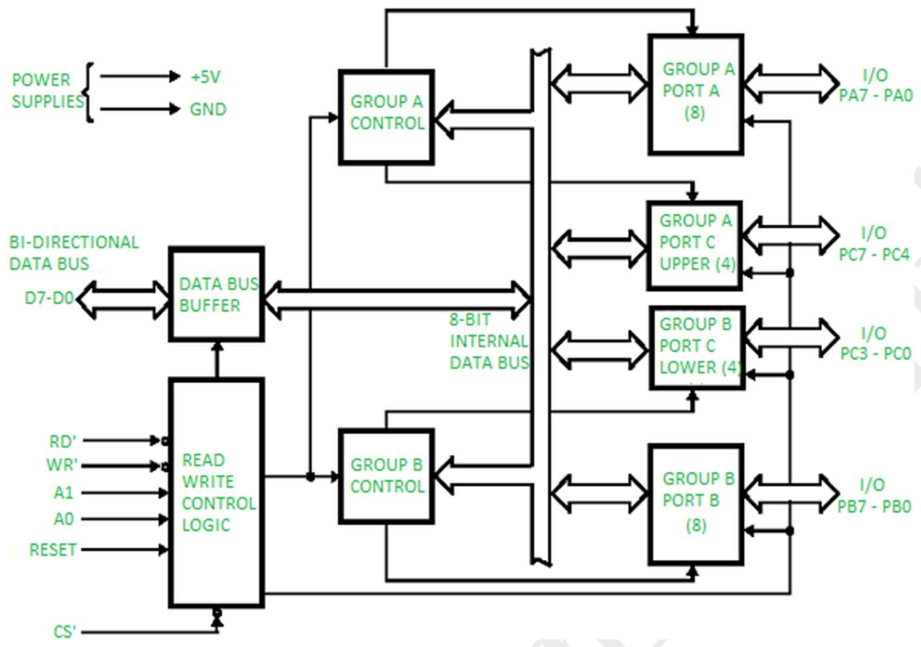
- **Instruction Unit:**

  - It includes an instruction decoder and a three decoded instruction queue. The instruction unit decodes up to three prefetched instructions and holds them in the queue. With this instruction unit there is decode and execution overlap, which speeds up the operation of a processor.

- **Execution Unit:**

  - The execution unit includes ALU, register same as 8085 and the CPU. The register consists general purpose register, indexed register, pointer registers, flag register and 16-bit machine status word (MSW) register.
  - The output of the decoded instruction queue is fed to a control circuit of the execution unit.
  - This unit is responsible for executing the instructions received from the decoded instruction queue.
  - The execution unit consists of the register bank, arithmetic and logic unit (ALU) and control block.

(3) Draw and explain the block diagram of the programmable peripheral interface 8255A.



➢ **Data Bus Buffer:**

- The tri-state bi-directional buffer is used to interface the internal data bus of 8255 to the system data bus.
- Input or Output instructions executed by the CPI either Read data from, or write data into the buffer.
- Output data from the CPU to the ports or control register, and input data to the CPU from the ports or status register are all passed through the buffer.

➢ **Control Logic:**

- The control logic block accepts control bus signals as well as inputs from the address bus, and issues commands to the individual group control block (Group A control and Group B control).

- It issues appropriate enabling signals to access the requires data/control words or status words. The input pins for the control logic section are described here.

➢ **Group A and Group B Controls:**

- Each of Group A and B control blocks receives control words from the CPU and issues appropriate commands to the ports associated with it.
- Group A control block controls Port A and PC_7-PC_4 while Group B control block control port B and PC_3-PC_0.
- **Port A:** This has an 8-bit latched and buffered output and an 8-bit input latch. It can be programmed in three modes: mode0, mode1, mode2.
- **Port B:** This has an 8-bit data I/O /latch/buffer and 8-bit data input buffer. It can be programmed in mode0 and mode 1.
- **Port C:** This has one 8-bit unlatched input buffer and an 8-bit output latch/buffer. Port C can be separated into two parts and each can be used as control signals for Port A and Port B in the handshake mode. It can be programmed for bit set/reset operation.

Q.5

(1) List and explain the segment registers of 8086 microprocessor.

- Segmentation allows for the efficient use of memory, as the data and code segments can be located in different areas of memory, allowing for more efficient use of available memory. The segment registers are an essential part of the memory addressing scheme of the 8086 microprocessors.
- **Code Segment (CS):** The Code Segment register holds the starting address of the code segment, where the program code is stored. The CS register is used in conjunction with the Instruction Pointer (IP) register to form the complete memory address of the instruction being executed.
- **Data Segment (DS):** The Data Segment register holds the starting address of the data segment, where the program's static and dynamic data is stored. It is used to access program data, such as arrays, structures, and variables.
- **Stack Segment (SS):** The Stack Segment register holds the starting address of the stack segment, where the program's stack is located. The stack is used for storing temporary data and return addresses during subroutine calls.
- **Extra Segment (ES):** The Extra Segment register is an additional data segment register that is used for storing extra data. It is often used in conjunction with string and block transfer instructions.

(2) Draw block diagram of 80386 microprocessor.

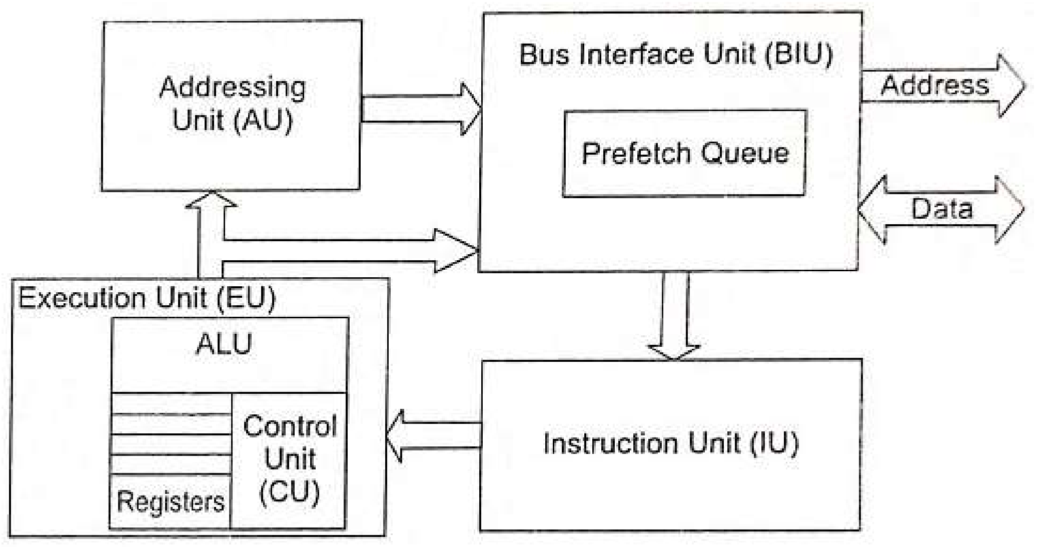


Fig. 11.31 *The simplified block diagram of 80386 processor*

- The 8086 is a 16-bit microprocessor that was introduced by Intel in 1978. It has a 20-bit address bus, which can access up to 1MB of memory. The 8086 microprocessor consists of four main units: the Addressing Unit (AU), the Bus Interface Unit (BIU), the Instruction Unit (IU), and the Execution Unit (EU).
- **<u>Addressing Unit (AU):</u>** The Addressing Unit (AU) is responsible for calculating the physical address of the memory location that is being accessed. It receives the segment register values and offset value from the BIU and then computes the physical address by multiplying the segment register value by 16 and adding it to the offset value. The physical address generated by the AU is sent to the BIU.

- **<u>Bus Interface Unit (BIU):</u>** The Bus Interface Unit (BIU) is responsible for interfacing the 8086 microprocessors with external devices. It is also responsible for fetching instructions and data from memory or I/O devices.

The BIU is further divided into two subunits: the Instruction Prefetch Queue (IPQ) and the Instruction Pointer (IP).

- **Instruction Unit (IU):** The Instruction Unit (IU) is responsible for decoding the instructions fetched from memory by the BIU. The IU consists of an instruction decoder and a queue for storing the decoded instructions. The instruction decoder decodes the instructions fetched from the IPQ and stores them in the queue.

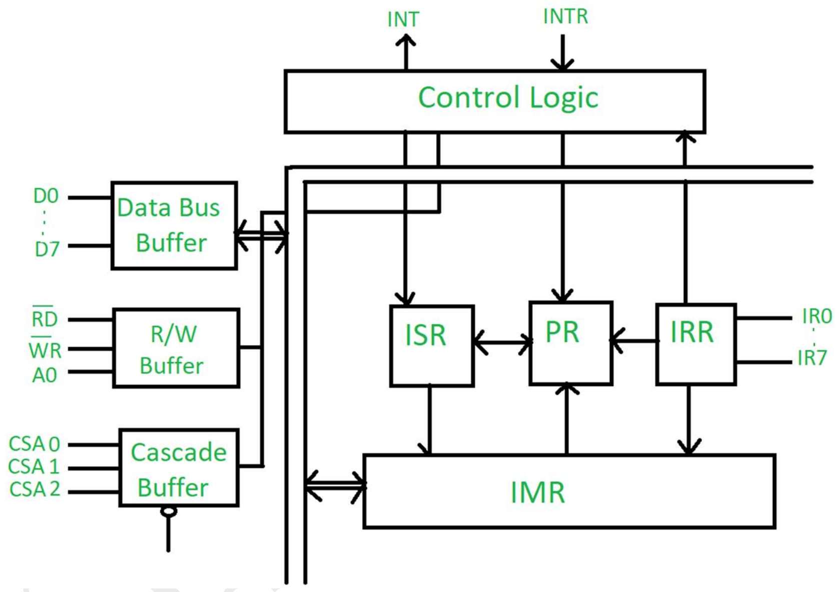
- **Execution Unit (EU):** The Execution Unit (EU) is responsible for executing the instructions decoded by the IU. It consists of a register set, an arithmetic and logic unit (ALU), and a control unit. The register set stores data and intermediate results during the execution of an instruction. The ALU performs arithmetic and logical operations on data stored in the registers. The control unit coordinates the execution of instructions by controlling the flow of data between the registers and the ALU.

80286	80386
It is 16-bit microprocessor.	It is a 32-bit microprocessor.
It is also known as i APX 286 and often called intel 286.	It also known as i386 or just 386.
It was the first 8086 based CPU with separate, non-multiplexed address and data buses and also the first with memory management and wide protection abilities.	The 80386-instruction set, programming model, and binary encoding are still the common denominator for all 32-bit x86 processors, which is termed the i386-architecture, x86 or IA-32 depending on context.
It is an advance version of the 8086 microprocessor that is designed for multi user and multitasking environment,	It is an enhanced version of the 80286 microprocessor and includes a memory management unit is enhanced to provide memory paging.
The 80286 addresses 16 Mbyte of physical memory and 1 GBytes of	It has physical memory size of 4GBytes and addressed as a virtual

virtual memory by using its memory management system.	memory with up to 64 TBytes.

(3) Draw and explain the block diagram of the programmable interrupt controller 8259A.

- The 8259A is a Programmable Interrupt Controller (PIC) that is commonly used in microcomputer systems to manage and control interrupt requests from peripheral devices. It was widely used in early personal computers such as the IBM PC. The 8259A is designed to work with the 8085, 8086 and other similar microprocessors.



- **Data Bus Buffer:** This block is used to communicate between 8259 and 8085/8086 by acting as buffer. It takes the control word from 8085/8086 and send it to the 8259.

