Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills)
4) While assessing figures, examiner may give credit for principal components indicated in the Figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any Equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant. Values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgment on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

Q.1 a) Attempt any SIX of the following: 12 M

(i) State the function of following pins of 8085 microprocessor.
(1) Ready
(2) Trap

Ans: (One function of each: 1 Mark)

(1) Ready:

- It is an active high signal used to **synchronize µp with slower peripherals.**
- µp samples Ready input in the beginning of every Machine cycle.
- If found low, µp executes WAIT CYCLE, after which it resamples READY pin till it finds Ready pin HIGH. Therefore, µp remains in the WAIT STATE until the READY pin becomes high again.

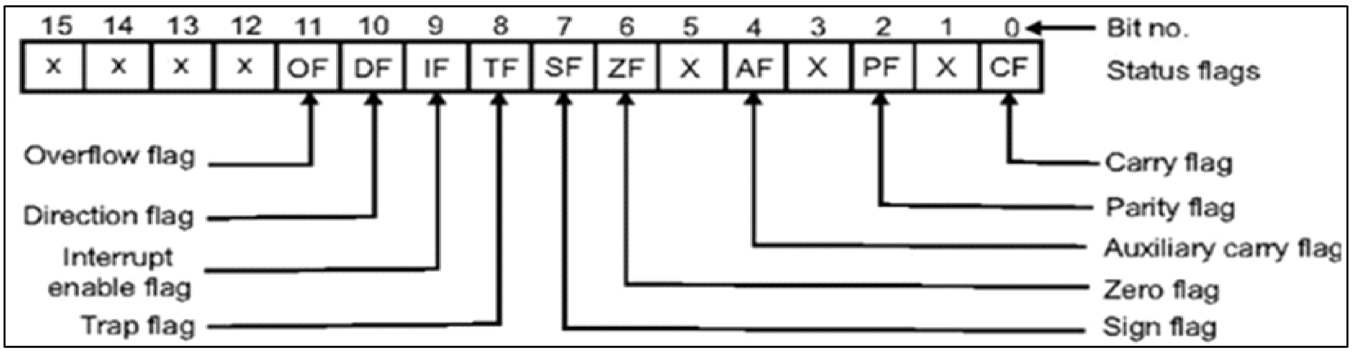
(2) Trap:

- It is an edge as well as level triggered, highest priority, non-maskable vectored interrupt.
- This interrupt transfers the microprocessor's control to location 0024H

(ii) Draw labelled flag register format of 8086 microprocessor.

Ans: (Correct Diagram: 2 Marks)

Flag Register Format Of 8086:



(iii) State any two example of immediate addressing mode and two example of direct addressing mode.

Ans: (Each mode 2 examples: ½ Mark each)
[NOTE- Any other valid example can be considered]

Examples of immediate addressing mode:

- MOV AX , 0030H
- ADD AL, 20H

Examples of direct addressing mode:

- MOV [3000H], AL
- AND BX, [2050H]

(iv) Define flowchart and algorithm.

Ans: (Correct Definition: 1 Mark each)

Flowchart:

- The flowchart is a graphically representation of the program operation or task.

Algorithm:

- The formula or sequence of operations to be performed by the program can be specified as a step in general. English is called algorithm.

(v) List any four salient features of 8085 microprocessor.

Ans : (Any Four Features – ½ Mark each)

Features of 8085:

1. 16 address line so 2^{16}=64 Kbytes of memory can be addressed.
2. Operating clock frequency is 3MHz and minimum clock frequency is 500 KHz.
3. On chip bus controller.
4. Provide 74 instructions with five addressing modes.
5. 8085 is 8 bit microprocessor.
6. Provides 5 level hardware interrupts and 8 software interrupts.
7. It can generate 8 bit I/O address so 2^{8}=256 input and 256 output ports can be accessed.
8. Requires a single +5 volt supply
9. Requires 2 phase, 50% duty cycle TTL clock
10. Provide 2 serial I/O lines, so peripheral can be interfaced with 8085 µp

(vi) What is pipelining? How it is implemented in 8086 microprocessor.

Ans: (**Definition -1 Mark, Description -1 Mark)**

Definition:

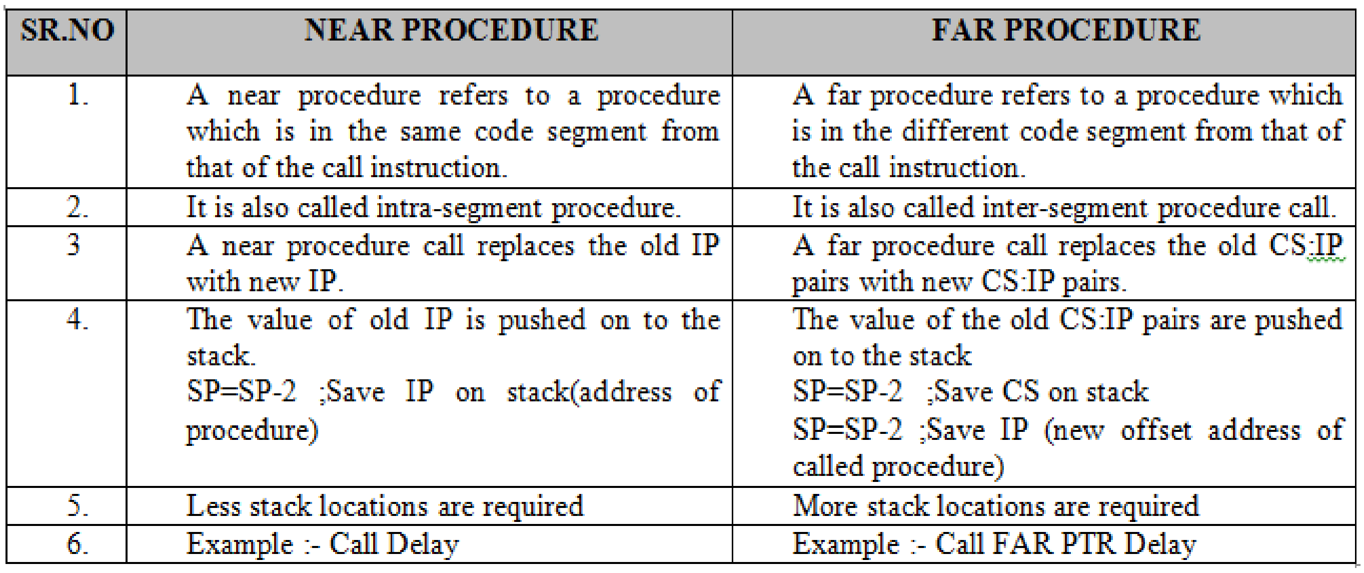
- Process of fetching the next instruction while the current instruction is executing is called pipelining which will reduce the execution time.

