Q.1) SOLVE ANY FIVE : (20 MARKS)

a) Describe the function of SID and SOD pins of 8085 microprocessor

Ans: - **SID: -** **(2 Mark)**

Serial Input Data – SID pin is used to receive data serially from external devices to the accumulator.

- At a time only one bit is received.
- LSB of 8-bit data is received first.

SOD: - **(2 Mark)**

Serial Output Data – SOD pin is used to transmit data serially from accumulator to the external devices connected to the pin.

- At a time only one bit is transmitted.
- LSB of data is transmitted first.

b) List eight features of 8086.

Ans: - **(½ mark for each one)**

- 20 bit address lines so 2^{20} = 1Mbyte of memory can be addressed.
- Operating clock frequencies 5MHz, 8MHz, 10MHz.
- Arithmetic operation can be performed on 8-bit or 16-bit signed & unsigned data.
- Provide 20 address lines so, 1 Mbytes of memory can be addressed.
- The instruction set is powerful, flexible and can be programmed in high level language like C language.
- Can operate in single processor and multiprocessor configuration i.e. operating modes.
- Provides 6-bytes instruction queue for pipelining of instructions executions.
- Provides 256 types of vectored software interrupts.
- Operate in maximum and minimum mode to achieve high performance level.
- Provides separate instructions for string manipulation.

c) Describe segmented memory and list its four advantages.

Ans: - Description of segmentation **(2 Mark)**

The memory in 8086 based system is organized as segmented memory. 8086 can access 1Mbyte memory which is divided into number of logical segments. Each segment is 64kb in size and addressed by one of the segment register. The 4 segment register in B10 hold the 16-bit starting address of 4 segments. CS holds program instruction code. Stack segment stores interrupt & subroutine address. Date segment stores data for program. Extra segment is used for shared data.

Advantages of segmentation **(Any 4)** **(1/2 Mark each)**

1) With the use of segmentation the instruction and data is never overlapped.
2) The major advantage of segmentation is Dynamic Relocability of program which means that a program can easily be transferred from one code memory segment to another code memory segment without changing the effective address.
3) Segmentation can be used in multi-user time shared system.
4) Segmentation allows two processes to share data.
5) Segmentation allows you to extend the addressability of a processor.
6) Programs and data can be stored separately from each other in segmentation.

d) Describe the immediate addressing modes in 8086 with two examples.

Ans: - Description of Immediate addressing mode: - **(2 Mark)**

(Any 2 examples – 1 mark each example)

operand in the instruction is 8-bit or 16-bit data which is to be moved/copied in the destination specified in the instruction. Since data to be moved is part of an instruction, processor do not take any time to search for it.

- MOV AL, 25H – Move 25H to AL or AL = 25H
- MOV CX, 1234H – Move 16-bit data to CX or CX = 1234H. MOV[BX], 23H – Move 23H to memory location which offset or effective address is in BX and base or starting address is in DS .
- MOV [SI], 0B29H – Move 0B29H (16-bit data) to two consecutive memory locations which effective address or offset is in SI and base starting address is in DS.

e) Which instruction are used for stack access? Illustrate use of any one with example.

Ans: - Instruction use for stack access: - **(2 Mark)**

(PUSH, POP, CALL, RET – ½ Mark each)

(Any 1 example-(2 Mark))

- PUSH BX: Decrement SP by 2,copy contents of BX to stack memory locations which offset is in SP and base address is in SS.

OR

- SS: [SP] – Higher byte of source i.e. BH.

SS [SP-1] – Lower byte of source i.e. BL.

- POP BX: Increment SP by 2, copy stack memory contents back to BX which is in SP and base address is in SS.

OR

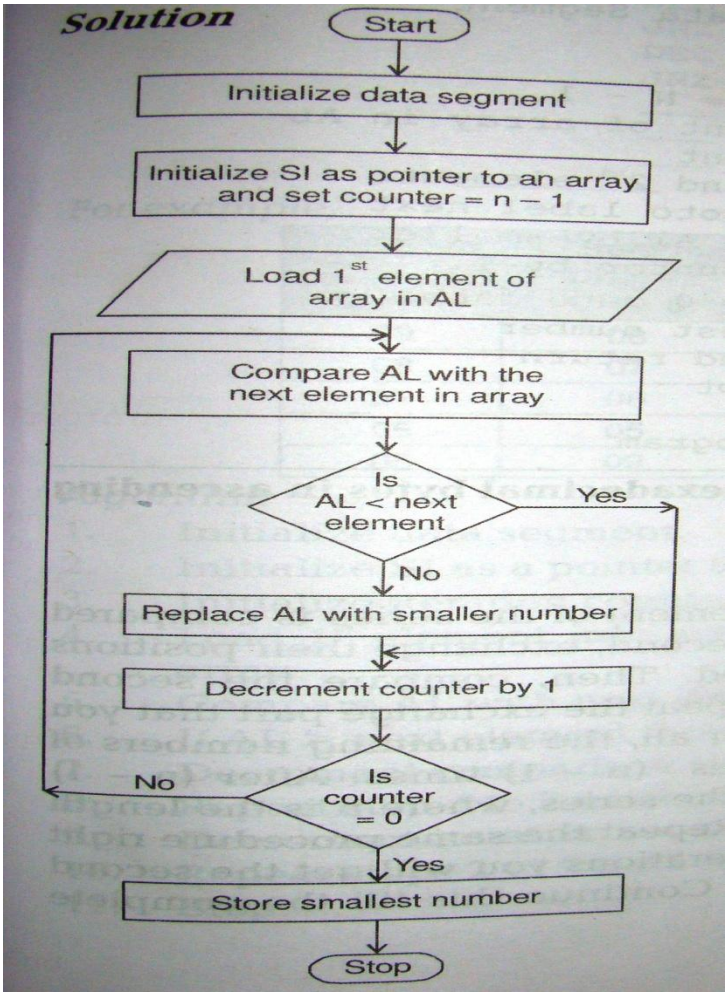
- SS: [SP] – Higher byte of destination i.e. BL.

  SS [SP+1] – Lower byte of destination i.e. BH.

- CALL Procedure_ name: Content of Instruction pointer gets stored on stack memory.
- RET: Content of stack memory gets back to instruction pointer.

f) Draw flowchart and write an ALP to find smallest numbers of given block of data.