- **R/W Buffer:** This block works when the value of pin CS is 0. This block is used to flow the data depending upon the inputs of RD and WR.

- **Control Logic:** This block generates the control signals that are used to manage the interrupt request and response. It has pin called INTR. This is connected to other microprocessors for taking the interrupt request. The INT pin is used to give output.

- **<u>Cascade Buffer:</u>** To increase number of interrupt pin, we can cascade more number of pins, by using cascade buffer. When we are going to increase the interrupt capability, CSA lines are used to control multiple interrupts.
- **<u>Interrupt Status Register (ISR):</u>** This register contains the status of each interrupt request. It is used to determine which IR input line has generated the interrupt request.
- **<u>Interrupt Request Register (IRR):</u>** It store all interrupt level that are requesting for the interrupt service.
- **<u>Priority Resolver (PR):</u>** This block determines the priority of the interrupt requests and ensure that the highest priority interrupt request is serviced first.
- **<u>Interrupt Mask Register (IMR):</u>** This register is used to make or enable individual interrupt request. The IMR can be used to temporarily disable an interrupt request, For example, during a critical section of code execution.

Winter 2022

Q.1

(1) Explain Discuss various types of addressing mode in 8085.

Special Notes:
The "I" in the instruction indicates "Immediate Addressing Mode"
Hence the number in the instruction must be DATA.
Hereafter, when you see a number in any instruction look for an "I".
If "I" is present then the number is DATA else its an address.
The "X" in the instruction (LXI) indicates "Register Pair".

➢ Immediate Addressing Mode:

- In this mode, the **data** is specified in the instruction itself.

EX.

MVI A, 35H; Move immediately the value 35 into Accumulator.
; A ←35H

LXI B, 4000H; Move immediately the value 4000 into register pair BC.
; BC ←4000H

Advantage:

Programmer can easily identify the operands.

Disadvantages:

Always more than one byte hence requires more space.
The µP requires 2 or 3 machine cycles to fetch instruction hence slow.

➢ Register Addressing Mode:

- In this mode, the **data** is specified in registers.

EX.

MOV B, C; Move the contents of C-register into B-register.
; B ←C

INR B ; Increment the content of B-register.
; B ← B+1

Advantage:

The µP requires only one machine cycle to fetch instructions.

Disadvantages:

Operands can not be easily identified.

➢ Direct Addressing Mode:

- In this mode, the address of the operand is specified in the instruction itself.

EX.

LDA 2000H; Load the Accumulator with contents of location 2000.
; A ← [2000]

STA 2000H; Store the content of the accumulator at location 2000.
; [2000] ←A

Advantage:

Programmer can easily identify the address of operands.

Disadvantages:

These are 3 bytes instructions hence 3 fetch cycles require.

➢ In-Direct Addressing Mode:

- In this mode, the address of the operand is specified in the register.
- Hence, the instruction indirectly points to the operands.
- Even the memory pointer 'M' can be used as it is pointed by HL pair.

EX.

STAX B; Store the content of the accumulator at the location pointed by BC pair.
; [BC] ← A

INR M ; Increments the contents of the location pointed by HL pair.
; [HL] ← [HL]+1

Advantage:

Address of the operand is not fixed and hence it can be used in loop.
Size of the instruction is small as compared to direct addressing mode.

Disadvantages:

Requires initialization of the register pair hence requires at least one more instruction.

Special Notes:
Remember, during programming when you want to access only 1 or 2 locations, use Direct addressing mode as it is simpler.
But when you want to access a series of locations, use Indirect addressing mode.
Initialize the first address in a register pair.
Thereafter increment/decrement that pair in a loop to access a series of locations.

➢ Implied Addressing Mode:

- In this mode, the operand is implied in the instruction.

EX.

STC ; Set the carry flag in flag register.

; Cy ← 1

CMC; Complement the carry flag in flag register.

; [HL] ← [HL]+1

Advantage:

Instructions are generally only one byte.

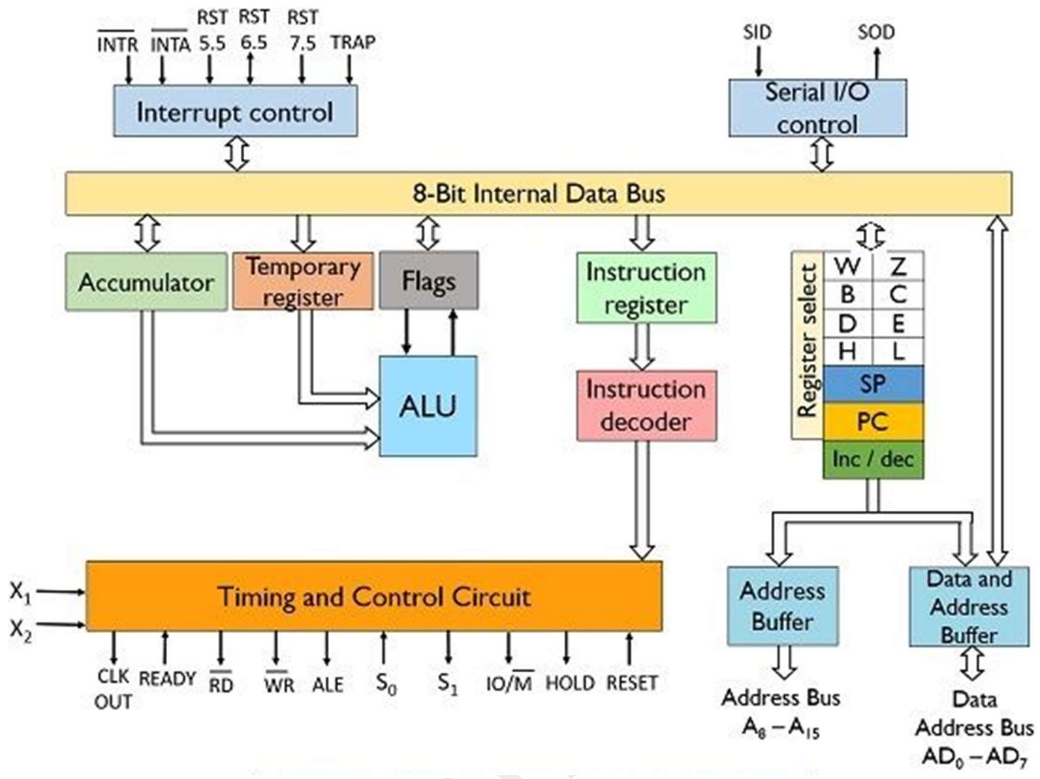
Disadvantages:

Programmer can not easily identify the value of the operand.

(2) What ae the advantages of an assembly language with comparison to high level language?

ASSEMBLY LEVEL LANGUAGE	HIGH-LEVEL LANGUAGE
It needs an assembler for conversion	It needs a compiler/interpreter for conversion
In this, we convert an Assembly level language to machine level language	In this, we convert a high-level language to Assembly level language to machine level language
It is machine dependent	It is machine-independent
In these mnemonics, codes are used	In this English statement is used
It supports low-level operation	It does not support low-level language
In this, it is easy to access hardware component	In this, it is difficult to access hardware component
In this more compact code	No compactness

(3) Draw the block diagram of internal architecture of 8085 and explain its working.



➢ **<u>ALU (Arithmetic and Logic Unit):</u>**

- ALU performs the computation function and it includes accumulator, temporary registers, arithmetic and logical circuits, and five flags.
- Temporary register holds the data temporarily during the arithmetic or logical operation of the processor.
- Accumulator stores the final result of the arithmetic or logical operation.
- Flags are the set of flip-flops. When the processor performs some operation, the flags are set or reset according to result of operation.

➢ **<u>Accumulator (register A):</u>**

- It is an 8-bit register and also it is a part of ALU.
- It is used to store intermediate results during arithmetic and logical operation

➢ Temporary Registers (W and Z):

- W and Z are the temporary registers. These are 8-bit processors.
- It is not accessible to the programmer. Temporary register holds the data temporarily during the arithmetic or logical operation of the processor.

➢ Instruction Register (IR):

- IR is and 8-bit register. It is also not accessible to the programmer.
- It stores the instruction that is currently being executed by the microprocessor. This instruction is fetched from memory and store in the IR.

➢ Instruction Decoder (ID):

- It is responsible for decoding the instruction stored in IR. It reads the opcode and determines what operation the microprocessor needs to perform.

➢ Register Array (B, C, D and E):

- B, C, D, E, H and L are the register array which is of 8-bits.
- These registers are available for programmers during the programming of 8085 processor. The programmer can store data in these register during program execution.
- We can use these register array in pairs such as BC, DE as 16-bit register.
- **Stack Pointer (SP):** SP is a 16-bit register that works as a memory pointer. It points to the stack which is the location in the R/W memory.
- **Program Count (PC):** It is used to sequence the execution of the instruction. It points to the memory address from which the next byte of information is to be fetched.

➢ **Timing and Control Unit:**

- Timing unit synchronizes the operation of the microprocessor with the clock.
- The control unit is responsible for generating signals for the microprocessor to communicate with peripherals.

➢ **Flags (C, Z, S, P and AC):**

- Flags registers consist of a combination of five flip-flops.
- Each flip-flop will hold the status of different state of the arithmetic and logical operation performed by microprocessor.
- **Carry** Set (1) if the last operation generates carry otherwise reset (0).
  Zero: If the result of the last operation of the processor is zero then this flag is set (1). Otherwise, the flag is reset (0).
  Sign: Set (1) if the MSB of the result of the last operation is 1 (as MSB=1 signifies negative), otherwise reset (0).
  Parity: Set (1) if the result of the last operation has even numbers of 1's (i.e. even parity) otherwise reset (0).
  Auxiliary Carry: Set (1) if the last operation yield carries from the lower half-word otherwise reset (0).

➢ **Interrupt Control:**

- To handle the interrupt in the microprocessor there are various interrupt control signals such as INTR, RST 5.5, RST 6.5, RST 7.5, INTA.

➢ **Serial I/O Controls:**

- For serial data transmission, SID and SSIO are two serial I/O control signals available in the 8085 processors.

Q. 2

(1) How does the microprocessor differentiate among a positive or negative number and a bit pattern.

- The leftmost bit or the most significant bit of the number is the sign bit. It tells the processor about the sign of the number – that is, whether the number is positive or negative. 0 in the sign bit represents a positive value and 1 represents a negative value.
- For ex., consider a decimal number 4. Binary representation of 4 is 00000100 and the binary representation of -4 is 11111100. The 1s in the leading bits tell the processor that the number is a negative number.
- A bit pattern is simply a sequence of 1s and 0s. The microprocessor doesn't differentiate between a bit pattern and a number since it treats all data as binary patterns of bits.

(2) LOOP: LXI H, 1234H
DCX H
JNZ LOOP
Find out the mistake(s) in the above program and write the correct program so that it does not become infinite loop.

- The mistake in the original program was that there was no counter to control the number of iterations in the loop, leading to an infinite loop. The corrected program uses a counter initialized with the value 10 (9H in hexadecimal) and decrements it with each iteration of the loop using the DCR instruction. The JNZ instruction checks if the counter has reached zero before branching back to the LOOP label, thereby ensuring that the loop executes only 10 times.

```
      MVI C,09H; set counter to 10
LOOP: LXI H,1234H
      DCX H;
      DCR C;
      JNZ LOOP;
```

Q. 3

(1) What are the states of the Auxiliary Carry (AC), Carry (CY), sign(S) and parity (P) flags after executing the following 8085 program?
MVI L, 5DH
MVI A, 6BH
ADD L

➢

```
1111 111  (Carry)
----------------------
0101 1101 (5DH)
0110 1011 (6BH)
-----------------------
1000 0000
```

➢ **MVI L, 5DH;** loads the lower byte of register pair HL with 5D.
  - No any flag affected.