Description:

- In 8086, pipelining is the technique of overlapping instruction fetch and execution mechanism.
- To speed up program execution, the BIU fetches as many as six instruction bytes ahead of time from
- memory. The size of instruction prefetch queue in 8086 is 6 bytes**.**
- While executing one instruction other instruction can be fetched. Thus it avoids the waiting time for execution unit to receive other instruction.

- BIU stores the fetched instructions in a 6 byte FIFO queue. The BIU can be fetching instructions bytes while the EU is decoding an instruction or executing an instruction which does not require use of the buses.
- When the EU is ready for its next instruction, it simply reads the instruction from the queue in the BIU.
- This is much faster than sending out an address to the system memory and waiting for memory to send back the next instruction byte or bytes.
- This improves overall speed of the processor.

(vii) State any two differences between NEAR and FAR procedure.
Ans: (Any 2 points : 1 Mark Each)



SR.NO	NEAR PROCEDURE	FAR PROCEDURE
1.	A near procedure refers to a procedure which is in the same code segment from that of the call instruction.	A far procedure refers to a procedure which is in the different code segment from that of the call instruction.
2.	It is also called intra-segment procedure.	It is also called inter-segment procedure call.
3	A near procedure call replaces the old IP with new IP.	A far procedure call replaces the old CS:IP pairs with new CS:IP pairs.
4.	The value of old IP is pushed on to the stack. SP=SP-2 ;Save IP on stack(address of procedure)	The value of the old CS:IP pairs are pushed on to the stack SP=SP-2 ;Save CS on stack SP=SP-2 ;Save IP (new offset address of called procedure)
5.	Less stack locations are required	More stack locations are required
6.	Example :- Call Delay	Example :- Call FAR PTR Delay

(viii) Write assembly language instructions of 8086 microprocessor to
(1) Divide the content of AX register by 50H
(2) Rotate the content of BX register by 4 bit towards left
Ans: (Correct Instruction: 1 Mark each)
(1) Divide the content of AX register by 50H:

```
MOV BL,50H
DIV BL
```

(2) Rotate the content of BX register by 4 bit towards left:

```
MOV CL,04H
ROL BX,CL
```

OR

```
MOV CL,04H
RCL BX,CL
```

b) Attempt any TWO of the following: **8M**
(i) State the function of linker and assembler.
Ans: (Any ONE correct function of each: 2 Marks)
Linker:

- It is a programming tool used to convert Object code into executable program called .EXE module.
- It combines, if requested, more than one separated assembled modules into one executable module such as two or more assembly programs or an assembly language with C program.

Assembler:

- Assembler is a program that translates assembly language program to the correct binary code for each instruction i.e. machine code and generate the file called as object file with extension .obj .
- It also displays syntax errors in the program, if any.
- It can be also be used to produce list (.lst) which contains assembly language statements, binary codes, and offset address for each instruction.

(ii) Explain the following assembler directive.
(1) ORG
(2) EQU
(3) DD
(4) ASSUME

Ans: (Explanation of each : 1 Mark)

1) **ORG : Originate**
The directive ORG assigns the location counter with value specified in the directive. It helps in placing the machine code in the specified location while translating instructions into machine codes by the assembler. $ is used to indicate current value of location counter
Syntax: ORG [$+] Numeric_value
Example: ORG 2000H ; set location counter to 2000H
ORG $+100 ; increment value of location counter by 100 from its current.

2) **EQU :Equate to**
The EQU directive is used to declare the micro symbols to which some constant value is assigned. Micro assembler will replace every occurrence of the symbol in a program by its value.
Syntax: Symbol_name EQU expression
Example: CORRECTION_FACTOR EQU 100

3) **DD: -Define Double word (32-bits)**
It is used to declare a variable of type double word or to reserve memory locations which can be accessed as type double word(32-bits)
Example: NUMBER DD 12345678H ; Reserve 2 words in memory.

4) **ASSUME: -** Assume directive is used to tell Assembler the name of the logical segment it should use for the specified segment.
Example: Assume CS: MAP_CODE, DS: MAP_DATA

(iii) What is MACRO? Explain MACRO with suitable example.
Ans: (Definition or Syntax: 1 Mark, Macro Example: 3 Marks)

MACRO:

- Small sequence of the codes of the same pattern are repeated frequently at different places which perform the same operation on the different data of same data type, such repeated code can be written separately called as Macro.

(OR)
Macro definition or (Macro directive):
Syntax:
MACRO_NAME MACRO[ARG1,ARG2,…..ARGN)
…..
ENDM