Ans: - **Flowchart: -** **(2 mark)**



Program: - **(2 mark)**

Program

```
Data Segment
Array db 25H, 79H, 12H, B2H, 3DH    ; given array
smallest db ?                       ; store smallest element
Data Ends

Code Segment
Assume CS: code, DS: Data
Start: MOV  AX,  Data               ; Initialize Data Segment
       MOV  DS, AX
       LEA  SI, Array               ; Set Pointer
       MOV  CL,   04H               ; Set Counter = N - 1
       MOV  AL, [SI]                ; Get 1st Element of array in AL
UP:    INC  SI                      ; Get 2nd Element
       CMP  AL, [SI]                ; Compare 1st and 2nd elements
       JC   Next                    ; If 1st < 2nd goto label next
       MOV  AL, [SI]                ; Else replace Al by smaller
Next:  DEC  CL                      ; Decrement counter by 1
       JNZ  Up                      ; If counter ≠ 0 goto label up
       MOV  Smallest, AL            ; Store smallest number
       MOV  AH, 4CH                 ; Terminate and return
       INT  21H                     ; To Dos prompt
Code   Ends
End    Start                        ; End of a program
```

g) Describe following directives:

Ans: - **(Description 1 mark, Example 1 mark)**

i) **DD: -** **(2 mar**k)

Define Double word (32-bits)

- It is used to declare a variable of type 32-bits.
- It reserves memory locations which can be accessed as type double word.
  e.g. NUMBER DD 12345678H – allocated 16 memory locations.

ii) **DW: -** **(2 mark)**

Define Word (16-bits)

It is used to tell the assembler to define a variable of type word in memory or to reserve storage locations of type word (16) in memory.

e.g. BLOCK DW 1234H, 3456H, 5678H : Declare array of 3 words.

STORAGE DW 10DUP (2222H): Reserve array of 10 nos. with no. 2222H.

Q.2) Solve any two: (16 marks)

a) Describe the function of following blocks of 8085:
i) General purpose register
ii)ALU
iii) Timing and control unit
iv) Instruction decoder and m/c cycle encoding

Ans: - (2 Mark each function)

1) **General purpose register: -** The microprocessor 8085 has six 8-bit general purpose registers as B, C, D,E, H, and L. These registers can be used to store any 8-bit user data. For storing a 16-bit number, we have to make pairs of two 8-bit registers like, BC, DE, and HL.
2) **ALU: -** ALU stands for arithmetic and logical. The ALU of microprocessor 8085 is 8-bit microprocessor. The ALU is responsible to perform all arithmetic and logical operation like addition, subtraction, comparison, ANDING, etc.
3) **Timing and control unit: -** The timing and control unit accepts information from the instruction decoder and generates different control signal. This unit synchronizes all the microprocessor operation and generates control and status signal necessary for communication between the microprocessor and peripherals.
4) **Instruction decoder & m/c cycle encoder: -** It accepts an op. code of the instruction from the instruction register decode it and give information to control logic. The information includes what operation is to be performed who is going to perform, how many operand bytes the instruction has, etc.

b) List significant differences between minimum and maximum mode operation of 8086. Describe under what situation maximum mode operation is useful.

Ans: - Difference **(Any 6 points – 6 mark)**

Sr. No.	**Minimum mode**	**Maximum mode**
1.	MN/MX pin is connected to V_{CC} i.e. MN/MX = 1.	MN/MX pin is connected to ground i.e. MN/MX = 0.
2.	Control system M/O, RD, WR is	Control system M/IO, RD, WR is not

	available on 8086 directly.	available directly in 8086.
3.	Single processor in the minimum mode system.	Multiprocessor configuration in maximum mode system.
4.	In this mode, no separate bus controller is required.	Separate bus controller (8288) is required in maximum mode.
5.	Control signals such as IOR, IOW, MEMW, MEMR can be generated using control signals M/IO, RD, WR which are available on 8086 directly.	Control signals such as MRDC, MWTC, AMWC, IORC, IOWC and AIOWC are generated by bus controller 8288.
6.	ALE, DEN, DT/R and INTA signals are directly available.	ALE, DEN, DT/R and INTA signals are not directly available and are generated by bus controller 8288.
7.	HOLD and HLDA signals are available to interface another master in system such as DMA controller.	RQ/GT_0 and RQ/GT_1 signals are available to interface another master in system such as DMA controller and coprocessor 8087.
8.	Status of the instruction queue is not available.	Status of the instruction queue is available on pins QS_0 and QS_1 .

Situation in which maximum mode is useful **(2 Mark)**

- Maximum mode is basically nothing but a multiprocessor system. When microprocessor 8086 based system contains external math co-processor like 8087.
- In this case 8086 also needs a bus controller 8288 to generate all control signal in order to transfer the data from or to memory or I/O devices. Also it supports the signal exchange between math co-processor & 8086 in order to handover the charge of data & address bus.
- In this situation maximum mode and specific pins of 8086 are used to establish the communication between 8086 and co-processor.

c) Describe pipeline architecture concept and how it helps in improving system throughput. Explain the function of stack and queue.

Ans: - Stack: - **(2 Mark)**

- Stack is a section of memory set aside to store addresses and data while a subprogram is executed.
- 8086 allows to set entire 64kb segment as a stack.

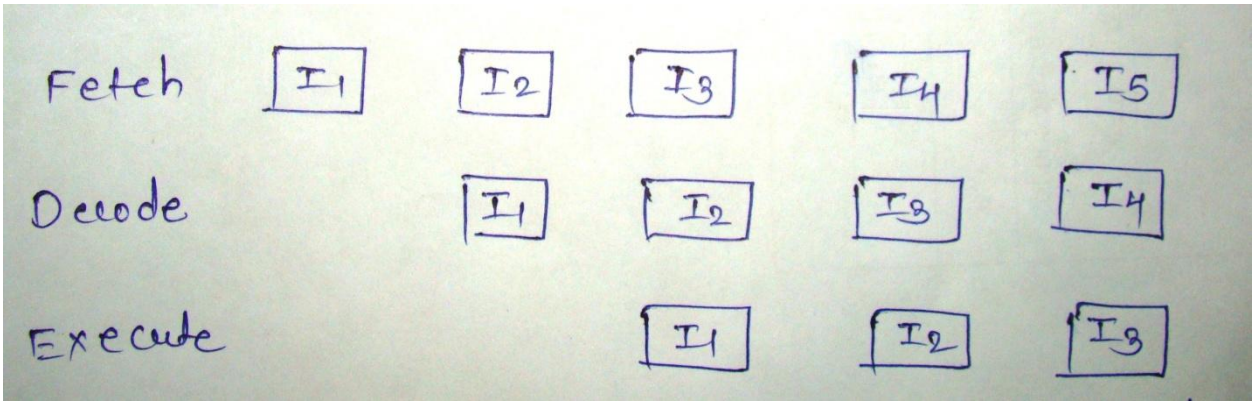
- Base or starting address of stack will be in SS & offset address will be in stack pointer.