➢ **MVI A, 6BH;** loads register A with 6B.
  - No any flag affected.

➢ **ADD L:** adds the contents of L (5D) to A (6B), and stores the result in A (C8).
AC flag: Set (there was a carry from bit 3 to bit 4 during the addition, indicating that the result is invalid for BCD arithmetic)
CY flag: Not affected
S flag: Set (the result is negative)
P flag: Set (the result has an even number of set bits)

(2) Explain 8085 Programming model and classify instruction set on the basis of different addressing modes.

➢ **<u>8085 Programming Model:</u>**

- The 8085 microprocessors programming model consists of six registers, namely the accumulator (A), the program counter (PC), the stack pointer (SP), the flag register (FLAGS), the temporary register (register pair H and L), and the memory address register (MAR).
- The accumulator (A) is an 8-bit register that stores data and arithmetic operations are performed on this register.
- The program counter (PC) is a 16-bit register that holds the address of the next instruction to be executed.
- The stack pointer (SP) is also a 16-bit register those points to the top of the stack in memory.
- The flag register (FLAGS) is an 8-bit register that stores the status of the arithmetic and logic operations.
- The temporary register (register pair H and L) is a 16-bit register used for holding intermediate values during arithmetic and logical operations.
- The memory address register (MAR) is a 16-bit register that holds the address of the memory location to be accessed.

➢ **<u>Instruction Set:</u>** Refer : Q.(1) => (1)

(4) 2100 LXI H, 1234H
MVI A, 55H
ADD M
What is the size of ADD M instruction? Name the machine cycles. Draw machine cycle and T-state diagram and specify the content of address bus, data bus and control signals *RD, *WR, IO/*M and ALE signals and status signals S1 and S0 for every T states of ADD M instruction only

Q. 3

(1) What are the states of the Auxiliary Carry (AC), Carry (CY), sign(S) and parity (P) flags after executing the following 8085 program?

```
MVI A, A9H
MVI B, 57H
ADD B
ORA A
```

➢

```
   1111 111  (Carry)
   ----------------------
   1010 1001 (A9H)
   0101 0111 (57H)
   -----------------------
1  0000  0000 (Result)
```

➢ **ORA A;**

```
    0000 0000 (Result)
    1010 1001
-----------------------
    1010 1001
```

(2) Explain One byte, two byte and three bytes and write short note on different types of instruction Set.

➢ **<u>1-byte:</u>**

- A one-byte instruction includes the opcode and the operand in the same byte.
- <u>Examples:</u>
  MOV A, B
  ADD B
  RAL

➢ **<u>2-byte:</u>**

- In a 2-byte instruction, the 1st-byte specifies the opcode and the 2nd-byte specifies the operand.
- <u>Examples:</u>
  MVI B, 05
  IN 01 etc.

➢ **<u>3-byte:</u>**

- In a 3-byte instruction, the 1st byte specifies the opcode, and the following two bytes specify the 16-bit address such that the 2nd byte is a low order address and the 3rd-byte is high order address.
- <u>Examples:</u>

  LXI H, 2400H
  LDA 2500H
  JMP 2085H etc.

➢ Types of Instruction set of 8085 based on Addressing Mode: **<u>Refer Q.1.(1)</u>**

(3) Specify the addressing mode, required Machine cycles, T-States and function for following instructions :

1. MVI M, 45H
2. RAL
3.LHLD 2300H

➢ **<u>MVI M, 45H</u>**
- Addressing mode: Direct addressing mode
- Required machine cycles: 3
- Required T-states: 10
- Function: Move immediate 8-bit data 45H to memory location specified by the content of HL register pair.

➢ **<u>RAL</u>**
- Addressing mode: (Impllied) Accumulator addressing mode
- Required machine cycles: 1
- Required T-states: 4
- Function: Rotate the contents of the accumulator left through carry.

➢ **<u>LHLD 2300H</u>**
- Addressing mode: Direct addressing mode
- Required machine cycles: 3
- Required T-states: 16
- Function: Load the contents of the memory location specified by the 16-bit address 2300H into the HL Pair register.

Q.4

(1) Difference between RLC and RAL instructions.

- ROTATE is a logical operation of the 8085 microprocessors. It is a 1-byte instruction.
- This instruction does not require any operand after the opcode. It operates the content of the accumulator and the result is also stored in the accumulator. The Rotate instruction is used to rotate the bits of accumulator.
- Types of ROTATE Instruction: There are 4 categories of the ROTATE instruction: Rotate accumulator left (RLC), Rotate accumulator left through carrying (RAL), Rotate accumulator right (RRC), Rotate accumulator right through carry (RAR).

- **Rotate left with carry (RLC) –** In this instruction, Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. CY is modified according to bit D7.
  For example: - A = D7 D6 D5 D4 D3 D2 D1 D0

  //before the instruction
  A = 10101010; CY=0
  //after 1st RLC
  A = 01010101; CY=1
  //after 2nd RLC
  A = 10101010; CY=0

- **Rotate Arithmetic Left (RAL) –** In this instruction, Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7.
  example: A = D7 D6 D5 D4 D3 D2 D1 D0

  //before the instruction
  A = 10101010; CY=0
  //after 1st RAL
  A = 01010100; CY=1
  //after 2nd RAL
  A = 10101001; CY=0

(2) Differentiate between maskable and non-maskable interrupts.

Maskable Interrupt	**Non Maskable Interrupt**
Maskable interrupt is a hardware Interrupt that can be disabled or ignored by the instructions of CPU.	A non-maskable interrupt is a hardware interrupt that cannot be disabled or ignored by the instructions of CPU.
When maskable interrupt occur, it can be handled after executing the current instruction.	When non-maskable interrupts occur, the current instructions and status are stored in stack for the CPU to handle the interrupt.
Maskable interrupts help to handle lower priority tasks.	Non-maskable interrupt help to handle higher priority tasks such as watchdog timer.
In maskable interrupts, response time is high.	In non-maskable interrupts, response time is low.
It may be vectored or non-vectored.	All are vectored interrupts.
Operation can be masked or made pending.	Operation Cannot be masked or made pending.
RST6.5, RST7.5, and RST5.5 of 8085 are some common examples of maskable Interrupts.	Trap of 8085 microprocessor is an example for non-maskable interrupt.

(3) What is flag register? Enlist and explain various types of flag register.

- The flag register in the 8085 microprocessor is a special register that holds the status of various operations performed by the microprocessor.
- It is a vital component of the 8085 microprocessor, as it provides information about the result of arithmetic and logical operations, and is used to control the flow of program execution.
- The flag register is 8-bit register, with each bit representing a specific flag.
- Flag register includes five flip-flops, which are set or reset after an operation according to the data conditions of the result in the accumulator and other registers.
- The microprocessor uses these flags to set and test data conditions.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

- **S (Sign) Flag:** Indicates the sign of the result of an arithmetic operation. If the result is negative, the flag is set (1), otherwise it is reset (0).
- **Z (Zero) Flag:** Indicates result of mathematical or logical operation is zero or not. If result is zero this flag will be set (1), otherwise reset (0).
- **AC (Auxiliary Carry) Flag:** Indicates if there is carry out from lower nubble to the upper nibble during arithmetic operation. This flag is used in binary-coded decimal (BCD) arithmetic operations.
  (1) => If carry generated, (0) => If there is no carry generated.
- **P (Parity) Flag:** Parity is the number of 1's in a number. If the is even then that number is known as even parity number.
  If the number is odd then that number is known as an odd parity number.
  This flag indicates whether the current result is of even parity (set) (1) or of odd parity (reset) (0).
- **CY (Carry) Flag:** Indicates if there is carry out from most significant bit during an arithmetic operation. (1)=> last operation generate carry, (0)=> no carry from last operation

Q.4

(1) Difference between RRC and RAR instruction.

- ROTATE is a logical operation of the 8085 microprocessors. It is a 1-byte instruction.
- This instruction does not require any operand after the opcode. It operates the content of the accumulator and the result is also stored in the accumulator. The Rotate instruction is used to rotate the bits of accumulator.
- Types of ROTATE Instruction: There are 4 categories of the ROTATE instruction: Rotate accumulator left (RLC), Rotate accumulator left through carrying (RAL), Rotate accumulator right (RRC), Rotate accumulator right through carry (RAR).

- **Rotate Right with carry (RRC) –** In this instruction, Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D0.
  For example: - A = D7 D6 D5 D4 D3 D2 D1 D0

  //before the instruction
  A = 10000001; CY=0
  //after 1st RRC
  A = 11000000; CY=1
  //after 2nd RRC
  A = 01100000; CY=0

- **Rotate Arithmetic Right (RAR) –** In this instruction, Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0.
  For example: A = D7 D6 D5 D4 D3 D2 D1 D0

  //before the instruction
  A = 10000001; CY=0
  //after 1st RAR
  A = 01000000; CY=1
  //after 2nd RAR
  A = 10100000; CY=0

(2) What is vectored and non-vectored interrupts?

- **Vectored Interrupts:** In a vectored interrupt, the microprocessor is directed to a specific location in memory where the ISR is located. The interrupting device sends a unique interrupt request signal to the microprocessor to identify the ISR's memory location. This signal is called the interrupt vector.
    - When the microprocessor receives the interrupt request, it saves the current program counter on the stack and fetches the address of the ISR from the interrupt vector. The ISR then executes, and after it completes its task, the microprocessor retrieves the program counter from the stack and resumes execution of the interrupted program.
- **Non-vectored Interrupts:** In a non-vectored interrupt, the microprocessor does not know the ISR's memory location. When the interrupting device sends an interrupt request, the microprocessor searches for the ISR's memory location by checking all possible memory locations until it finds the ISR's address.
    - Since the microprocessor has to search for the ISR's memory location, non-vectored interrupts are slower and less efficient than vectored interrupts.
- Overall, vectored interrupts are faster and more efficient than non-vectored interrupts.

(3) Describe the functions of (1) READY PIN (2) ALE (3) HOLD (4) X1 and X2 (5) SID and SOD (6) IO/M 22. (7) HLDA

- **READY PIN:** The READY pin is an input pin used to synchronize the operation of the microprocessor with other slower devices in the system. When the READY pin is high, it indicates that the slower device is ready to receive or transmit data. The microprocessor will wait until the READY pin goes high before proceeding with its operations.
- **ALE:** ALE stands for Address Latch Enable and it is a control signal used to latch the address into external latch. The microprocessor uses this signal to indicate that the address on the address bus is stable and can be latched into an external latch for further use.
- **HOLD:** The HOLD signal is an input signal used to pause the microprocessor's operation so that another device can take control of the system bus. When a HOLD signal is received, the microprocessor stops executing instructions and releases control of the system bus.
- **X1 and X2:** X1 and X2 are the two input pins of the microprocessor used to connect an external crystal oscillator or clock source. The microprocessor uses these pins to receive a clock signal that synchronizes its internal operations.
- **SID and SOD:** SID stands for Serial Input Data and SOD stands for Serial Output Data. These pins are used to implement serial data transfer in the microprocessor. The SID pin is used to input serial data into the microprocessor, and the SOD pin is used to output serial data from the microprocessor.
- **IO/M:** The IO/M pin is an output pin used to indicate whether the microprocessor is performing a memory or I/O operation. When this pin is high, it indicates that the microprocessor is performing an I/O operation. When it is low, it indicates that the microprocessor is performing a memory operation.
- **HLDA:** HLDA stands for Hold Acknowledge, and it is an output signal used to acknowledge that the microprocessor has received the HOLD signal. When the microprocessor receives a HOLD signal, it responds with a HLDA signal to indicate that it has relinquished control of the system bus.