Example: ***(NOTE: Any Same Type of Example can be considered)***

```
MYMACRO MACRO P1, P2, P3       ; Macro definition with arguments
MOV AX, P1
MOV BX, P2
MOV CX, P3
ENDM                           ; Indicates end of macro.
DATA SEGMENT
DATA ENDS
CODE SEGMENT
START:    ASSUME CS:CODE, DS:DATA

MOV AX,DATA
MOV DS,AX
MYMACRO 1, 2, 3                ; macro call
MYMACRO 4, 5, DX
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

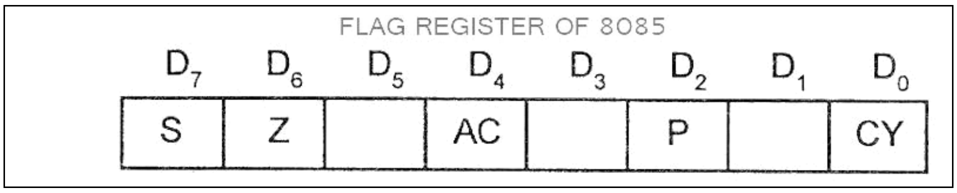
Q.2 Attempt any FOUR of the following: 8M

a) Draw the labelled flag register format of 8085 and explain the function of all flags.

Ans: (Diagram: 2 Marks, Description: 2 Marks)

Flags of 8085:

- ALU contains 5 flip-flop, which are set or reset after operation, according to data , conditions of result.



Carry Flag (CY):

- It is set when carry/borrow is generated from MSB(D7 bit)
- It is reset when no such carry/borrow is generated.

Parity Flag (P):

- 1= if result contains even no of 1's.
- 0=if result contains odd no of 1's

Auxiliary Carry Flag (AC):

- 1= it is set when carry/borrow is generated from lower nibble(bit D3) to higher nibble(bit D4)in 8-bit operations..
- Not used in 16 bit operation.
- It is used in BCD operation.

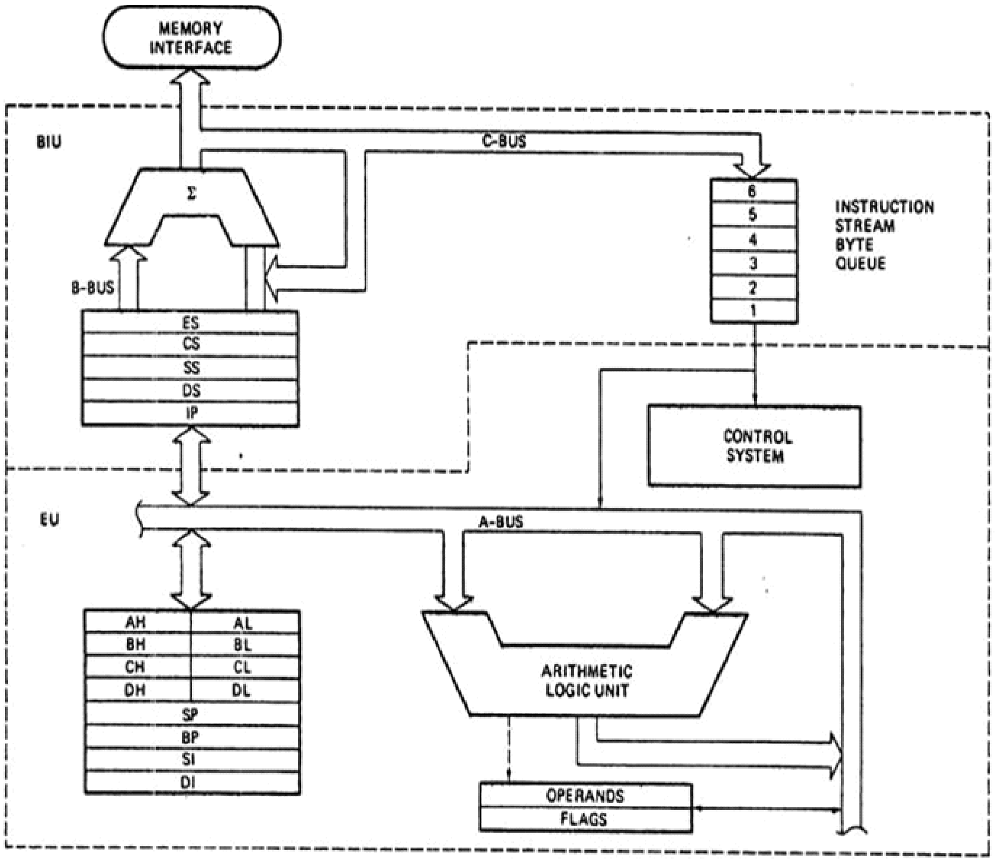
Zero Flag (Z)

- 1= it is set when result is equal to zero.
- 0=it is reset if the result is not equal to zero.

Sign flag (S)

- 1=it is set when MSB of result is 1(result is -ve no).
- 0=it is reset when MSB of result is 0.(result is +ve no).

b) Draw the neat labelled architecture diagram of 8086 microprocessor.
Ans: (Correct Diagram: 4 Marks)



c) State the function of following pins of 8086 microprocessor.
(i) NMI
(ii) $\overline{\text{TEST}}$
(iii) $\overline{\text{DEN}}$
(iv) MN/$\overline{\text{MX}}$

Ans: (Correct one function of each: 1 Mark)

(i) NMI:
- An edge triggered signal on this pin causes 8086 to interrupt the program it is executing and execute

- Interrupt service Procedure corresponding to Type-2 interrupt.
- NMI is Non-maskable by software.

(ii) $\overline{\text{TEST}}$:

- Used to test the status of math co-processor 8087.
- If signal goes low, execution will continue, else processor remains in an idle state.

(iii) $\overline{\text{DEN}}$:

- This is active low signal, to indicate availability of valid data over AD0-AD15.
- Used to enable transceivers(bi-directional buffers) 8286 or 74LS245 to separate data from multiplexed address/data signal.

(iv) MN/$\overline{\text{MX}}$:

- This signal indicates operating mode of 8086, minimum or maximum.
- When this pin connected to:
  1) Vcc, the processor operates in minimum mode,
  2) Ground, processor operates in maximum mode.

d) Explain the function of Stack Pointer (SP) and Program Counter (PC) of 8085 microprocessor.

Ans: **(Any 2 functions of each: 2 Marks)**

Stack pointer:

- It is a 16 bit register which is used to store the address of topmost filled memory location of stack memory.
- SP always points current top of stack.
- If data is stored in stack memory, the content of stack pointer is auto-decremented by two and if data is picked out from stack memory, the content of SP is auto-incremented by two.

Program counter:

- It maintains sequential execution of program written in memory.
- The PC stores the address of the next instruction which is going to execute.
- Since program counter stores the address of memory and in 8085 the address of memory is 16 bit. Hence program counter is 16 bit register.

e) Analyze the content of AL register and status of carry and auxiliary carry flag after execution of following instructions
MOV AL, 99H
ADD AL, 01H
DAA

Ans: **(Correct Flag Status: 2 Marks, Valid Steps: 2 Marks)**

```
                1001  1001  (99H
   ADD AL,01H          +  0000  0001  (01H)
                          -----------------------
                          1001  1010  (9AH)
            DAA        +  0110  0110  (66 H)  ; here Lower nibble > 9 (add 06H) and AL > 99H(add 60H)
                          -----------------------
                        1 0000  0000    ( CF=1 , AF=1 )
```

Final Answer in AL= 100

f) Explain how 20-bit physical address is generated by 8086 microprocessor. Calculate the physical address if CS=2308H and IP=76A9H

Ans: (Description: 2 Marks, Example: 2 Marks)

- **Generation of a physical address in 8086:-**

Segment registers carry 16 bit data, which is also known as base address. BIU attaches four 0 bits to LSB of the base address. So now this address becomes 20-bit address. Any base/pointer or index register carry 16 bit offset. Offset address is added into 20-bit base address which finally forms 20 bit physical address of memory location.

Example:- Given : CS = 2308H, IP = 76A9H

```
   CS :  23080H  ……...0 added by BIU(or Hardwired 0)
 + IP :   76A9H
 -------------------------
       2A729H
```

Q.3 Attempt any FOUR of the following: 16M

a) Explain DAA instruction with suitable example.

Ans:- (Explain: 2 marks, example: 2marks)

DAA – (Decimal Adjust AL after BCD Addition)
Syntax- DAA

Explanation:

This instruction is used to make sure the result of adding two packed BCD numbers is adjusted to be a correct BCD number.

The result of the addition must be in AL for DAA instruction to work correctly.

If the lower nibble in AL after addition is > 9 or Auxiliary Carry Flag is set, then add 6 to lower nibble of AL.

If the upper nibble in AL is > 9H or Carry Flag is set, and then add 6 to upper nibble of AL.

Example: - (Any Same Type of Example)

if AL=99 BCD and BL=99 BCD

Then ADD AL, BL