Queue: - **(2 Mark)**

- 8086 has a 6-byte instruction stream in Bus Interface Unit.
- As per the information given by EU, BIU fetches the opcodes of instructions & data from memory & stores them in instruction queue.
- Execution unit EU executes the instructions one by one & BIU keeps fetching & filling the queue, so queue will never be empty.
- This increases speed of execution.

Pipelining: - **(4 Mark)**

- In a pipelined processor, fetch, decode and execute operation are performed simultaneously or in parallel.
- When 1st instruction is being decoded, same time next instruction's code is fetches. When 1st instruction is getting executed, 2nd one's is decoded, and 3rd instruction code is fetches from memory. This process is known as pipelining.



- Pipelining result in a tremendous performance gain which improves speed of operation to great extent.
- Since 8086 has 6-byte instruction queue, BIU takes care that queue will not remain empty & EU keeps decoding & executing the instruction bytes.

Q.3 Solve any two

a) List addressing modes of 8086 and explain any two in brief

Ans:- (2 marks for listing and 3 marks each for explaining any two modes in detail)

Immediate addressing mode :

- In this mode, the immediate data is the part of the instruction and appear in the form of successive byte or bytes.
- So immediate data may be 8 bit [byte] or 16 bit [word] in length.
- Immediate data can be accessed quickly as they are available in an instruction queue, hence no extra bus cycle is required to read data.

Examples :

MOV AL, 46H	AL is loaded with 8 bit immediate data 46H.
MOV BX, 1234H	BX is loaded with 16 bit immediate data 1234H.

2. **Direct addressing mode :**

- In this mode, a 16-bits memory address (offset) of operand is directly specified in the instruction as a part of it.
- The offset of displacement may be either 8 bit or 16 bit which follows the instruction op-code.
- So, the physical address is calculated by adding this offset to the base segment registers i.e. CS, DS, ES, SS.

Examples :

MOV AL, [3000H]	AL will be loaded with the content of memory location whose offset is 3000H from base address.
AND AX, [8000H]	AX will be ANDed with the content of memory location whose offset is 8000H from the base.

- In above examples, DS is the default base address register.
- Suppose, data is stored in EXTRA segment, then data can loaded in register by specifying address using following way.

MOV AX, ES:[4000H]

3. **Register addressing mode :**

- In this mode, the data is stored in a registers and it is referred using the particular register i.e. all register except IP may be used in this mode.
- Register may be source operands, destination operand or both.
- The instruction of this addressing mode are compact and faster in execution as all registers are reside in chip and no external bus is required to read data.
- Registers may be 8 bit or 16 bit.

Examples :

MOV AX, CX	Copies the content of CX reg. to AX reg.
AND AL, BL	ANDing the content of BL with AL, store result in AL.
ROR AL, CL	Rotate the contents of AL CL times.

4. **Register indirect addressing mode :**

- In this mode, the address of the memory location that contains data or operand is determined in an indirect way, using offset register such as BX, SI, DI register.
- The default segment register is either DS or ES.
- If BP is used, then SS is the default segment register.

Examples :

MOV AX, [BX]	Copies the contents of memory location whose offset is in BX register.
SUB [SI], AL	Subtract the content of AL from the memory location whose offset is in SI register and store result in same memory location.

Indexed addressing modes :

- In this mode, the offset of the operand is stored in one of the index register.
- DS and ES are the default segments for index register SI and DI respectively.
- This mode allows the use of a signed displacement.

Examples :

MOV BL, [SI]	Copies the byte from memory location whose offset is in index register SI to AL.
ADD AX, [DI+8]	Copies the word from memory location whose offset will be calculated by adding 8 with the content of DI register.

5. **Register relative addressing mode :**

- In this mode, the data is available at an effective address formed by adding 8-bits or 16-bits displacement with content of any one of the registers such as BX, BP, SI and DI in the default DS and ES segment.

Examples :

MOV AX, 50[BX]	Copies the word from memory location whose offset will be calculated by adding the 50 with the content of BX register.
ADD AX, 5000[BP]	Copies the word from memory location whose offset will be calculated by adding 5000H to the content of BP register.

6. **Base indexed addressing mode :**

- In this mode, the effective address of data is formed by adding the content of a base register BX or BP to the content of an index register SI or DI with default segment DS and ES.

Examples :

MOV AX, [BX][SI]	Copies the word from memory location whose offset is calculated by adding the content of BX with the contents of SI.
ADD AL, [BX][DI]	Copies the byte from memory location whose offset is calculated by adding the content of BX with the contents of DI.

7. **Relative base indexed addressing mode :**

- In this mode, the effective address of data is calculated by adding the 8-bits or 16-bits displacement with the sum of base register BX or BP and SI or DI register in the default segment DS and ES.

Examples :

MOV AX, 60[BX][SI]	Copies word from memory location whose offset is calculated by adding the 60H with the contents of BX and SI i.e. [60+BX+SI]

b) With suitable example explain following instructions

Ans:- (Each Instruction with explaination 1 mark &relevant example carries 1 marks)

i) INC Reg

This instruction increments the contents of register specified in the instruction by 1. And the contents are stored in the register itself.

The syntax for this instruction is : **INC Reg**

Addressing mode: Register addressing mode.(optional)

Operation: reg = reg + 1

Example:

INC AX

ii) AAM

AAM [ASCII adjust after multiplication] :

- This instruction can be used to convert the result of the multiplication of two valid unpacked BCD numbers.
- This instruction should be issued after the multiplication instruction.

Operation :

(a) AL=AL MOD 10.

(b) AH=AL / 10 [Only integer part is considered].

Example :

AL = 06	BL = 08.
MUL BL	AX = 30 H (48 in decimal).
AAM	AH = 04 and AL = 08.

iii) XLAT

XLAT :

- The XLAT instruction replaces a byte in the AL register with a byte from a lookup table in memory.

Operation :

AL ← DS:[BX+AL]

Example :

```
.DATA
    TABLE DB   '0123456789ABCDEF'
    CODE  DB   11
.CODE
    .
    .
    .
    MOV BX, offset TABLE   Point BX to the start of
                           lookup table in DS
    MOV AL, CODE
    XLAT                   Replace code in AL with
                           code from lookup table.
                           The content of AL will
                           be 0BH
```

NOTE :- Examples are not mendetore

iv) XCHG

XCHG destination, source :

- This instruction exchanges the contents of a registe with the contents of another register or memor location.

Operation performed :

Destination ⇆ Source

Examples :

XCHG AX, BX	Exchange the word in AX with word in BX.
XCHG BL, CH	Exchange the byte in BL with byte in CH.
XCHG AX, [7000H]	Exchange the word in AX with memory i.e. AH with the content of 7000H memory location and AL with the content of 7001H memory location.