Q.5

(1) List features of 80386 microprocessor.

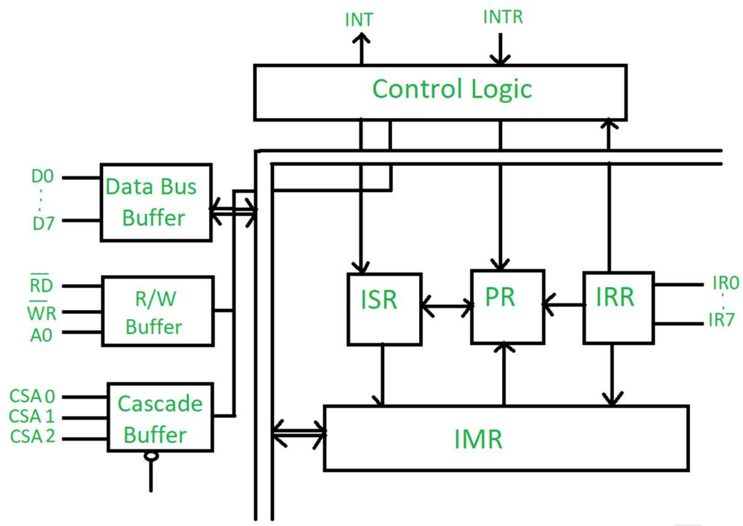
➢ The 80386 microprocessor is a 32-bit microprocessor and an improvement over the 80286 microprocessors. Some of its features are:

- **32-bit architecture:** The 80386 microprocessor is a 32-bit microprocessor, which means it can address 2^32 or 4 GB of memory.
- **Paging support:** The 80386 supports paging, which allows the memory to be divided into small, fixed-sized pages, making it easier to manage memory.
- **Protected mode:** The 80386 supports protected mode, which allows the operating system to protect programs from each other and provides a secure environment for system processes.
- **Virtual memory support:** The 80386 supports virtual memory, which allows the memory to be extended beyond the physical RAM of the system.
- **Multitasking support:** The 80386 supports multitasking, which means that it can run multiple programs simultaneously.
- **Instruction set extensions:** The 80386 introduces several new instructions and extensions to the x86 instruction set, such as new addressing modes and support for floating-point operations.
- **Improved performance:** The 80386 is faster and more efficient than its predecessors, with an improved architecture and support for pipelining.
- **Lower power consumption:** The 80386 uses less power than its predecessors, making it more energy-efficient.
- **Backward compatibility:** The 80386 is backward compatible with its predecessors, which means that it can run software written for older x86 processors.

(2) Draw block diagram of SUN SPARC architecture.

- The SUN SPARC (Scalable Processor Architecture) is a RISC-based processor architecture developed by Sun Microsystems (now Oracle Corporation) in the mid-1980s. The architecture consists of several functional units that work together to execute instructions. The basic block diagram of the SUN SPARC architecture includes the following units:
- **Instruction Fetch Unit (IFU):** The IFU fetches instructions from memory and sends them to the instruction queue.
- **Instruction Queue:** The instruction queue holds up to four instructions, allowing for some degree of instruction-level parallelism.
- **Register File:** The register file contains 32 general-purpose registers, each 32 bits wide. It also includes two special-purpose registers, the Program Counter (PC) and the Condition Codes Register (CCR).
- **Arithmetic Logic Unit (ALU):** The ALU performs arithmetic and logical operations on data stored in registers.
- **Floating-Point Unit (FPU):** The FPU performs arithmetic operations on floating-point numbers.
- **Memory Management Unit (MMU): The** MMU manages virtual memory by translating virtual addresses to physical addresses.
- **Cache Controller:** The cache controller manages the level 1 (L1) cache, which is a small, fast memory used to store frequently accessed data.
- **Bus Interface Unit (BIU):** The BIU manages communication between the processor and the system bus.

(3) Explain internal block diagram of 8259A.



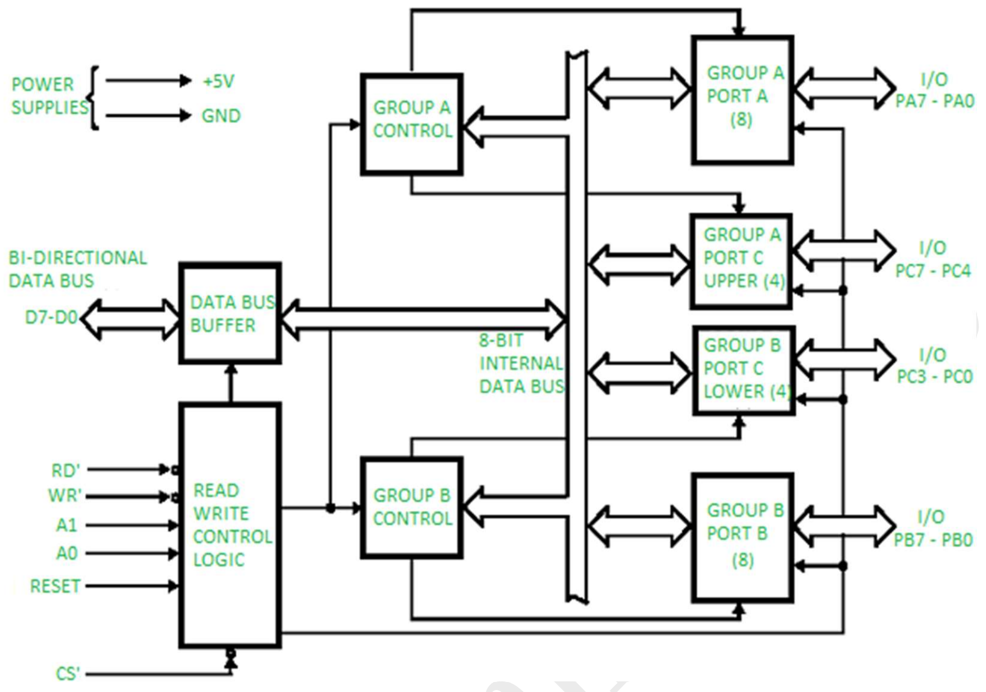
- **Interrupt Request Register (IRR):** It stores those bits which are requested for their interrupt services.
- **Interrupt Service Register (ISR):** It stores the interrupt levels which is currently being served.
- **Interrupt Mask Register (IMR):** It stores interrupt levels that have to be masked. These interrupt levels are already accepted by the 8259 microprocessor.
- **Priority Resolver (PR):** It examines all the 3 registers and sets the priority of interrupts and sets the interrupt levels in ISR which has the highest priority and the rest of the interrupt bit is IRR which is already accepted.
- **Data Bus Buffer:** The data bus buffer allows the 8085 to send control words to the 8259A and read a status word from the 8259 Block Diagram. The 8-bit data bus buffer also allows the 8259A to send interrupt opcode and address of the interrupt service subroutine to the 8085.
- **Read/Write Logic:** The RD and WR inputs control the data flow on the data bus when the device is selected by asserting its chip select (CS) input low.
- **Control Logic:** The Control Logic has an input and an output line. When the 8259A is properly enabled, an interrupt request will cause the 8259A to assert its INT output pin high. If this pin is connected to the INTR pin of an 8085 and if the 8085 Interrupt Enable (IE) flag is set, then this high signal will cause the 8085 to respond with an interrupt.
-

Q.5

(1) List features of 80486 microprocessor.

- **32-bit architecture:** The 80486 is a 32-bit microprocessor, which means it handle data and instruction in 32-bit chunks.
- **Clock Speed:** In 80486 the Clock speed ranging from 16 MHz to 100 MHz.
- **Transistors:** In 80486 microprocessor there are 1.2 million transistors.
- **Pipelined Architecture:** The 80486 uses a Pipelined architecture with dual instruction pipeline.
- **MMU:** Built-in memory management unit (MMU) for virtual memory and protection.
- **Cache Memory:** Built-in cache memory (8 KB or 16 KB) for faster access to frequently used instructions and data
- **Data and address bus:** 32-bit external data bus and 32-bit address bus
- **Modes:** Support for both real and protected mode of operation
- **Compatibility:** Instruction set compatible with 8086, 80186, 80286, and 80386 microprocessors
- **SMM:** Support for system management mode (SMM) for power management and system security.

(3) Draw and explain the internal block Diagram of 8255A.



➢ **Data Bus Buffer:**

- The tri-state bi-directional buffer is used to interface the internal data bus of 8255 to the system data bus.
- Input or Output instructions executed by the CPI either Read data from, or write data into the buffer.
- Output data from the CPU to the ports or control register, and input data to the CPU from the ports or status register are all passed through the buffer.

➢ **Control Logic:**

- The control logic block accepts control bus signals as well as inputs from the address bus, and issues commands to the individual group control block (Group A control and Group B control).
- It issues appropriate enabling signals to access the requires data/control words or status words. The input pins for the control logic section are described here.

➢ **<u>Group A and Group B Controls:</u>**

- Each of Group A and B control blocks receives control words from the CPU and issues appropriate commands to the ports associated with it.
- Group A control block controls Port A and PC_7-PC_4 while Group B control block control port B and PC_3-PC_0.
- **Port A:** This has an 8-bit latched and buffered output and an 8-bit input latch. It can be programmed in three modes: mode0, mode1, mode2.
- **Port B:** This has an 8-bit data I/O /latch/buffer and 8-bit data input buffer. It can be programmed in mode0 and mode 1.
- **Port C:** This has one 8-bit unlatched input buffer and an 8-bit output latch/buffer. Port C can be separated into two parts and each can be used as control signals for Port A and Port B in the handshake mode. It can be programmed for bit set/reset operation.

SUMMER 2022

Q.1

(5) Explain the Flag Register in 8085 microprocessors.

- The flag register in the 8085 microprocessor is a special register that holds the status of various operations performed by the microprocessor.
- It is a vital component of the 8085 microprocessor, as it provides information about the result of arithmetic and logical operations, and is used to control the flow of program execution.
- The flag register is 8-bit register, with each bit representing a specific flag.
- Flag register includes five flip-flops, which are set or reset after an operation according to the data conditions of the result in the accumulator and other registers.
- The microprocessor uses these flags to set and test data conditions.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

- **S (Sign) Flag:** Indicates the sign of the result of an arithmetic operation. If the result is negative, the flag is set (1), otherwise it is reset (0).
- **Z (Zero) Flag:** Indicates result of mathematical or logical operation is zero or not. If result is zero this flag will be set (1), otherwise reset (0).
- **AC (Auxiliary Carry) Flag:** Indicates if there is carry out from lower nubble to the upper nibble during arithmetic operation. This flag is used in binary-coded decimal (BCD) arithmetic operations.
  (1) => If carry generated, (0) => If there is no carry generated.
- **P (Parity) Flag:** Parity is the number of 1's in a number. If the is even then that number is known as even parity number.
  If the number is odd then that number is known as an odd parity number. This flag indicates whether the current result is of even parity (set) (1) or of odd parity (reset) (0).
- **CY (Carry) Flag:** Indicates if there is carry out from most significant bit during an arithmetic operation. (1)=> last operation generate carry, (0)=> no carry from last operation