```
  1001 1001 = AL= 99 BCD
+ 1001 1001 = BL = 99 BCD
---------------------------------------
0011 0010 = AL =32 H and CF=1, AF=1
```

After the execution of DAA instruction, the result is CF = 1

```
 0011 0010 =AL =32 H AF =1
+ 0110 0110
------------------------
1001 1000 =AL =98 in BCD
```

(OR)

Explanation: (DAA: Decimal Adjust Accumulator)

- This is the only instruction that uses the auxiliary flag to perform the binary to BCD conversion, and the conversion procedure is described below. S, Z, AC, P, CY flags are altered to reflect the results of the operation .If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag

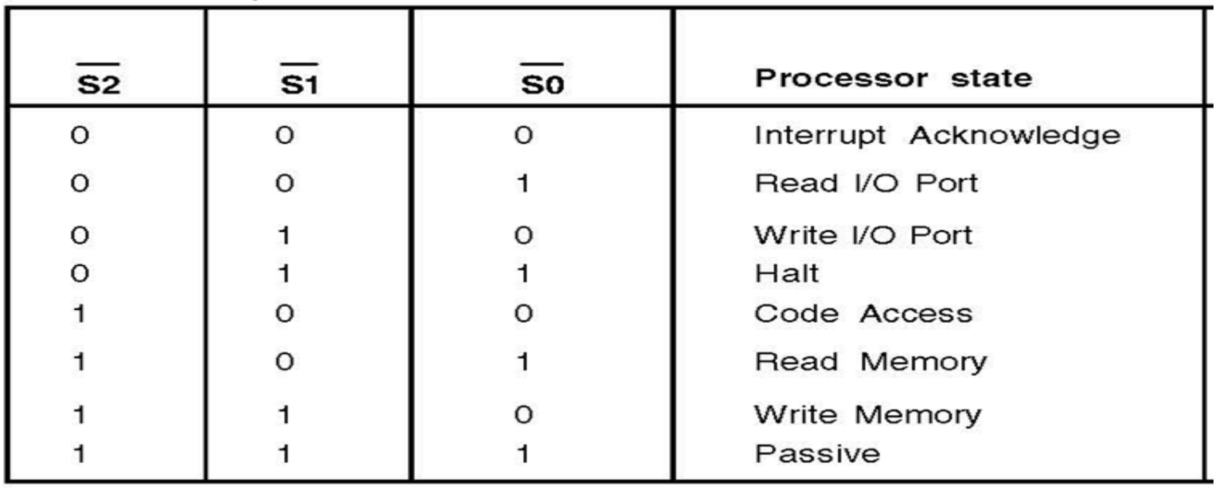
- is set, the instruction adds 6 to the low-order four bits. If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits.
- The DAA instruction (Decimal Adjust Accumulator) allows conversion of the 8-bit accumulator value to Binary Coded Decimal (BCD). If the low-order 4 bits of the accumulator are greater than 9, or the auxilliary carry flag is set, 6 is added to the low-order 4 bits of accumulator, then if the high-order 4 bits of the accumulator are greater than 9, or the carry flag is set, 6 is added to the high-order 4 bits of the accumulator.
- Execute DAA only after executing an ADD instruction that leaves a two-BCD-digit byte result in the AL register. The ADD operands should consist of two packed BCD digits. The DAA instruction adjusts AL to contain the correct two-digit packed decimal result.

Example :

```
  mov   al,38h        ;packed decimal "38"
  add   al,45h        ;add packed decimal "45"
  daa                 ;AL = 7Dh -> 83h  (with CF clear = 83 packed decimal)
                              AL    CF     AF
; after addition              7Dh   clear  clear
; daa 1st part (al+6)         83h   clear  set   (because al AND 0Fh > 9)
; daa 2nd part (nothing)      83h   clear  set   (al !> 99h && CF clear)
```

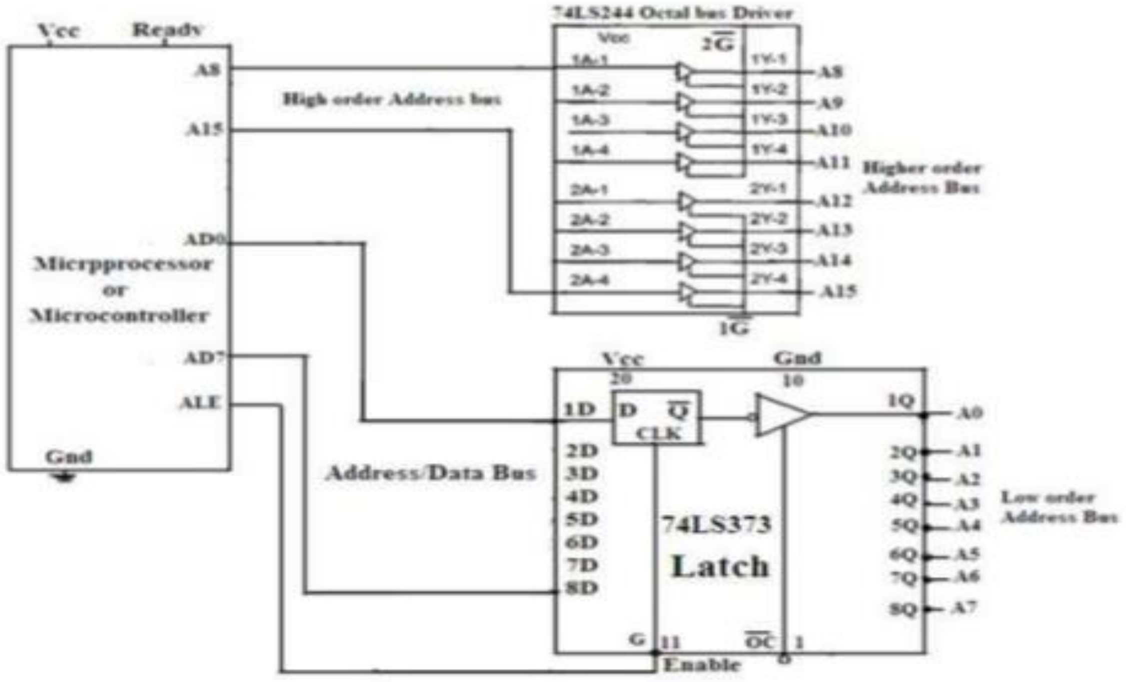
b) State all the control signal generated by So, S1, S2 with their function by 8086 micro Processor.

Ans:- (Correct control signal 4 marks)



$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Processor state
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O Port
0	1	0	Write I/O Port
0	1	1	Halt
1	0	0	Code Access
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive

c) Draw interfacing diagram of 74LS373 octal latch with 8086 microprocessor and explain it.
Ans: (Diagram 2 marks, explanation 2 marks)



Similarly ,in microprocessor /microcontrollers the address/data bus lines are multiplexed.The latching of lower order address bus is done by using the ALE signal from the microprocessor.For demultiplexing the bus , the IC 74LS373 which is transparent 8-bit Latch is used. This IC consists of 8 D flip-flops.The ALE signal is connected to clock through the Enable pin . So,when this ALE is high,the clock will allow any data at its input to the output.[This is the reason for calling it as transparent Latch]. So , **74LS373** 8-bit (octal) D latch latches the address bits A_0-A_7 when ALE is HIGH, and keeps them available when ALE is LOW.

d) Explain any two string operation instructions with suitable example.

Ans: (Any two each 2 marks)

MOVSB / MOVSW: Move String Byte or String Word

- Suppose a string of bytes stored in a set of consecutive memory locations is to be moved to another set of destination locations. The starting byte of source string is located in the memory location whose address may be computed using SI (Source Index) and DS (Data Segment) contents. The starting address of the destination locations where this string has to be relocated is given by DI (Destination Index) and ES (Extra Segment) contents.

CMPS: Compare String Byte or String Word

- The CMPS instruction can be used to compare two strings of byte or words. The length of the string must be stored in the register CX. If both the byte or word strings are equal, zero Flag is set. The REP instruction Prefix is used to repeat the operation till CX (counter) becomes zero or the condition specified by the REP Prefix is False.

SCAN: Scan String Byte or String Word

- This instruction scans a string of bytes or words for an operand byte or word specified in the register AL or AX. The String is pointed to by ES:DI register pair. The length of the string s stored in CX. The DF controls the mode for scanning of the string. Whenever a match to the specified operand, is found in the string, execution stops and the zero Flag is set. If no match is found, the zero flag is reset.

LODS: Load String Byte or String Word

- The LODS instruction loads the AL / AX register by the content of a string pointed to by DS : SI register pair. The SI is modified automatically depending upon DF, If it is a byte transfer (LODSB), the SI is modified by one and if it is a word transfer (LODSW), the SI is modified by two. No other Flags are affected by this instruction.

STOS: Store String Byte or String Word

- The STOS instruction Stores the AL / AX register contents to a location in the string pointer by ES : DI register pair. The DI is modified accordingly; No Flags are affected by this instruction. The direction Flag controls the String instruction execution, the source index SI and Destination Index DI are modified after each iteration automatically. If DF=1, then the execution follows auto decrement mode, SI and DI are decremented automatically after each iteration. If DF=0, then the execution follows auto increment mode. In this mode, SI and DI are incremented automatically after each iteration.

e) Write an assembly language program to multiply two 16 bit numbers.

Ans: (Correct program 4 marks)