Note:- Any of the example can be considered

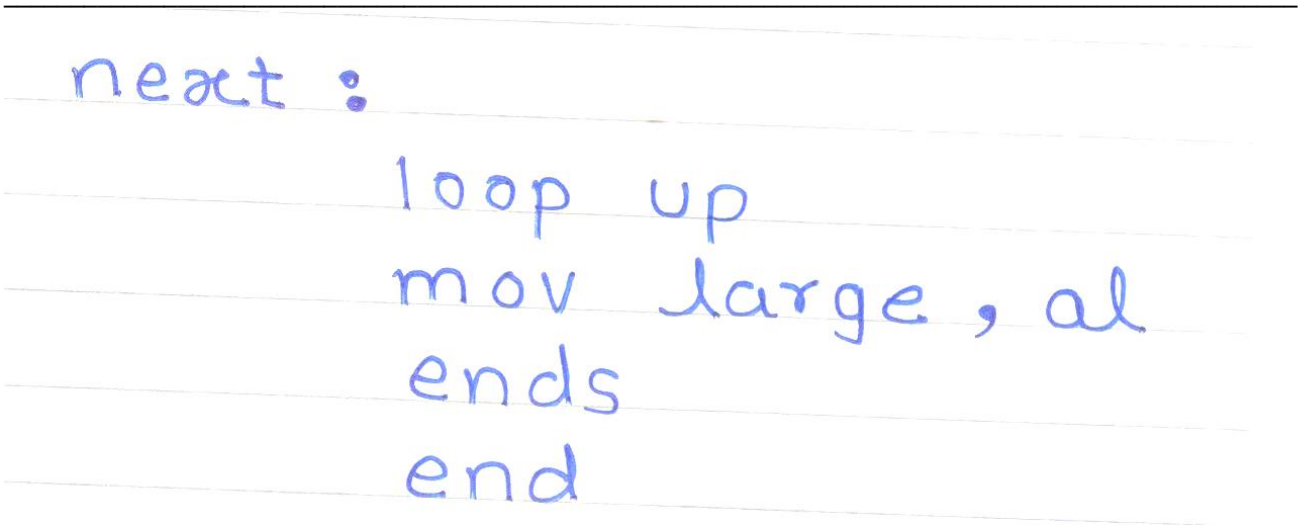
c) Write an ALP to find largest of ten numbers(correct program 8M)

Ans:- (The program for 16it numbers can be too considered)

```
Using 8-bit.

.model small
.data
    array  db  12h,31h,02h,45h,40h,51h,48h
               18h,20h,09h
    large  db  0
.code
    mov ax, @data
    mov ds, ax
    mov cx, 0A
    mov si, offset array

    mov al, [si]
    dec cx
up: inc si
    cmp al, [si]
    jnc next
    mov al, [si]
```



Q. 4) Solve any Two

a) Write an ALP to compare two strings of 50 words each

Ans:- Program **(correct program 08 marks)**

Assume CS: CODE, DS: DATA

CODE SEGMENT

MOV AX, @DATA

MOV DS, AX

MOV SI, OFFSET STR1

MOV DI, OFFSET STR2

MOV CL, 32H **(COUNT IS 50 CONVERTING TO HEX)**

CLD

REP: CMPBSW

HLT

CODE ENDS

DATASEGMENT

STR1 DW 0001H, 0002H, 0003H, 0004H…….

ORG 0100H

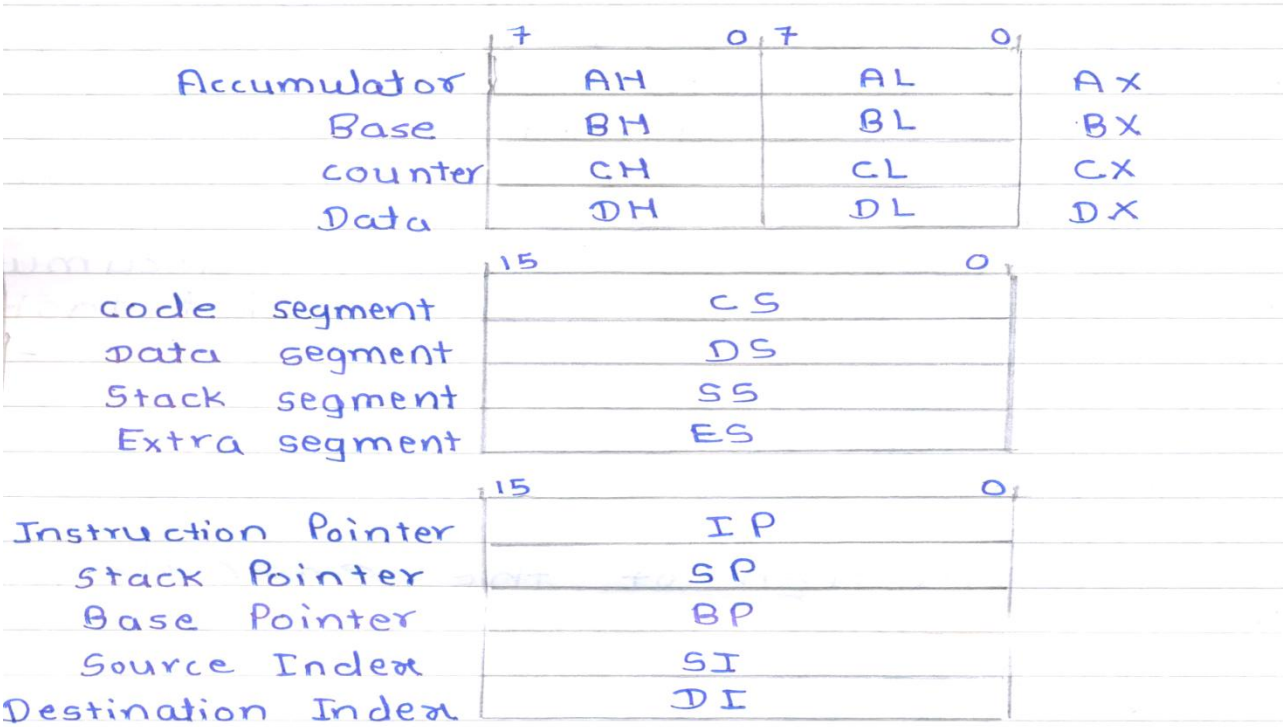
STR2 DW 0032H, 3654H, 5647H, 2145H……..

DATA ENDS

END

b) Explain programming model of 8086

Ans:- (diagram 4 marks, pointer register, index register, segment register, general purpose register : each 1 mark)



General Purpose register of 8086

EU has 8 general purpose registers AH, AL, BH, BL, CH, CL, DH & DL. These registers can be used as 8-bit registers individually or can be used as 16-bit in pair to have AX, BX, CX & DX

AL register is called as accumulator.