(6) Explain the following pins of 8085: (i) INTR (ii) HOLD (iii) SOD (iv) READY

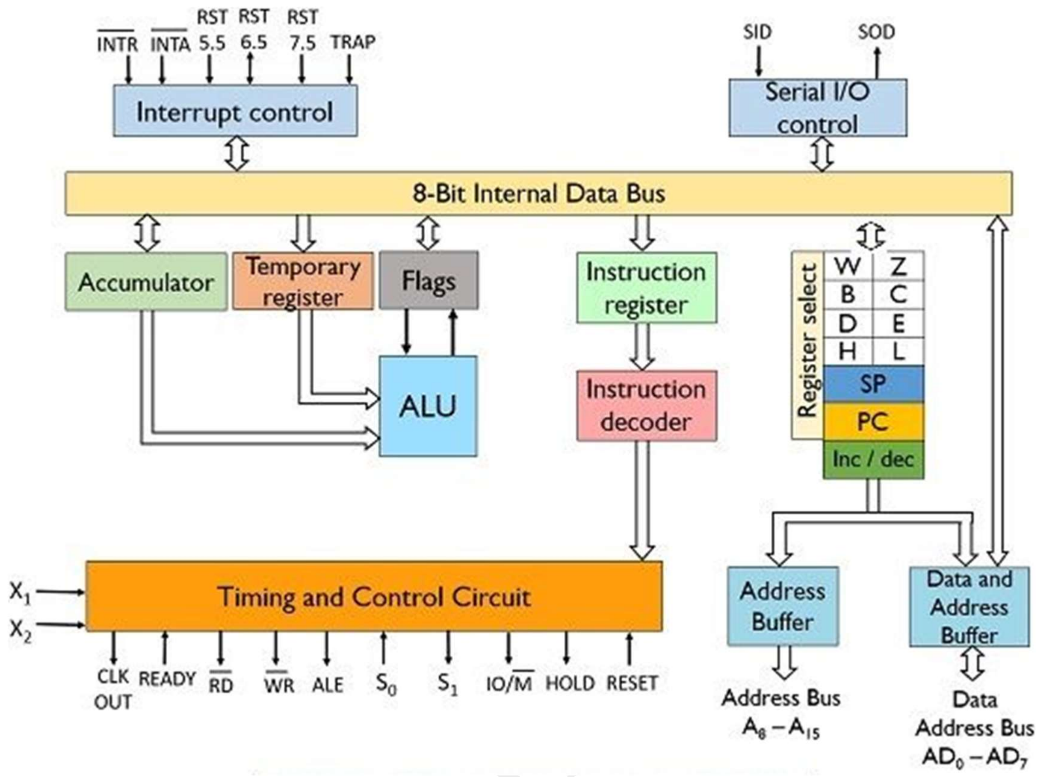
- **INTR (Interrupt Request):** The INT pin allows an external device to interrupt the normal operation of the 8085 microprocessors. When this happens, the microprocessor stops what it's doing and jumps to a specific section of the program to handle the interrupt. After handling interrupt, the microprocessor goes back to where it left off.

- **HOLD:** The HOLD pin allows an external device to temporarily stop the operation of the 8085 microprocessor and take control of the bus. When the HOLD pin is activated, the microprocessor stops what it's doing and release the bus. When the external device is finished, the HOLD pin is turned off and the microprocessor goes back to where it left off.

- **SOD (Serial Output Data):** The SOD pin is used to output serial data from the 8085 microprocessors. This pin is typically used for communication with external devices, such as a serial display or a serial printer.

- **READY:** The READY pin is used to indicate that an external device is ready to accept data or has complete a data transfer. When external device is ready, it activates the READY pin, and the microprocessor begin the transfer. When the transfer is complete, the external device deactivates the READY pin to signal that it has finished. Microprocessor checks the READY pin before starting the transfer.

(7) Draw the block diagram of internal architecture of 8085 and explain its working.



➢ ALU (Arithmetic and Logic Unit):

- ALU performs the computation function and it includes accumulator, temporary registers, arithmetic and logical circuits, and five flags.
- Temporary register holds the data temporarily during the arithmetic or logical operation of the processor.
- Accumulator stores the final result of the arithmetic or logical operation.
- Flags are the set of flip-flops. When the processor performs some operation, the flags are set or reset according to result of operation.

➢ Accumulator (register A):

- It is an 8-bit register and also it is a part of ALU.
- It is used to store intermediate results during arithmetic and logical operation

➢ <u>Temporary Registers (W and Z):</u>

- W and Z are the temporary registers. These are 8-bit processors.
- It is not accessible to the programmer. Temporary register holds the data temporarily during the arithmetic or logical operation of the processor.

➢ <u>Instruction Register (IR):</u>

- IR is and 8-bit register. It is also not accessible to the programmer.
- It stores the instruction that is currently being executed by the microprocessor. This instruction is fetched from memory and store in the IR.

➢ <u>Instruction Decoder (ID):</u>

- It is responsible for decoding the instruction stored in IR. It reads the opcode and determines what operation the microprocessor needs to perform.

➢ <u>Register Array (B, C, D and E):</u>

- B, C, D, E, H and L are the register array which is of 8-bits.
- These registers are available for programmers during the programming of 8085 processor. The programmer can store data in these register during program execution.
- We can use these register array in pairs such as BC, DE as 16-bit register.
- **Stack Pointer (SP):** SP is a 16-bit register that works as a memory pointer. It points to the stack which is the location in the R/W memory.
- **Program Count (PC):** It is used to sequence the execution of the instruction. It points to the memory address from which the next byte of information is to be fetched.

- **Timing and Control Unit:**
  - Timing unit synchronizes the operation of the microprocessor with the clock.
  - The control unit is responsible for generating signals for the microprocessor to communicate with peripherals.

- **Flags (C, Z, S, P and AC):**
  - Flags registers consist of a combination of five flip-flops.
  - Each flip-flop will hold the status of different state of the arithmetic and logical operation performed by microprocessor.
  - **Carry** Set (1) if the last operation generates carry otherwise reset (0).

    Zero: If the result of the last operation of the processor is zero then this flag is set (1). Otherwise, the flag is reset (0).

    Sign: Set (1) if the MSB of the result of the last operation is 1 (as MSB=1 signifies negative), otherwise reset (0).

    Parity: Set (1) if the result of the last operation has even numbers of 1's (i.e. even parity) otherwise reset (0).

    Auxiliary Carry: Set (1) if the last operation yield carries from the lower half-word otherwise reset (0).

- **Interrupt Control:**
  - To handle the interrupt in the microprocessor there are various interrupt control signals such as INTR, RST 5.5, RST 6.5, RST 7.5, INTA.

- **Serial I/O Controls:**
  - For serial data transmission, SID and SSIO are two serial I/O control signals available in the 8085 processors.

Q. 2

(1) Explain following instructions.

(i) LHLD (ii) RAA (iii) DAA

➢ **<u>LHLD (Load H and L Direct):</u>**

- LHLD is an instruction in the 8085 microprocessor that is used to load the contents of a memory location into the H and L registers.
- The instruction has the following format: LHLD address, where "address" is the memory location where the data is stored.

➢ **<u>RAL (Rotate Accumulator Left):</u>**

- RAL is an instruction in the 8085 microprocessor that is used to rotate the contents of the accumulator one bit to the left.
- The instruction shifts the contents of the accumulator one bit to the left, and the most significant bit is shifted into the carry flag.

➢ **<u>DAA (Decimal Adjust Accumulator):</u>**

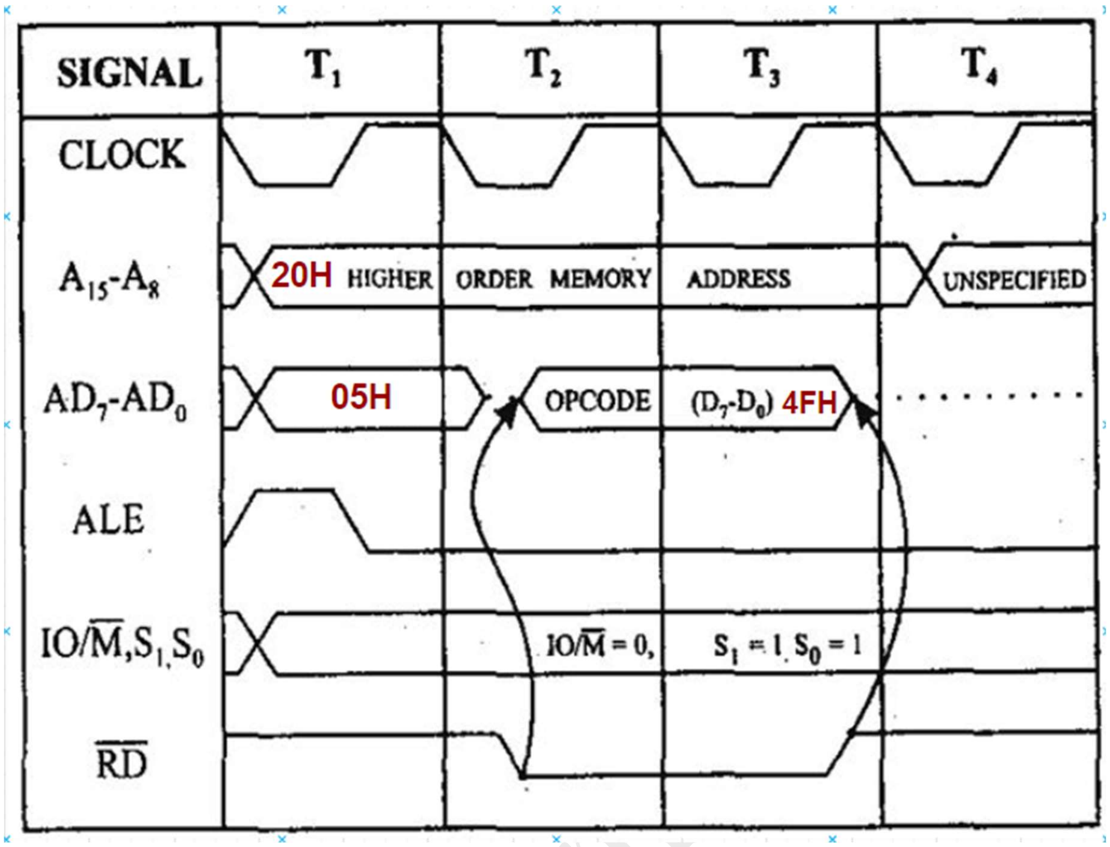
- DAA is an instruction in the 8085 microprocessor that is used to adjust the contents of the accumulator after performing an arithmetic operation.
- The instruction is used to correct the contents of the accumulator so that the result is in proper decimal format.
- The instruction takes into account the state of the lower nibble and upper nibble of the accumulator, as well as the carry flag, to determine the proper adjustment to be made to the contents of the accumulator.

(2) Explain demultiplexing of data and address bus of 8085.

- Demultiplexing of data and address bus in 8085 microprocessor refers to the process of separating the data and address signals into separate lines for use by the microprocessor and external devices.
- The 8085 microprocessor has a multiplexed data-address bus, which means that the data and address signals are transmitted on the same set of lines.
- Demultiplexing is necessary because the microprocessor needs to access both data and address information, and the external devices need to access only the data or address information, but not both simultaneously.
- Demultiplexing of the data and address bus is accomplished through the use of a demultiplexer circuit.
- The demultiplexer circuit uses control signals from the microprocessor to separate the data and address signals into separate lines.
- When the microprocessor needs to access data, the demultiplexer circuit separate the data signals from the address signals and route them to appropriate destinations.
- Similarly, when microprocessor needs to access the address information, the demultiplexer circuit separates the address signals from the data signals and routes them to appropriate destinations.
- The demultiplexing of the data and address bus in the 8085 is important because it allows the microprocessor to access both data and address information efficiently.
- It allows external devices to access the data or address information they need without interfering with the microprocessor's operations.
- The demultiplexing of the data and address bus in the 8085 microprocessor is a key feature that contributes to its overall performance and functionality.