```
DATA SEGMENT
      NUM1 DW 0004H
      NUM2 DW 0002H
      PROD_LOW   DW ?
      PROD_HIGH  DW  ?
DATA ENDS
CODE SEGMENT
      ASSUME  CS:CODE,DS:DATA
   START:MOV DX,DATA
      MOV DS,DX
      MOV AX,NUM1
      MOV BX,NUM2
      MUL BX
      MOV PROD_LOW,AX
      MOV PROD_HIGH,DX
      MOV AX,4C00H
      INT 21H
CODE ENDS
END START
```

AX * BX = 0004 * 0002 = 0000 0008 (Answer in DX:AX pair)

(OR)

```
ORG 100h
MOV AX, 0004H Move 1st 16-bit number to AX.
MOV BX, 0002H Move 2nd 16-bit number to BX.
MUL BX Multiply BX with AX and the result will be in DX:AX.
.END
```

f) Differentiate between 8085 and 8086 microprocessor.

Ans: (Any four points each point 1 mark)

SR.NO	POINT	8085	8086
1	Size	8bit	16 bit
2	Address bus	16 bit	20 bit
3	Memory	64KB	1MB
4	Instruction queue	Does not have	It has
5	Pipeline	Not support	Support
6	Multiprocessing support	Not support	support
7	I/O	256 inputs/ outputs	65536 inputs / outputs
8	Arithmetic Support	Integer and decimal	Integer and decimal , ASCII
9	Multiplication and division	Not supports	supports
10	Operating mode	one	two
11	External hardware	Less	More
12	Cost	low	High
13	Memory segmentation	Not Segmented	Segmented

Q.4 Attempt any FOUR of the following: 16M

a) Identify the addressing mode of following instructions.

(i) INC [4712H]
(ii) ADD AX, 4712H
(iii) DIV BL
(iv) MOV AX, [BX + SI]

Ans: (each 1 mark)

(i) INC [4712H] **:**Direct Addressing Mode
(ii) ADD AX, 4712H **:**Immediate Addressing Mode
(iii) DIV BL **:**Register Addressing Mode
(iv) MOV AX, [BX + SI] **:**Base Plus Index Mode (OR) Based Indexed Addressing Mode

b) Explain the following instructions with suitable of 8086 with suitable example.

(i) XLAT
(ii) AAA

Ans: (each instruction 2 marks)

(i) XLAT:

XLAT No operands Translate byte from table.
Copy value of memory byte at DS:[BX + unsigned AL] to AL register.

Algorithm:

AL = DS:[BX + unsigned AL]

Example:

```
ORG 100H
LEA BX, DAT
```

```
MOV AL, 2
XLATB    ; AL = 33H
RET
DAT DB 11H, 22H, 33H, 44H, 55H
```

ii) AAA (ASCII Adjust after Addition):

Corrects result in AH and AL after addition when working with BCD values.
It works according to the following Algorithm:

```
if low nibble of AL > 9 or AF = 1 then:
AL = AL + 6
AH = AH + 1
AF = 1
CF = 1
else
AF = 0
CF = 0


In both cases:
clear the high nibble of AL.
```

Example:

```
MOV AX, 15   ; AH = 00, AL = 0Fh
AAA          ; AH = 01, AL = 05
RET
```

c) Write an assembly language program to subtract two 16 bit numbers.

Ans: (correct program 4 marks).