Some register have special task such as,

- CX is used as a counter
- BX can be used as a pointer
- DX is used for I/O addressing to hold the I/O address in some instruction of the 8086 µp.

The other register in EU are SP, BP, SI & DI.

- SP & BP are pointer register, which holds 16-bit offset within the particular segment.
- SI & DI are Index register
- The register SI used to stored the offset of source data or string in data segment while the register DI is used to store the offset of destination in data or extra segment.

* Segment Register

→ There are 4 segment register of 16-bit each.

→ The code segment CS register is used for addressing a memory location in the code segment of memory, where the executable program is stored.

→ The Data segment DS register points to the data segment of the memory, where the data is stored.

→ The Extra Segment ES register also refers to a segment. Thus extra segment also contain the data.

→ The Stack segment SS register is used for addressing stack segment of the memory. The stack segment is that segment of memory, which used to store stack data

* Instruction Pointer

→ The instruction pointer IP register hold 16-bit address of the next code byte within the code segment.

→ The value store in IP is called as offset or displacement.

c) Write an ALP to arrange no in the array in descending order using procedure

Ans:- **(correct program 8M**

```
.model small
.data
    array dw 12h, 11h, 21h, 9h, 19h
.code
    mov ax, @data
    mov ds, ax
    call dsc_order          ;call procedure to
                            arrange.
    mov ah, 4ch
    int 21h
dsc_order proc
    mov bx, 5
up1:
    lea si, array
    mov cx, 4
up:
    mov ax, [si]
    cmp al, [si+2]
    jnc dn
    xchg ax, [si+2]
    xchg ax, [si]
dn:
    add si, 2
    loop up
    dec bx
    jnz up1
    ret                     ;return to calling
                            program
endp
ends
end
```

Q.5) Solve any two **16**

a) Explain any four rotate instruction.

Ans:- (**Each instruction 2M [Syntax 1M,Description 1M])**

1) **ROL [Rotate left without carry]**

Syntax: ROL Destination, Count

This instruction rotates all of the bits of the specified byte or word count times towards left. The bit moves out of MSB is rotated around into LSB and copied to CF.

Flags Affected: OF, CF

Operation

Example: CF=0 BL=1011 1010

ROL BL, 1 ; Rotate all bits in BL left by one bit position.

CF=1 BL=0111 0101

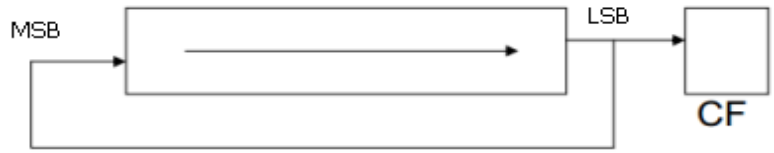
2) ROR [Rotate right without carry]

Syntax: ROR Destination, Count

This instruction rotates all of the bits of the specified byte or word count times towards right. The bit moves out of LSB is rotates around into the MSB and also copied to CF.

Flags Affected: OF, CF (optional)

Operation



Example CF=0 BL=0011 1011

ROL BL,1 ;Rotate all bits in BL right by one bit position.

CF=1 BL=1001 1101

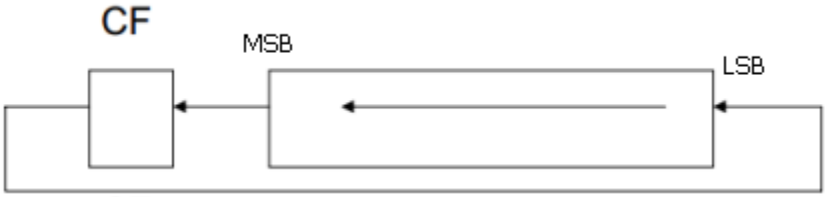
3) **RCL [Rotate left with carry]**

Syntax: RCL Destination, Count

This instruction is used to rotate all bits in a specified byte or word with carry by count times towards left. The operation is circular because the MSB of the operand is rotated into the carry flag and bit in the carry flag is rotated around into the LSB of the operand.

Flags Affected: OF, CF (optional)

Operation



Example. CF=1 BL=0011 1011

RCL BL, 1 ; Rotate all bits in BL left by one bit position.

CF=0 BL=0111 0111

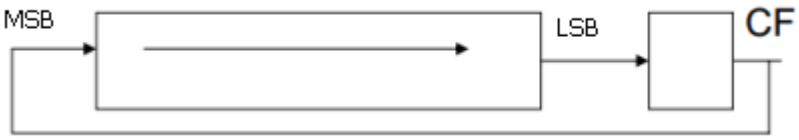
4) **RCR [Rotate right with carry]**

Syntax: RCR Destination, Count

This instruction is used to rotate all bits in a specified byte or word with carry by count times towards right. The operation is circular because the LSB of the operand is rotated into the carry flag and bit in the carry flag is rotated around into the MSB of the operand.

Flags Affected: OF, CF (optional)

Operation



Example. CF=0 BL=0011 1011

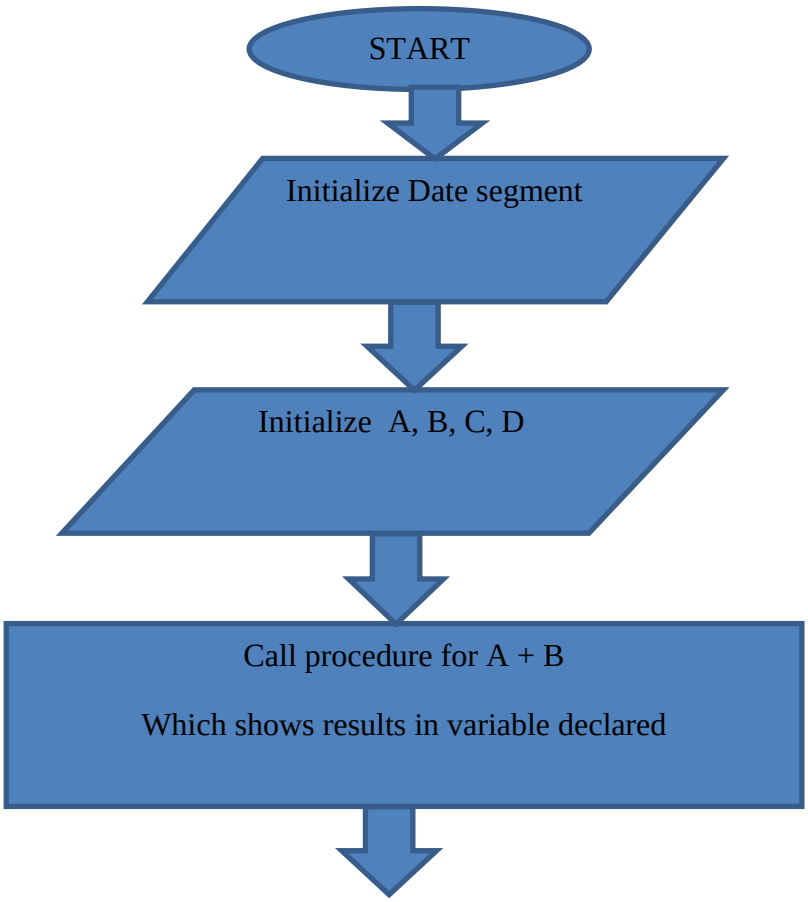
RCR BL, 1 ; Rotate all bits in BL right by one bit position.