(3) Explain the timing diagram of the instruction MOV C, A (4FH) stored in location 2005H is being fetched. Define T-state, Machine Cycle, and Instruction cycle.

- **T-state:** A T-state, also known as a clock cycle, is the basic unit of time in the 8085 microprocessors. The duration of a T-state is determined by the clock frequency of the 8085. In the 8085, one T-state lasts for 1/2 clock period.
- **Machine cycle:** A machine cycle is a sequence of T-states that make up a complete instruction execution. Each instruction in the 8085 microprocessor requires a specific number of machine cycles to be executed. The number of machine cycles required for a particular instruction depends on the instruction itself and the number of operands it requires.
- **Instruction cycle:** An instruction cycle is the total time required to execute an instruction in the 8085 microprocessors. An instruction cycle is made up of one or more machine cycles. The number of machine cycles in an instruction cycle depends on the instruction being executed.
- The timing diagram of the instruction MOV C, A (4FH) in the 8085 microprocessor shows the sequence of events that occur during the execution of the instruction. A timing diagram is a graphical representation of the timing signals generated by the microprocessor during the execution of an instruction.
- In the case of the instruction MOV C, A (4FH) stored in location 2005H, the instruction cycle would involve the following steps:
  - **Fetching the instruction:** The 8085 microprocessor fetches the instruction (4FH) from the memory location 2005H and stores it in the instruction register.
  - **Decoding the instruction:** The 8085 microprocessor decodes the instruction (MOV C, A) to determine the operation to be performed.
  - **Execution of the instruction:** The 8085 microprocessor transfers the contents of the accumulator (A) to the (C).
  - **Completion of the instruction:** The 8085 microprocessor increments the program counter to point to the next instruction in memory.


SIGNAL
T1
T2
T3
T4
CLOCK
A15-A8
20H HIGHER
ORDER MEMORY
ADDRESS
UNSPECIFIED
AD7-AD0
05H
OPCODE
(D7-D0) 4FH
ALE
IO/M̄,S1,S0
IO/M̄ = 0,
S1 = 1, S0 = 1
RD̄

(8) Explain interfacing of 4KB EPROM with 8085 using decoder and gates as required. Assume starting address as 0000H.

- The process of interfacing a 4KB EPROM with the 8085 microprocessor involves the following steps:
  - Decoding the address range of the EPROM using a decoder circuit.
  - Using gates to control the data transfer between the 8085 microprocessor and the EPROM.
  - Connecting the EPROM to the 8085 microprocessor's address and data bus.
- Interfacing of a 4KB EPROM with the 8085 microprocessor involves connecting the EPROM to the microprocessor's address and data bus. The starting address of the EPROM is assumed to be 0000H in this case.
- The first step in interfacing the EPROM with the 8085 microprocessor is to decode the address range of the EPROM. A decoder circuit is used to determine the specific address range of the EPROM. The decoder circuit receives the address signals from the 8085 microprocessor and generates an enable signal for the EPROM when the address falls within the EPROM's address range.
- Next, a set of gates such as AND gates or OR gates are used to control the data transfer between the 8085 microprocessor and the EPROM. The gates are connected to the data bus and the read and write signals from the 8085 microprocessors. When the 8085 microprocessor wants to read data from the EPROM, it asserts the read signal and the gates allow the data from the EPROM to be transmitted to the data bus. When the 8085 microprocessor wants to write data to the EPROM, it asserts the write signal and the gates allow the data from the data bus to be transmitted to the EPROM

Q. 3

(1) Write a program to find 2's complement of a number stored at 2050H and store result at 2055H.

➢ **Program:**

- **MVI A, 00H;** load the accumulator with 0
- **LDA 2050H;** load the accumulator with the value at 2050H
- **CMA;** complement the accumulator
- **INR A;** increment the accumulator
- **STA 2055H;** store the result in 2055H
- **HLT;** stop the program

➢ **Explanation:**

- The first instruction MVI A, 00H loads the accumulator with 0.
- The second instruction LDA 2050H loads the accumulator with the value stored at the memory location 2050H.
- The third instruction CMA complements the contents of the accumulator, i.e., it inverts all the bits of the contents of the accumulator.
- The fourth instruction INR A increments the contents of the accumulator by 1.
- The fifth instruction STA 2055H stores the result in the memory location 2055H.
- The final instruction HLT stops the program execution.

(2) Comparison Between memory mapped I/O and I/O mapped I/O.

Memory Mapped IO	IO Mapped IO
IO devices are accessed like any other memory location.	They cannot be accessed like any other memory location.
They are assigned with 16-bit address values.	They are assigned with 8-bit address values.
The instruction used are LDA and STA, etc.	The instruction used are IN and OUT.
Cycles involved during operation are Memory Read, Memory Write.	Cycles involved during operation are IO read and IO writes in the case of IO Mapped IO.
Any register can communicate with the IO device in case of Memory Mapped IO.	Only Accumulator can communicate with IO devices in case of IO Mapped IO.
No separate control signal required since we have unified memory space in the case of Memory Mapped IO.	Special control signals are used in the case of IO Mapped IO.
Arithmetic and logical operations are performed directly on the data in the case of Memory Mapped IO.	Arithmetic and logical operations cannot be performed directly on the data in the case of IO Mapped IO.

(3) What are interrupts? List and explain the interrupt available in microprocessor 8085?

- Interrupts are signals that are sent to the microprocessor to temporarily stop its normal execution of the program and jump to a specific section of the program to handle a specific task. Interrupts are used to allow external devices to request service from the microprocessor and to handle events that require immediate attention, such as a timer overflow or a key press.
- In the 8085 microprocessors, when an interrupt signal is received, the microprocessor saves the address of the next instruction on the stack and jumps to the corresponding interrupt service routine. After the interrupt service routine is executed, the microprocessor returns to the instruction that was being executed before the interrupt request.

- The 8085 microprocessor has five interrupt signals, which are:
  - **TRAP:** TRAP is a non-maskable interrupt, which means that it cannot be disabled. It has the highest priority among all the interrupts and is used for critical events that require immediate attention.
  - **RST 7.5:** This interrupt has a higher priority than the other interrupts and is used for high-priority tasks.
  - **RST 6.5:** This interrupt has a lower priority than the RST 7.5 interrupt but a higher priority than the other interrupts. It is used for medium-priority tasks.
  - **RST 5.5:** This interrupt has a lower priority than the RST 6.5 interrupt but a higher priority than the other interrupts. It is used for low-priority tasks.
  - **INTR:** This is a maskable interrupt, which means that it can be disabled. It is used for general-purpose interrupt requests and has the lowest priority among all the interrupts.

Q. 3

(1) Explain concept of stack.

- The stack is a memory area in the microprocessor used to temporarily store data such as part of process, such as subroutine or interrupt handling.
- It is implemented in LIFO (Last in First Out) data structure, where the last data item stored is the first one retrieved. This enables the microprocessor to handle nested subroutines and multiple interrupt request efficiently.
- Interfacing the stack with the microprocessor involves mapping the stack memory area to specific address in the memory.
- The 8085 microprocessors, for example, has a built-in stack pointer (SP) that keeps track of current location of the top of the stack.
- When subroutine is called or an interrupt is triggered, the SP is decremented and the return address or register values are stored at the new location. When subroutine finished or the interrupt is handled, the SP is incremented and the stored data is retrieved.
- The Stack is a critical components of microprocessors functionality, and its proper operation is essential for the correct execution of subroutines and interrupt handling.
- The size of stack and memory area it occupies must be carefully managed to ensure that it does not overflow, leading to unpredictable results or system crashes.

(2) Explain Arithmetic instruction of 8085.

- Arithmetic instruction in 8085 microprocessors is used for performing mathematical operations such as addition, subtraction, increment, decrement and comparison.
- These instructions are essential for performing complex calculation in a microprocessor-based system.

- **<u>ADD:</u>** The ADD instruction is used to add the content of a register or memory location to the accumulator.
  - **Syntax:** ADD destination
  - **Example:** ADD B
  - (This instruction adds the content of register B to the accumulator)

- **<u>ADC:</u>** The ADC instruction is used to add the content of register or memory location to the accumulator along with carry flag.
  - **Syntax:** ADC destination
  - **Example:** ADC C

  (This instruction adds the content of register C to the accumulator along with carry flag.)

- **<u>SUB:</u>** The SUB instruction is used to subtract the content of a register or memory location from the accumulator.
  - **Syntax:** SUB destination
  - **Example:** SUB D
  - (This instruction subtracts the content of register D from the accumulator)

- **<u>SBB:</u>** The SBB instruction is used to subtract the content of a register or memory location from the accumulator along with borrow flag.
  - **Syntax:** SBB destination
  - **Example:** SBB E
  - (This instruction subtracts the content of register E from the accumulator along with borrow flag.)

- **INR:** The INR instruction is used to increment the content of a register or memory location by 1.
  - **Syntax:** INR destination
  - **Example:** INR A
  - (This instruction increments the content of accumulator by 1)

- **DCR:** The DCR instruction is used to decrements the content of a register or memory location by 1.
  - **Syntax:** DCR destination
  - **Example:** DCR H
  - (This instruction decrements the content of register H by 1)

(3) Write an 8085 program to copy block of ten numbers starting from location 2050h to locations starting from 3050h.

➢

```
      LXI B,2050H;      Source pointer at 2050H
      LXI D,3050H;      Destination pointer at 3050H
      MVI L, 0AH;       Count of 10
BACK: LDAX B;           A gets source data
      STAX D;           A stored at destination
      INX B;            Increment source pointer
      INX D;            Increment destination pointer
      DCR L;            Decrement count register
      JNZ BACK;         Lop if count is not zero
      HLT;              end of the program
```

Q. 4

(1) State difference between PUSH and POP.

- The difference between the PUSH and POP instructions in a microprocessor and interfacing lies in their operations. The PUSH instruction stores a data value onto the top of the stack, while the POP instruction retrieves the topmost data value from the stack and removes it from the stack.

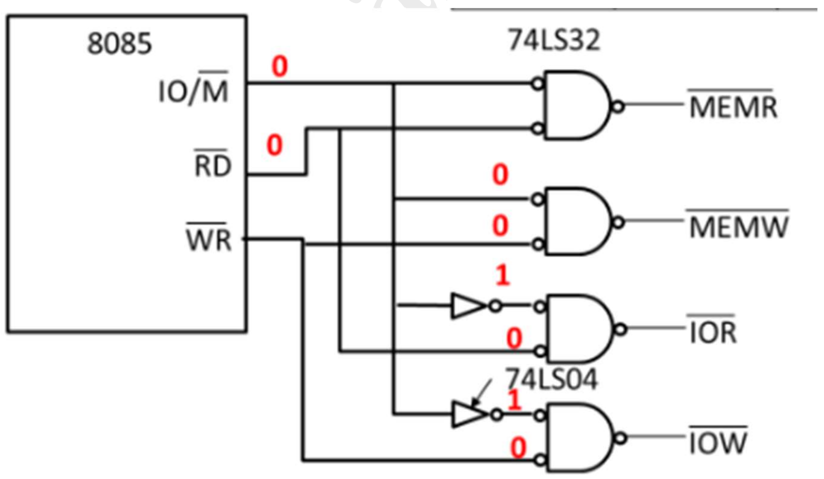
- The PUSH instruction pushes the contents of a register onto the stack. This is done by decrementing the stack pointer (SP), storing the contents of the register at the new location pointed to by SP, and then updating the value of SP. For example, the instruction "PUSH B" would store the contents of register B onto the top of the stack.