```
DATA SEGMENT
     NUMBER1    DW 1555H
     NUMBER2    DW 1000H
     DIFFERENCE   DW  2 DUP(0)
DATA ENDS
CODE SEGMENT
      ASSUME CS:CODE,DS:DATA
      START:
      MOV DX,DATA
      MOV DS,DX
      MOV AX,NUMBER1
      MOV BX,NUMBER2
      SUB AX,BX
      MOV DIFFERENCE,AX
      JNC EXIT
      MOV DIFFERENCE+1,02
   EXIT:MOV AH,4CH
      INT 21H
CODE ENDS
END START
```

(OR)

```
DATA SEGMENT
NUM DW 4567H,2345H
DIF DW 1 DUP(0)
DATA ENDS
CODE SEGMENT
ASSUME
CS:CODE,DS:DATA
START: MOV AX,DATA
MOV DS,AX
CLC
; Clearing Carry
LEA SI,NUM ; SI pointed to the NUM
MOV AX,[SI] ; Move NUM1 to AX
SBB AX,[SI+2] ; Move the SI to Num2 and subtract with AX(Takes care for both smaller as well as
    larger Number subtraction)
MOV DIF,AX ;Store the result
MOV AH,4CH; normal termination to DOS
INT 21H
CODE ENDS
END START
```

d) Write an assembly language program to find largest number from array of 10 numbers.

Ans: (correct program 4 marks).

[Note: consider if students writes by for another method]

```
DATA SEGMENT
    ARRAY DB 15H,45H,08H,78H,56H,02H,04H,12H,23H,09H

    LARGEST DB 00H
DATA ENDS
CODE SEGMENT
START:ASSUME CS:CODE,DS:DATA
    MOV DX,DATA
    MOV DS,DX

    MOV CX,09H
    MOV SI ,OFFSET ARRAY
    MOV AL,[SI]
```

```
          UP:INC SI
          CMP AL,[SI]
          JNC NEXT          ;CHANGE
          MOV AL,[SI]
          NEXT:DEC CX
          JNZ UP
          MOV LARGEST,AL  ; AL=78h

          MOV AX,4C00H
          INT 21H
      CODE ENDS
         END START
```

e) Write an instruction of 8086 to perform following operation.
(i) Shift the content of BX register'3 bit toward left.
(ii) Move 1234H in DS register.

Ans: (Each 2 marks)

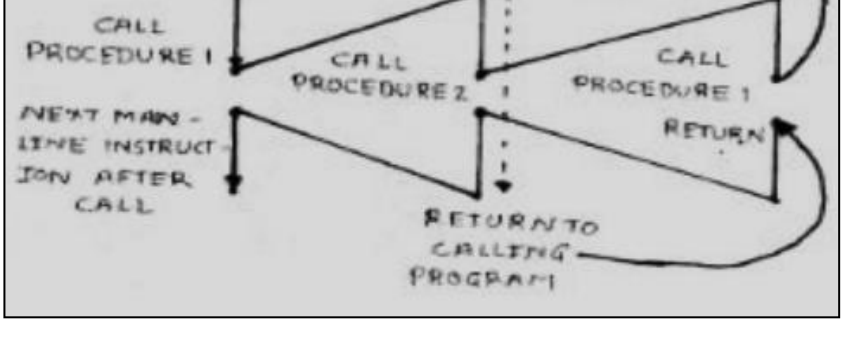
(i) <u>Shift the content of BX register'3 bit toward left</u> :
MOV CL, 03H
SHL BX, CL
(OR)
MOV CL,03H
SAL BX,CL

(ii) <u>Move 1234H in DS register</u>
MOV AX, 1234H
MOV DS, AX

f) Explain the re-entrant procedure with suitable diagram.
Ans: (Re-entrant procedure 1 M, Diagram with explanation 3 M).
[Note: Any program relevant to the diagram may also be considered]

- In some situation it may happen that Procedure 1is called from main program Procrdure2 is called from procedure1And procrdure1 is again called from procdure2.

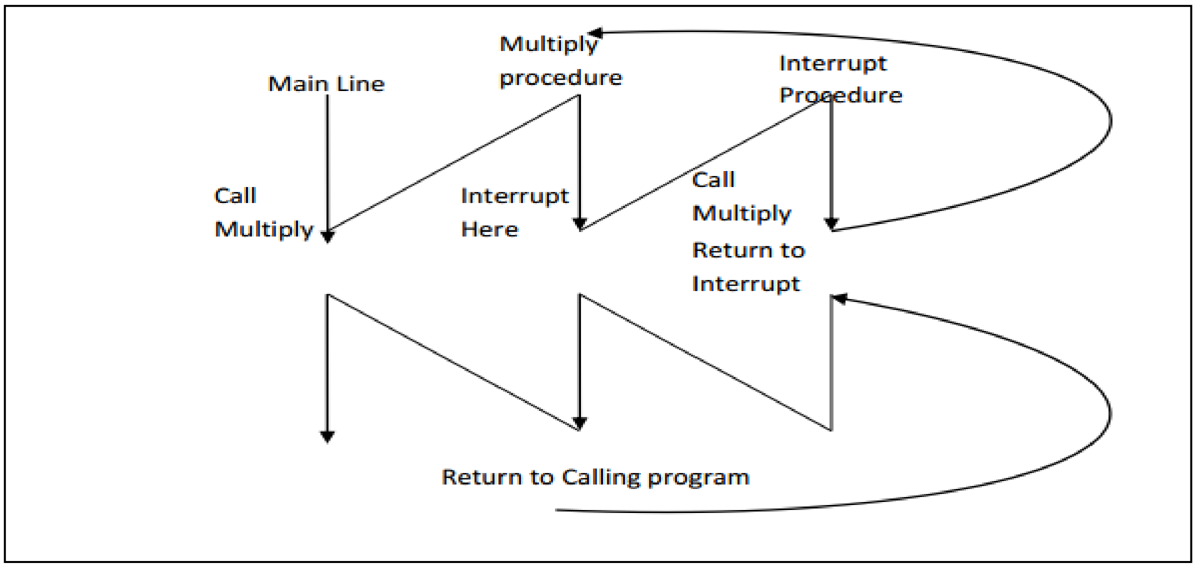
- In this situation program execution flow re-enters in the procedure1. These types of procedures are called re-entrant procedures.



OR

Re-entrant Procedures:

- A procedure is said to be re-entrant, if it can be interrupted, used and re-entered without losing or writing over anything. To be a re-entrant,
- Procedure must first push all the flags and registers used in the procedure.
- It should also use only registers or stack to pass parameters.
- The flow of re-entrant procedure for a multiply procedure when interrupt procedure is executed, as shown below.



Q.5 Attempt any FOUR of the following: 16M

a) Write an assembly language program to find length of string.

Ans: (Correct Program -4 Marks)

[Note: Any other logic may be considered]

```
    DATA SEGMENT
            STR1  DB  'STUDENT$'
            LENGTH_STRING DB?