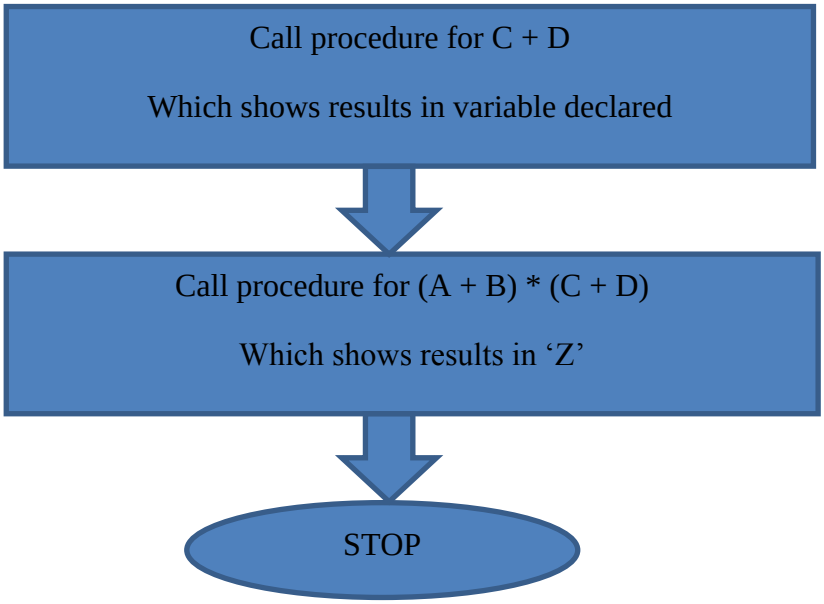
CF=1 BL=0001 1101

b) Write an ALP using procedure for addition for equation Z = (A+B) × (C+D). Draw flowchart and write result

Ans:- (Correct program 04 Marks, Flowchart 02 mark ,Result 02 marks[for any relevant input])

FlowChart


START
Initialize Date segment
Initialize A, B, C, D
Call procedure for A + B
Which shows results in variable declared



Source Code

```
data segment
        A    dw  0001h
        B    dw 0002h
        C    dw  0003h
        D    dw  0004h
       res_add1     dw   ?                    ; Result of A+B
        res_add2    dw    ?                   ; Result of C+D
        Z      dd     ?                 ; Result of (A+B) × (C+D)
data ends
code segment
Assume CS: code,DS: data
START : mov ax, data                         ;Initialize data segment
        : mov ds, ax
          call add_proc1          ; call procedure for A + B
          call add_proc2           ; call procedure for C + B
          call mul_num             ; call procedure for (A + B) × (C + D)
```

```
        Mov ah , 4ch        ; exit to DOS
        int 21h
add_proc1 Proc Near  ; procedure for A + B
        mov ax , A
        add ax , B
        mov res_add1, ax
        ret
add_proc1 endp
add_proc2 Proc Near   ; procedure for C + D
        mov ax , C
        add ax , D
        mov res_add2 , ax
        ret
add_proc2  endp
mul_num      Proc Near      ; procedure start to (A + B) × (C + D)
        mov ax , res_add1
        mul res_add2
        mov word_ptr Z , ax
        mov word_ptr Z + 2 , dx
        ret
add_proc2  endp
code ends
end START
```

Result :Z=(A+B) × (C+D)

Z=(1+2)*(3+4)

Z=15H

c) Interface two 8k ×8 RAM chips using memory mapping as an even addressed device.