- POP instruction pops the topmost data value from the stack into a register. This is done by retrieving the contents of the location pointed to by SP, incrementing the stack pointer, and then storing the retrieved value in the specified register. For example, the instruction "POP B" would retrieve the topmost value from the stack and store it in register B.

- PUSH instruction stores a value onto the stack, while the POP instruction retrieves and removes a value from the stack. These instructions are used to store and retrieve values for subroutines, interrupt routines, and other purposes that require the use of a stack.

(2) Explain the generation of control signal in 8085.

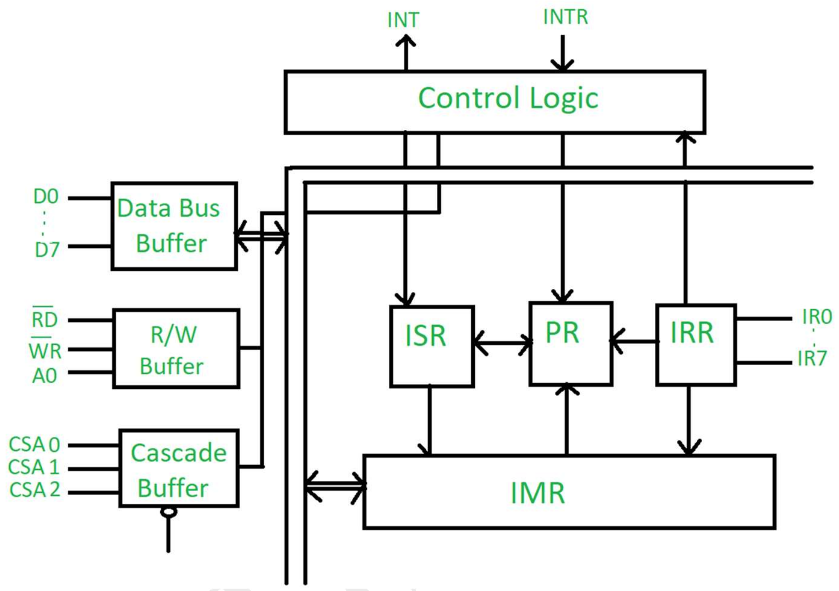
- The control signals in the 8085 microprocessor are signals that control flow of data and the operation of the microprocessor. The control signals are generated by the control unit, which is a part of microprocessors internal architecture.
- The control unit generates the control signal based on the instruction that is being executed by the microprocessor.
- It decodes the instruction and generates the necessary control signals to perform the required operation. Example, If the instruction is a memory read operation, the control unit will generate control signals to read data from the memory and store it in the microprocessors internal register.
- It includes signals such as memory read, memory write, I/O read, I/O write and interrupt acknowledge. These signals are used to control the flow of data between the microprocessor and the external device such as memory devices.
- The control signals in 8085 microprocessors are essential for the operation of the microprocessor, as they determine the flow of data and control the operation of microprocessor.
-



- $\overline{MEMR}$: To read data from memory.
- $\overline{MEMW}$: To write data in memory.
- $\overline{IoR}$: To read data from I/O devices.
- $\overline{IOW}$: To write data in I/O devices.

(3) Draw the internal block diagram of 82592A and explain the functions of each block in detail.

- The 8259A is a Programmable Interrupt Controller (PIC) that is commonly used in microcomputer systems to manage and control interrupt requests from peripheral devices. It was widely used in early personal computers such as the IBM PC. The 8259A is designed to work with the 8085, 8086 and other similar microprocessors.



- **Data Bus Buffer:** This block is used to communicate between 8259 and 8085/8086 by acting as buffer. It takes the control word from 8085/8086 and send it to the 8259.

- **R/W Buffer:** This block works when the value of pin CS is 0. This block is used to flow the data depending upon the inputs of RD and WR.

- **Control Logic:** This block generates the control signals that are used to manage the interrupt request and response. It has pin called INTR. This is connected to other microprocessors for taking the interrupt request. The INT pin is used to give output.

- **Cascade Buffer:** To increase number of interrupt pin, we can cascade more number of pins, by using cascade buffer. When we are going to increase the interrupt capability, CSA lines are used to control multiple interrupts.
- **Interrupt Status Register (ISR):** This register contains the status of each interrupt request. It is used to determine which IR input line has generated the interrupt request.
- **Interrupt Request Register (IRR):** It store all interrupt level that are requesting for the interrupt service.
- **Priority Resolver (PR):** This block determines the priority of the interrupt requests and ensure that the highest priority interrupt request is serviced first.
- **Interrupt Mask Register (IMR):** This register is used to make or enable individual interrupt request. The IMR can be used to temporarily disable an interrupt request, For example, during a critical section of code execution.

Q. 4

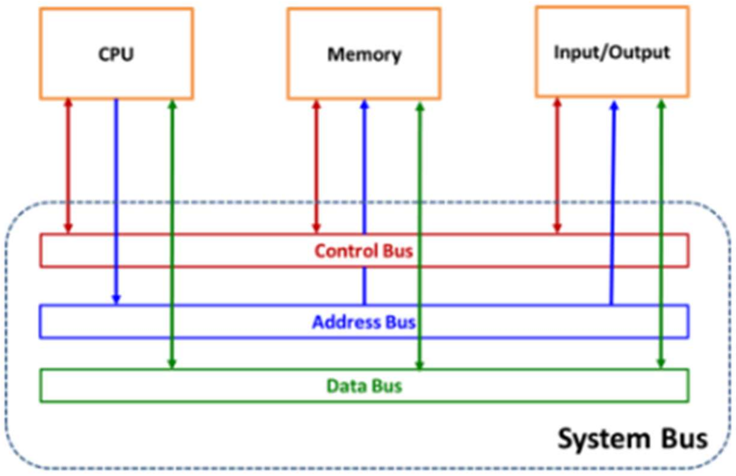
(1) Explain Machine level and Assembly level language.

➢

Machine Language	Assembly Language
It is only understand by the computers.	It is only understand by human beings not by computers.
In this data only represented with the help of binary, hexadecimal and octa decimal.	Data can be represented with the help of mnemonics such as Mov, Add, Sub, End, etc.
It is very difficult to understand by the human beings.	It is easy to understand by the human being as compare to machine language.
Modifications and error fixing cannot be done in this.	Modification and error fixing can be done in this.
It is very difficult to memorize so it is not possible to learn.	It is easy to memorize because of some alphabets and mnemonics are used.
Execution is fast.	Execution is slow.
No need of translator.	Need of translator to convert mnemonics in to machine language.
It is hardware dependent.	It is machine dependent and not portable.
Ex. 0000 1100	Ex. ADD, SUB

(2) Explain 8085 bus organization.

- The 8085 microprocessor has a well-defined bus organization, which consists of three major components: Data, Address and Control bus.
- The bus organization is a critical aspect od the microprocessor as it is responsible for communication between the microprocessor and external memory and I/O devices.



- **Data Bus:**
  - It is a group od conducting wires which carries data only.
  - Data bus is bidirectional because data flow in both directions, from microprocessor to memory or I/O devices and from memory or I/O devices to microprocessor.
  - Length of this is 8-bit, ranging from 00H to FFH.
  - When it is write operation, the processor will put the data on the bus, when it is read operation, the memory controller will get the data from specific memory block and put it into data bus.

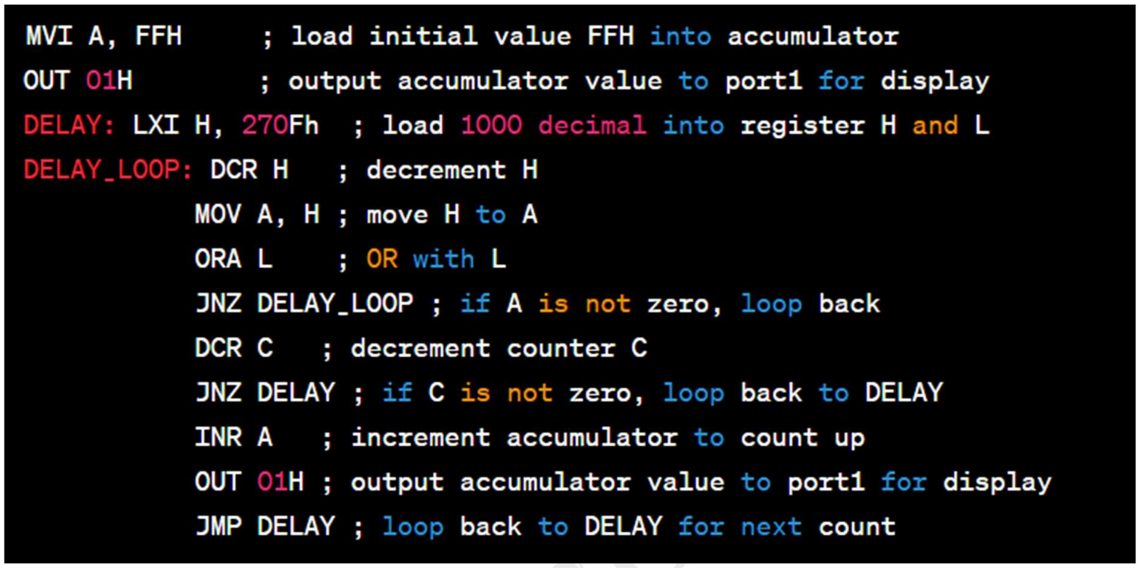
- **Address Bus:**

  - It is group of conducting wires which carries address only.
  - Address bus is unidirectional because data flow in one direction, from microprocessor to memory or from microprocessor to I/O devices.
  - Length of address bus is 16-bit. Ranging from 0000H to FFFFH.
  - It can transfer maximum 16-bit address which means it can address 65, 536 different memory location.

- **Control Signal:**

  - It is a group of conducting wires, which is used to generate timing and control all the associate peripherals, microprocessor uses control bus to process data.
  - The control bus is a group of signals that are used to control the flow of data between the microprocessor and external memory and I/O devices.
  - The control bus signals include read and write signals, chip select signals, and status signals.
  - The read and write signals are used to control the direction of data transfer on the data bus.
  - The chip select signals are used to select the external device that the microprocessor is communicating with.
  - The status signals are used to indicate the status of the microprocessor or external device.

(3) Write a program to count continuously in hexadecimal from FFH to 00H in a system with a clock period of 0.5 µs. Use register C to set up 1 millisecond delay between each count and display the number at the output port1.



```
MVI A, FFH      ; load initial value FFH into accumulator
OUT 01H         ; output accumulator value to port1 for display
DELAY: LXI H, 270Fh  ; load 1000 decimal into register H and L
DELAY_LOOP: DCR H    ; decrement H
            MOV A, H ; move H to A
            ORA L    ; OR with L
            JNZ DELAY_LOOP ; if A is not zero, loop back
            DCR C    ; decrement counter C
            JNZ DELAY ; if C is not zero, loop back to DELAY
            INR A    ; increment accumulator to count up
            OUT 01H ; output accumulator value to port1 for display
            JMP DELAY ; loop back to DELAY for next count
```

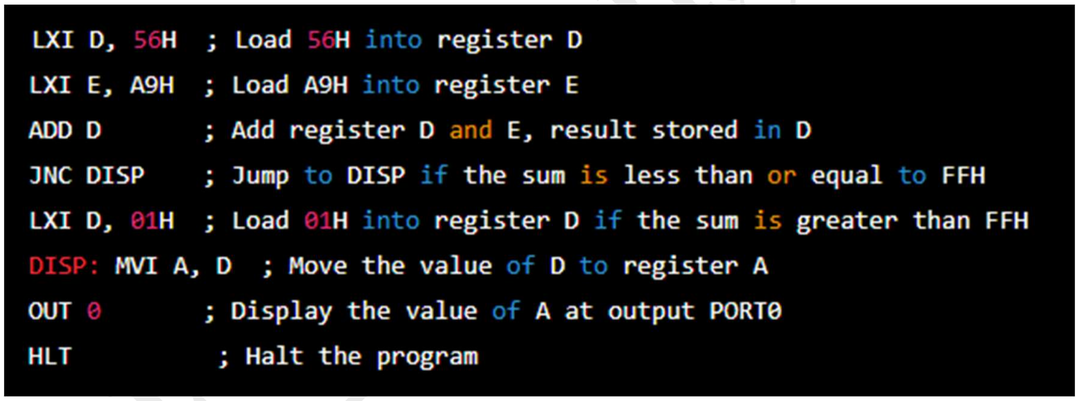
- The above program uses register C as a counter to generate a 1 millisecond delay between each count. The delay is implemented using a loop that decrements the value in register H and checks if it has reached zero. If it has not, the loop continues. Once the loop completes, the program increments the accumulator to count up and outputs the new value to port1 for display. Finally, the program jumps back to the delay loop for the next count.
- Note: The above program assumes that the output port1 is connected to a display device that can display hexadecimal values. If port1 is connected to a different type of device, the program may need to be modified accordingly.