    DATA ENDS
    ASSUME CS:CODE, DS:DATA
    CODE SEGMENT
          MOV AX, DATA
          MOV DS, AX
          MOV AL, '$'
          MOV CX, 00H
          MOV SI, OFFSET STR1
BACK: CMP AL, [SI]
          JE DOWN
          INC CL
          INC SI
          JMP BACK
DOWN: MOV LENGTH_STRING, CL
```

```
        MOV AX, 4C00H
         INT 21H
        CODE ENDS
  END
```

b) **How many times LOOPl will be executed in the following program. Write the content of AL register after the execution of following program.**

```
             MOV CL, OOH
             MOV AL, OOH
  LOOPI:     ADD AL, OIH
             DEC CL
             JNZ LOOP1
```

Ans: (Loop execution 2 M,content of AL 2 M)

The Loop will be executed 256 Times. The Value of CL will be decremented from FF_H to $00_{H.}$

The content of AL is '0' and it is incremented by 1 hence the value of AL will go from 00_H to FF_H but after the last iteration the Value of AL will be $00_{H.}$

Therefore, the value of AL=00_H and CL=00_H

c) **Write an ALP to count the number of '1' in a 16 bit number. Assume the number to be stored in BX register.**

Ans: (Correct Program -4 Marks)
[Note: Any other logic may be considered]

```
  .MODEL SMALL
  DATA
     NUM DW 0008H
     ONES DB 00H
   ENDS
  .CODE

        MOV AX, @DATA ; initialize data segment
         MOV DS, AX MOV CX, 10H ; initialize rotation counter by 16
        MOV BX, NUM ;load number in BX
  UP: ROR BX, 1 ; rotate number by 1 bit right
        JNC DN ; if bit not equal to 1 then go to dn
        INC ONES ; else increment ones by one
   DN: LOOP UP ;decrement rotation counter by 1 anf if not zero then go to up
        MOV CX, ONES ;move result in cx register.
        MOV AH, 4CH
        INT 21H
       ENDS
  END
```

d) **Explain the following instructions of 8086 with suitable example.**
(i) LOOP
(ii) INT D

Ans: (Description -1 Marks, Example -1 Marks)

[Note: Any Suitable Example can be considered]

(i) LOOP:

The LOOP instruction is a combination of a decrement of CX and a conditional jump. In the 8086, LOOP decrements CX and if CX is not equal to zero, it jumps to the address indicated by the label. If CX becomes a 0, the next sequential instruction executes.

Example:

```
          MOV BX, OFFSET PRICES     Point BX at first element in array
          MOV CX, 40                Load CX with number of elements in array
 NEXT:    MOV AL, [BX]              Get element from array
          INC AL                    Increment the content of AL
          MOV [BX], AL              Put result back in array
          INC BX                    Increment BX to point to next location
          LOOP NEXT                 Repeat until all elements adjusted
```

(ii) INT D (INT IMMEDIATE BYTE):

Interrupt numbered by immediate byte (0..255).
Flags Affected IF=0, CY,P,A,Z,S,O - unchanged
Algorithm: Push to stack
Flags register
CS
IP
IF = 0
Transfer control to interrupt procedure

Example:

```
MOV AH, 0Eh ; teletype.
MOV AL, 'A'
 INT 10h ; BIOS interrupt. RET
```

e) **Explain CALL and RET instructions with suitable example. Write syntax of CALL and RET instructions.**

Ans: (Description -1 Marks, Example -1 Marks)
[Note: Any Suitable Example can be considered]

- **CALL Instruction**: It is used to transfer program control to the sub-program or subroutine. The CALL can be NEAR, where the procedure is in the same segment whereas in FAR CALL, procedure is in a different segment.
  Syntax: CALL procedure name (direct/indirect)
  Operation: Steps executed during CALL
  Example:
  1) For Near CALL
  SP ← SP - 2
  Save IP on stack
  IP address of procedure
  2) For Far call
  SP ← SP-2
  Save CS on stack
  CS New segment base containing procedure
  SP ← SP-2

Save IP on stack
IP Starting address of called procedure

- **RET instruction:** it is used to transfer program execution control from a procedure to the next instruction immediate after the CALL instruction in the calling program.

  Syntax: RET
  Operation: Steps executed during RET

  Example:
  1) For Near Return
  IP Content from top of stack
  SP ←SP + 2
  2) For Far Return
  IP Contents from top of stack
  SP ←SP+2
  CS Contents of top of stack
  SP ←SP+2

f) Write an assembly language program using MACRO to perform following operations.
X = (A + B)*(C + D)

Ans: (correct Program 4 M) (Any other method can be considered)

```
ADD_NO1  MACRO A, B, RES_ADD1 ; MACRO DECLARATION (A+B)
MOV  AL, A
ADD  AL, B
MOV  RES_ADD1, AL
ENDM

ADD_NO2  MACRO C, D, RES_ADD2 ; MACRO DECLARATION (C+D)
MOV  AL, C
ADD  AL, D
MOV  RES_ADD1, AL
ENDM

MULTIPLY MACRO RES_ADD1, RES_ADD2,X; MACRO DECLARATION X=(A+B)*(C+D)
MOV AL, RES_ADD1
MUL RES_ADD2
MOV X,AL
MOV X+1,AH
END M

DATA SEGMENT
A DB 01H
B DB 02H
C DB 03H
D DB 04H
RES_ADD1    DB       ? ; RESULT OF A+B
RES_ADD2    DB      ? ; RESULT OF C+D
X         DW      ? ; RESULT OF (A+B) × (C+D)
```

```
DATA ENDS

CODE SEGMENT
START:ASSUME CS: CODE,DS: DATA
MOV AX, DATA ; INITIALIZE DATA SEGMENT
MOV DS, AX
ADD_NO1 A, B, RES_ADD1; CALL MACRO TO ADD
ADD_NO2 C, D, RES_ADD2;CALL MACRO TO ADD
MULTIPLY  RES_ADD1, RES_ADD2,X ;CALL MACRO TO MULTIPLY