Ans:-

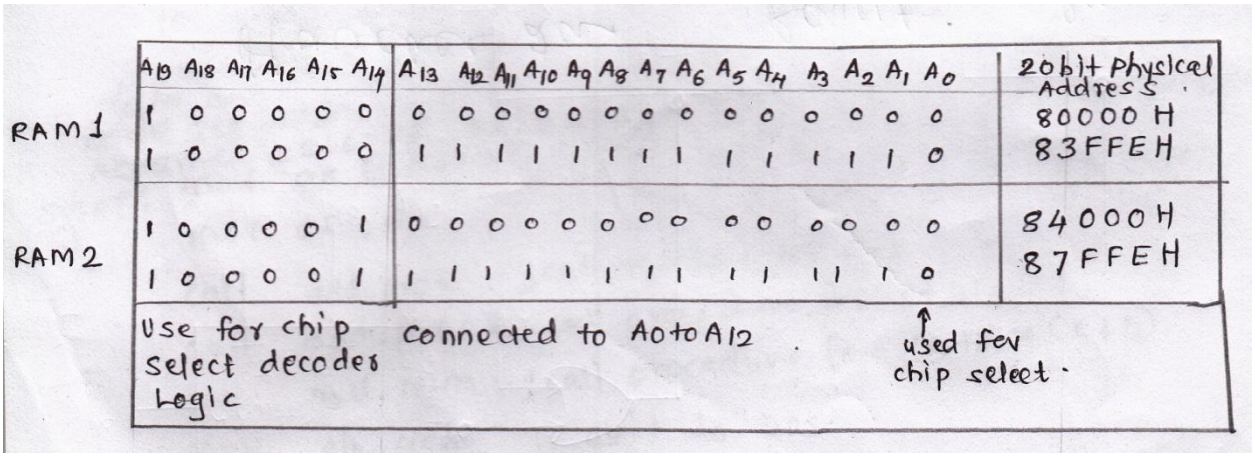
Address Mapping

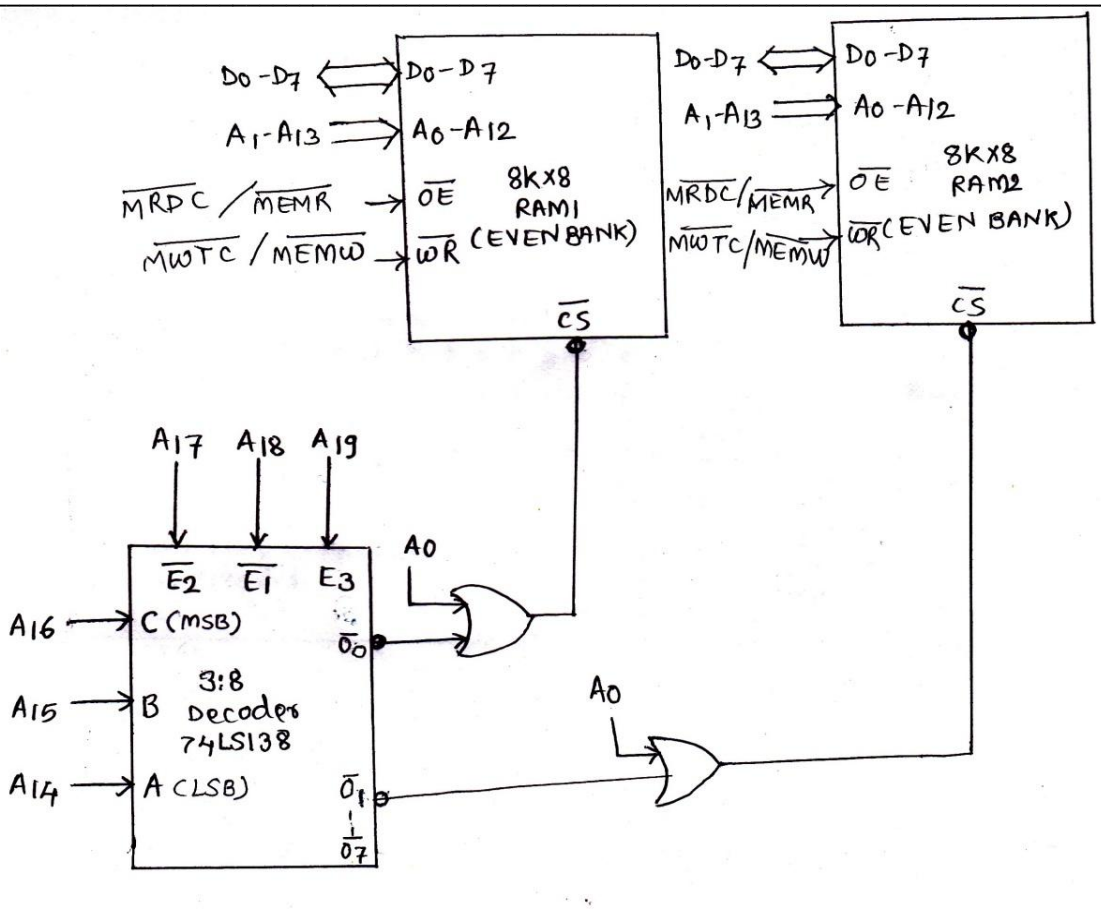
2^{13} = 8KB **(01 mark)**

RAM – 1 initial address **(01 mark)** Final address **(1 mark)**

RAM – 2 Initial address **(01 mark)** Final address **(1 mark)**

Interfacing **(03mark)**





Q.6 Solve any two **16**

a) Write the difference between memory mapped I/O and I/O mapped I/O.

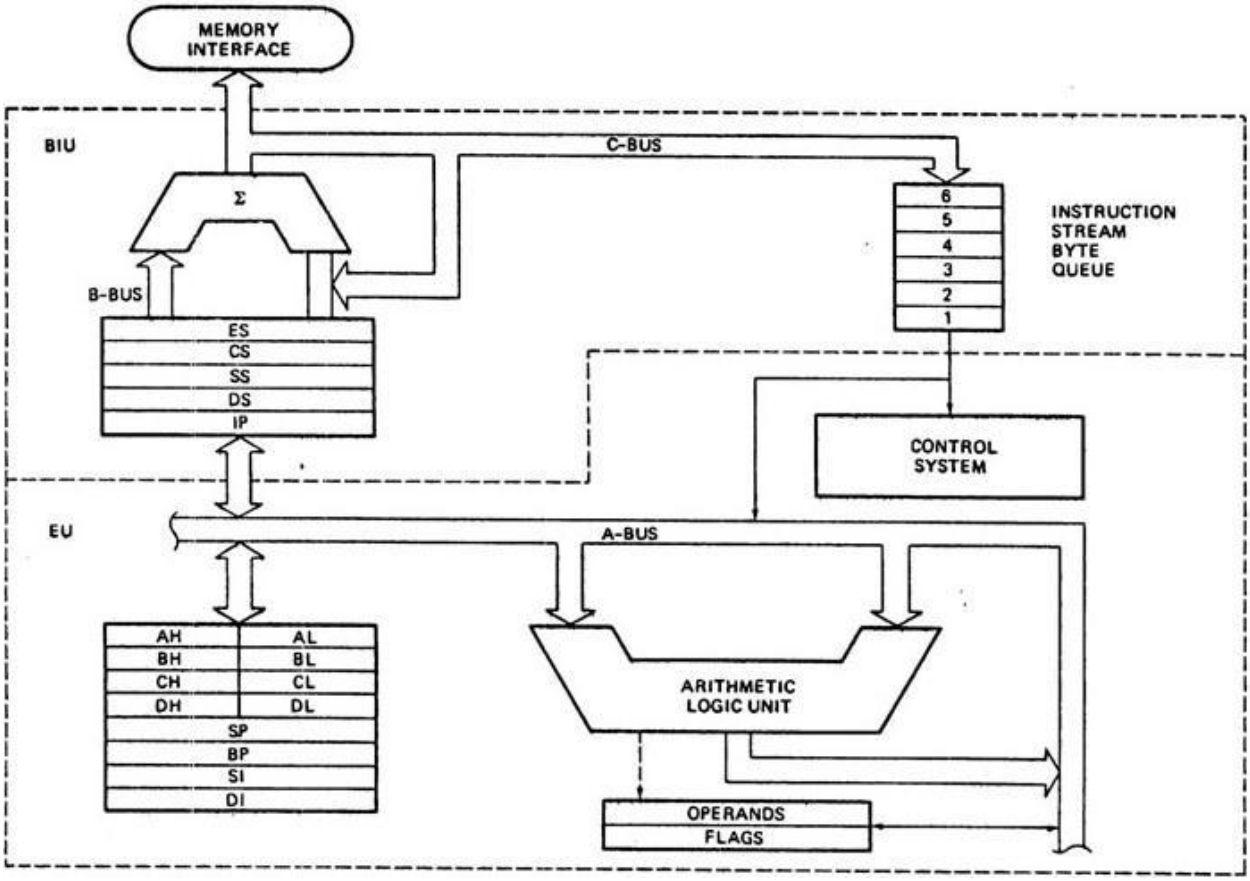
Ans:- **(Any 8 points. Each point carries 1M)**