Q. 5

(1) How many memory locations can be addressed by microprocessor with 14 address lines? Also specify how many address lines are required for 2KB memory.

- A microprocessor with 14 address lines can address ***2^14 = 16384*** memory locations.
- To address 2KB of memory, we would need 11 address lines, as ***2^11 = 2048.***

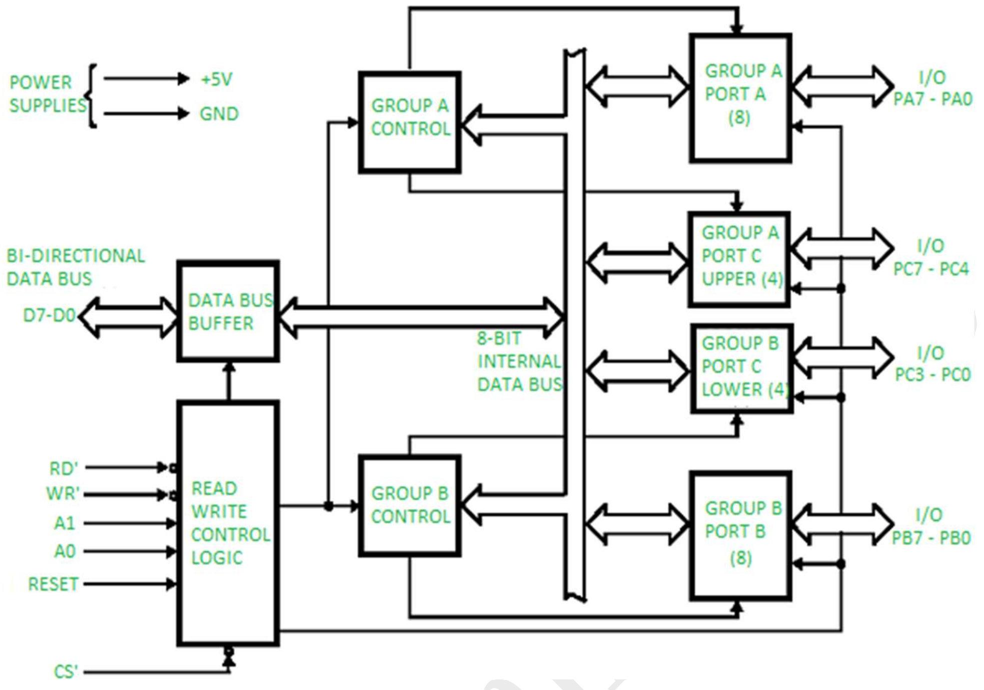
(2) Load the hexadecimal numbers 56H and A9H in registers D and E respectively and add them. If sum is greater than FFH, display 01H at output PORT0; otherwise display sum.

- 

```
LXI D, 56H  ; Load 56H into register D
LXI E, A9H  ; Load A9H into register E
ADD D       ; Add register D and E, result stored in D
JNC DISP    ; Jump to DISP if the sum is less than or equal to FFH
LXI D, 01H  ; Load 01H into register D if the sum is greater than FFH
DISP: MVI A, D  ; Move the value of D to register A
OUT 0       ; Display the value of A at output PORT0
HLT          ; Halt the program
```

- In this program, the hexadecimal numbers 56H and A9H are loaded into registers D and E using the LXI instruction.
- The ADD instruction is used to add the two values stored in D and E, and the result is stored in D.
- The JNC instruction checks if the carry flag is reset, indicating that the sum is less than or equal to FFH.
- If the carry flag is reset, the program jumps to DISP, where the value stored in D is moved to register A using the MVI instruction.
- The value of A is then displayed at output PORT0 using the OUT instruction.
- Finally, the program is halted using the HLT instruction.

(3) Draw and explain the internal block Diagram of 8255A.



➢ Data Bus Buffer:

- The tri-state bi-directional buffer is used to interface the internal data bus of 8255 to the system data bus.
- Input or Output instructions executed by the CPI either Read data from, or write data into the buffer.
- Output data from the CPU to the ports or control register, and input data to the CPU from the ports or status register are all passed through the buffer.

➢ Control Logic:

- The control logic block accepts control bus signals as well as inputs from the address bus, and issues commands to the individual group control block (Group A control and Group B control).
- It issues appropriate enabling signals to access the requires data/control words or status words. The input pins for the control logic section are described here.

➢ **Group A and Group B Controls:**

- Each of Group A and B control blocks receives control words from the CPU and issues appropriate commands to the ports associated with it.
- Group A control block controls Port A and PC_7-PC_4 while Group B control block control port B and PC_3-PC_0.
- **Port A:** This has an 8-bit latched and buffered output and an 8-bit input latch. It can be programmed in three modes: mode0, mode1, mode2.
- **Port B:** This has an 8-bit data I/O /latch/buffer and 8-bit data input buffer. It can be programmed in mode0 and mode 1.
- **Port C:** This has one 8-bit unlatched input buffer and an 8-bit output latch/buffer. Port C can be separated into two parts and each can be used as control signals for Port A and Port B in the handshake mode. It can be programmed for bit set/reset operation.

Q. 5

(1) Explain the given pins of 8086.

1.ALE 2.DEN 3.MN/MX

- **ALE:**
  - Stands for Address Latch Enable. This signal is provided by 8086 to demultiplex the AD_0-AD_{12} into A_0-A_{15} and D_0-D_{15} using external latches.
  - It is and active high (1) pulse during T1 of any bus cycle. ALE signal is never floated, is always integer.

- **DEN:**
  - This signal informs transceivers that the CPU is ready to send or receive data.
  - DEN is active LOW during each memory and IO access.
  - It will be low beginning with T2 until the middle of T4, while for write cycle, it is active form the beginning of T2 until the middle of T4.

- **MN/MX:**
  - This pin indicates what mode the processor is to operate in.
  - In minimum mode, the 8086 itself generates all bus control signals.
  - In maximum mode the three status signals are to be decoded to generate all the bus control signals.
  - Minimum Mode Pins The following 8 pins function descriptions are for the 8086 in minimum mode; MN/ MX = 1.
  - The corresponding 8 pins function descriptions for maximum mode is explained later.

(2) Explain the modes of operation of 8086 microprocessor.

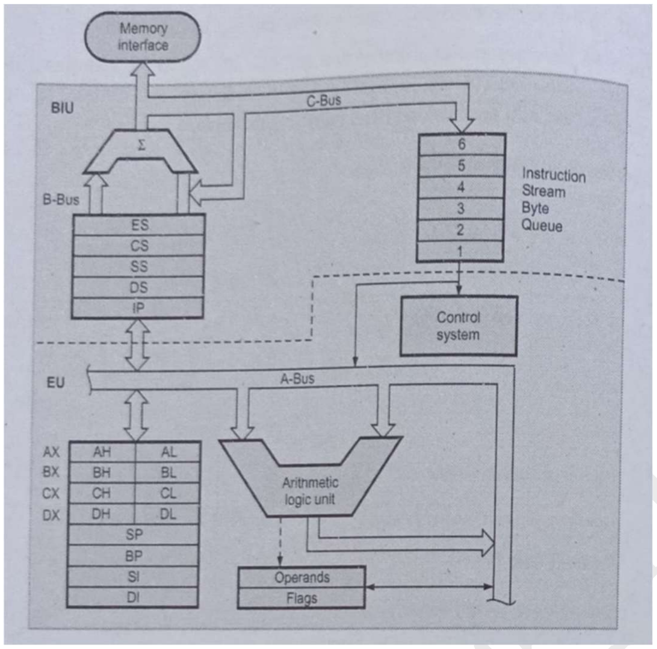
➢ The 8086 microprocessor can operate in two modes: minimum mode and maximum mode.

➢ **Minimum Mode:**

- In minimum mode, the 8086 processor is interfaced with a single memory and a single I/O port.
- In this mode, the processor is responsible for generating control signals for memory and I/O devices.
- The minimum mode is used in small systems that do not require a large amount of memory and I/O devices.

➢ **Maximum Mode:**

- In maximum mode, the 8086 processor can handle up to three memory devices and up to eight I/O devices.
- In this mode, the processor is responsible for generating control signals for memory and I/O devices, but it also communicates with a bus controller for coordinating access to memory and I/O devices.
- The maximum mode is used in larger systems that require a greater amount of memory and I/O devices.



(3) Explain logic block diagram of 8086 Microprocessor.

➢ **Bus Interface Unit (BIU):** The BIU is responsible for the external bus interface of the microprocessor. It contains the instruction queue, segment register, and address generation logic. It fetches instructions from memory and places them into the instruction queue. The BIU generates 20-bit physical addresses and sends them to the memory or I/O devices.

Execution Unit (EU): The EU is responsible for the execution of instructions. It contains the arithmetic and logic unit (ALU), general registers, flags register, and instruction pointer register. The EU fetches instructions from the instruction queue and executes them. It performs arithmetic and logical operations on the data stored in the general registers.

➢ **Segment Registers:** The 8086 microprocessor has four segment registers - CS, DS, SS, and ES. These registers are used to store the starting addresses of different segments in memory. The CS register holds the starting address of the code segment, the DS register holds the starting address of the data segment, the SS register holds the starting address of the stack segment, and the ES register holds the starting address of the extra segment.

- **Instruction Queue:** The instruction queue is a buffer that stores up to six bytes of instructions fetched from memory by the BIU. It allows the microprocessor to fetch instructions from memory more efficiently.

- **Address Generation Unit:** The address generation unit is responsible for generating the physical memory addresses from the segment addresses and offsets. It adds the segment and offset values to generate a 20-bit physical address.

- **Arithmetic and Logic Unit (ALU):** The ALU is responsible for performing arithmetic and logical operations on the data stored in the general registers. It can perform operations such as addition, subtraction, multiplication, division, bitwise AND, OR, XOR, and complement.

- **General Registers:** The 8086 microprocessor has eight 16-bit general-purpose registers - AX, BX, CX, DX, SI, DI, BP, and SP. These registers can be used to store data, addresses, and operands for arithmetic and logical operations.

- **Flags Register:** The flags register is a 16-bit register that contains status flags indicating the result of the most recent arithmetic or logic operation performed by the microprocessor. The flags register can be used to implement conditional instructions.

- **Instruction Pointer (IP):** The instruction pointer is a 16-bit register those points to the memory location of the next instruction to be executed. It is automatically updated after each instruction execution.