Sr.No.	Memory Mapped I/O	I/O Mapped I/O
1	In this technique I/O and memory both are treated as Memory	In this technique I/O is treated as I/O and memory is treated as memory.
2	$\overline{MEMR}$ and $\overline{MEMW}$ control signals are used to control read and write I/O operations.	$\overline{IOR}$ and $\overline{IOW}$ control signals are used to control read and write I/O operation.
3	Data transfer is between any resister and	Data transfer is between accumulator and I/O

	I/O device	device.
4	All memory related instructions are used to I/O devices.	IN and OUT instructions are required for I/O read and write operation.
5	In this case both I/O and memory have a 16-bit address.	In this case I/O has an 8 bit address and memory has 16-bit address.
6	In this device address is 16-bit.Thus A0 to A15 lines are used to generate the device address.	In this device address is 8-bit .Thus A0 to A7 or A8 to A15 lines are used to generate device address.
7	Address decoding logic is complicated and expensive.	Address decoding logic is simple as less hardware is requited.
8	I/O devices and memory are distinguished by only addresses.	I/O devices and memory are distinguished by control signals and addresses.
9	Arithmetic and logical operations can be performed on I/O ports.	Arithmetic and logical operations cannot be performed on I/O ports
10	Can interface maximum memory of 64KB which also includes the I/O ports.	Can interface maximum memory of 64KB and 256 I/O ports.
11	Size of memory is reduced	Size of memory is not reduced.

b) Draw functional block diagram of 8086 microprocessor and describe in detail.

Ans: - (Diagram 4M, BIU-2M,EU-2M)(Explanation of individual in detail is not expected)



The 8086 CPU is divided into two independent functional parts, the bus interface unit or BIU, and the execution unit or EU.

The Bus Interface Unit

The BIU handles all data and addresses on the buses for the execution unit such as it sends out addresses, fetches instructions from memory, reads data from ports and memory as well as writes data to ports and memory.

In BIU there are so many functional groups or parts these are as follows.

Instruction Queue

To increase the execution speed, BIU fetches as many as six instruction bytes ahead to time from memory. The prefetched instruction bytes are held for the EU in a first in first out group of registers called a instruction queue.

Segment Registers

The BIU contains four 16-bit segment registers. They are: the extra segment (ES) register, the code segment (CS) registers, the data segment (DS) registers, and the stack segment (SS) registers. These segment registers are used to hold the upper 16 bits of the starting address for each of the segments

Instruction Pointer (IP)

The instruction pointer (IP) holds the 16-bit address of the next code byte within this code segment.

The Execution Unit

The execution unit (EU) tells the BIU where to fetch instructions or data from, decodes instructions, and executes instructions.

A decoder in the EU translates instructions fetched from memory to generate different internal or external control signals that required performing the operation. The EU has a 16-bit ALU, which can perform arithmetic operations such as add, subtract etc. and logical operations such as AND, OR, XOR, increment, decrement etc.

Flag Register

A 16-bit flag register is a flip-flop which indicates some condition produced by the execution of an instruction or controls certain operations of the EU. They are modified automatically by CPU after mathematical operations. It has 9 flags and they are divided into two categories:

1. Conditional Flags
2. Control Flags

General Purpose Registers

The EU has eight general purpose registers labeled AH, AL, BH, BL, CH, CL, DH, and DL. These registers can be used individually for temporary storage of 8-bit data. The AL register is also called the **accumulator**

Stack Pointer Register

The stack pointer (SP) register contains the 16-bit offset from the start of the segment to the memory location where a word was most recently stored on the stack.

Other Pointer and Index Registers

The EU also contains a 16-bit source index (SI) register, base pointer (BP) registers, and Destination Index (DI) registers. These three registers can be mainly used for temporary storage of 16-bit data just like a general purpose registers.

d) Describe various string instructions in brief.

Ans:- (Any 4 instruction -2M each [1M –Syntax, 1M Description])

String is a collection of data stored in consecutive memory locations. The various string instructions in 8086 are:

1] **REP:** REP is a prefix which is written before one of the string instructions. It will cause during length counter CX to be decremented and the string instruction to be repeated until CX becomes 0.

Two more prefix.

REPE/REPZ: Repeat if Equal /Repeat if Zero.

It will cause string instructions to be repeated as long as the compared bytes or words are equal and CX≠0.

REPNE/REPNZ: Repeat if not equal/Repeat if not zero.

It repeats the strings instructions as long as compared bytes or words are not equal and CX≠0.

Example: REP MOVSB

2] MOVS/ MOVSB/ MOVSW - Move String byte or word.

Syntax

MOVS destination, source

MOVSB **destination, source**

MOVSW **destination, source**

Operation: ES:[DI]<----- DS:[SI]

It copies a byte or word a location in data segment to a location in extra segment. The offset of source is pointed by SI and offset of destination is pointed by DI.CX register contain counter and direction flag (DE) will be set or reset to auto increment or auto decrement pointers after one move.

Example

LEA SI, Source

LEA DI, destination

CLD

MOV CX, 04H

REP MOVSB

3] CMPS /CMPSB/CMPSW: Compare string byte or Words.

Syntax

CMPS destination, source

CMPSB **destination, source**

CMPSW **destination, source**

Operation: Flags affected < ----- DS:[SI]- ES:[DI]

It compares a byte or word in one string with a byte or word in another string. SI holds the offset of source and DI holds offset of destination strings. CS contains counter and DF=0 or 1 to auto increment or auto decrement pointer after comparing one byte/word.

Example

LEA SI, Source

LEA DI, destination

CLD

MOV CX, 100

REPE CMPSB

4] SCAS/SCASB/SCASW: Scan a string byte or word.

Syntax

SCAS/SCASB/SCASW

Operation: Flags affected < ----- AL/AX-ES: [DI]

It compares a byte or word in AL/AX with a byte /word pointed by ES:DI. The string to be scanned must be in the extra segment and pointed by DI. CX contains counter and DF may be 0 or 1.

When the match is found in the string execution stops and ZF=1 otherwise ZF=0

Example

LEA DI, destination

MOV Al,0DH

MOV CX,80H

CLD

REPNE SCASB

5] LODS/LODSB/LODSW: Load String byte into AL or Load String word into AX.

Syntax: LODS/LODSB/LODSW

Operation: AL/AX < ----- DS: [SI]

IT copies a byte or word from string pointed by SI in data segment into AL or AX.CX may contain the counter and DF may be either 0 or 1

Example

LEA SI, destination

CLD

LODSB

6] STOS/STOSB/STOSW (Store Byte or Word in AL/AX)

Syntax STOS/**STOSB/STOSW**

Operation: ES:[DI] < ----- AL/AX

It copies a byte or word from AL or AX to a memory location pointed by DI in extra segment CX may contain the counter and DF may either set or reset.