MOV AX, 4C00H
INT 21H
ENDS
END START
```

(OR)

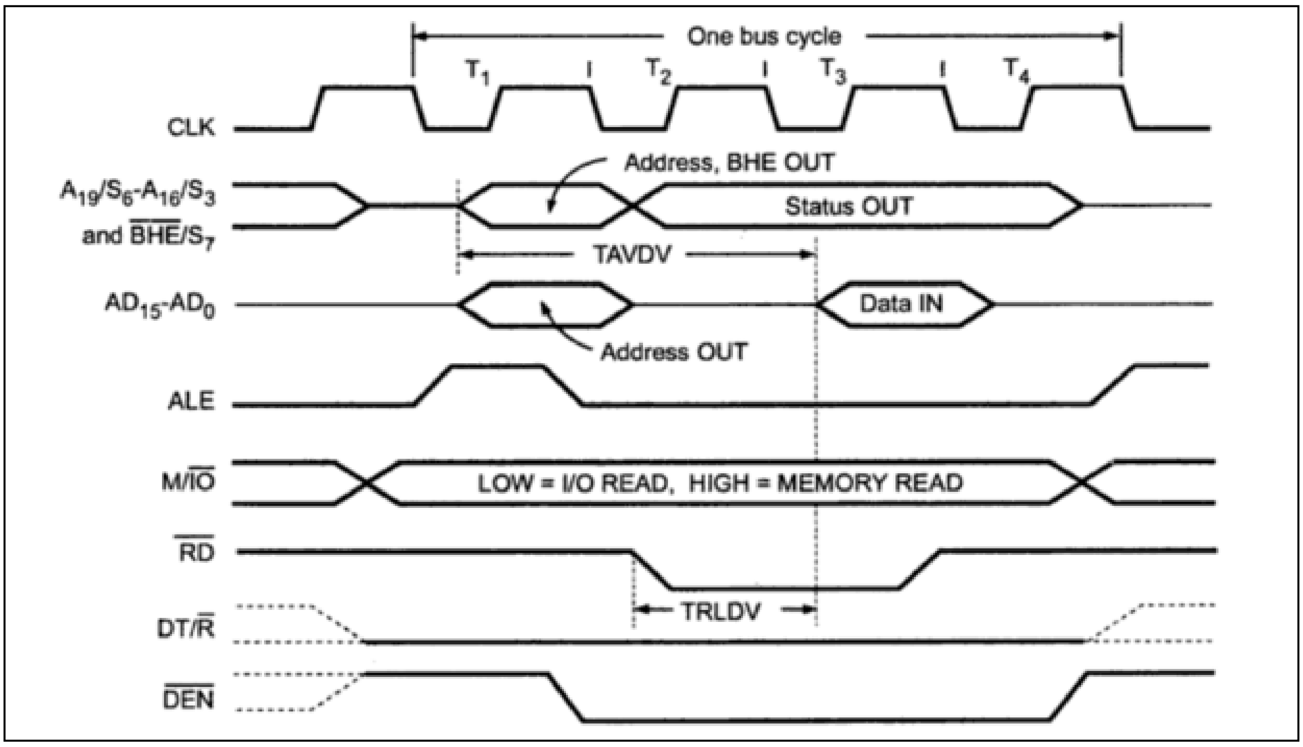
```
OPERATION MACRO A, B,C,D, RES_ADD1,RES_ADD2,X ; MACRO DECLARATION
(A+B)*(C+D)
MOV  AL, A
ADD  AL, B
MOV  RES_ADD1, AL
MOV AL,C
ADD AL,D
MOV RES_ADD2, AL
MOV AL, RES_ADD1
MUL RES_ADD2
MOV X,AL
MOV X+1,AH
ENDM

DATA SEGMENT
A DB 01H
B DB 02H
C DB 03H
D DB 04H
RES_ADD1    DB      ? ; RESULT OF A+B
RES_ADD2    DB      ? ; RESULT OF C+D
X         DW     ? ; RESULT OF (A+B) × (C+D)
DATA ENDS

CODE SEGMENT
ASSUME CS: CODE,DS: DATA

MOV AX, DATA ; INITIALIZE DATA SEGMENT
MOV DS, AX
OPERATION A, B,C,D RES_ADD1,RES_ADD2,X; CALL MACRO TO ADD & MULTIPLY
MOV AX, 4C00H
INT 21H
ENDS
END
```

6. Attempt any TWO of the following **16**

a) Draw and explain the timing diagram of 8086 in minimum mode.

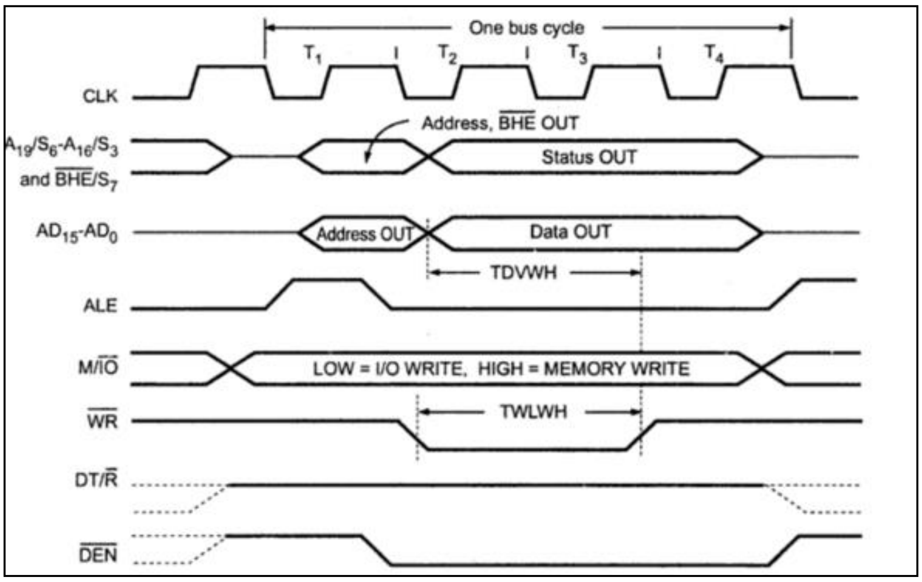
Ans: (Description 4 Marks and Diagram 4 Marks each for Read and Write Timing Diagrams)

- In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/$\overline{MX}$ pin to logic 1.
- The working of the minimum mode configuration system can be better described in terms of the timing diagrams.
- The opcode fetch and read cycles are similar. Hence the timing diagram can be categorized in two parts, the first is the timing diagram for read cycle and the second is the timing diagram for write cycle.
- The read cycle begins in T1 with the assertion of address latch enable (ALE) signal and also M / $\overline{IO}$ signal. During the negative going edge of this signal, the valid address is latched on the local bus.
- The BHE and A_0 signals address low, high or both bytes. From T1 to T4, the M/IO signal indicates a memory or I/O operation.
- At T2, the address is removed from the local bus and is sent to the output. The bus is then tristated. The read ($\overline{RD}$) control signal is also activated in T2.
- The read ($\overline{RD}$) signal causes the address device to enable its data bus drivers. After $\overline{RD}$ goes low, the valid data is available on the data bus.
- The addressed device will drive the READY line high. When the processor returns the read signal to high level, the addressed device will again tristate its bus drivers



- A write cycle also begins with the assertion of ALE and the emission of the address. The M/$\overline{IO}$ signal is again asserted to indicate a memory or I/O operation.
- In T 2, after sending the address in T1, the processor sends the data to be written to the addressed location.

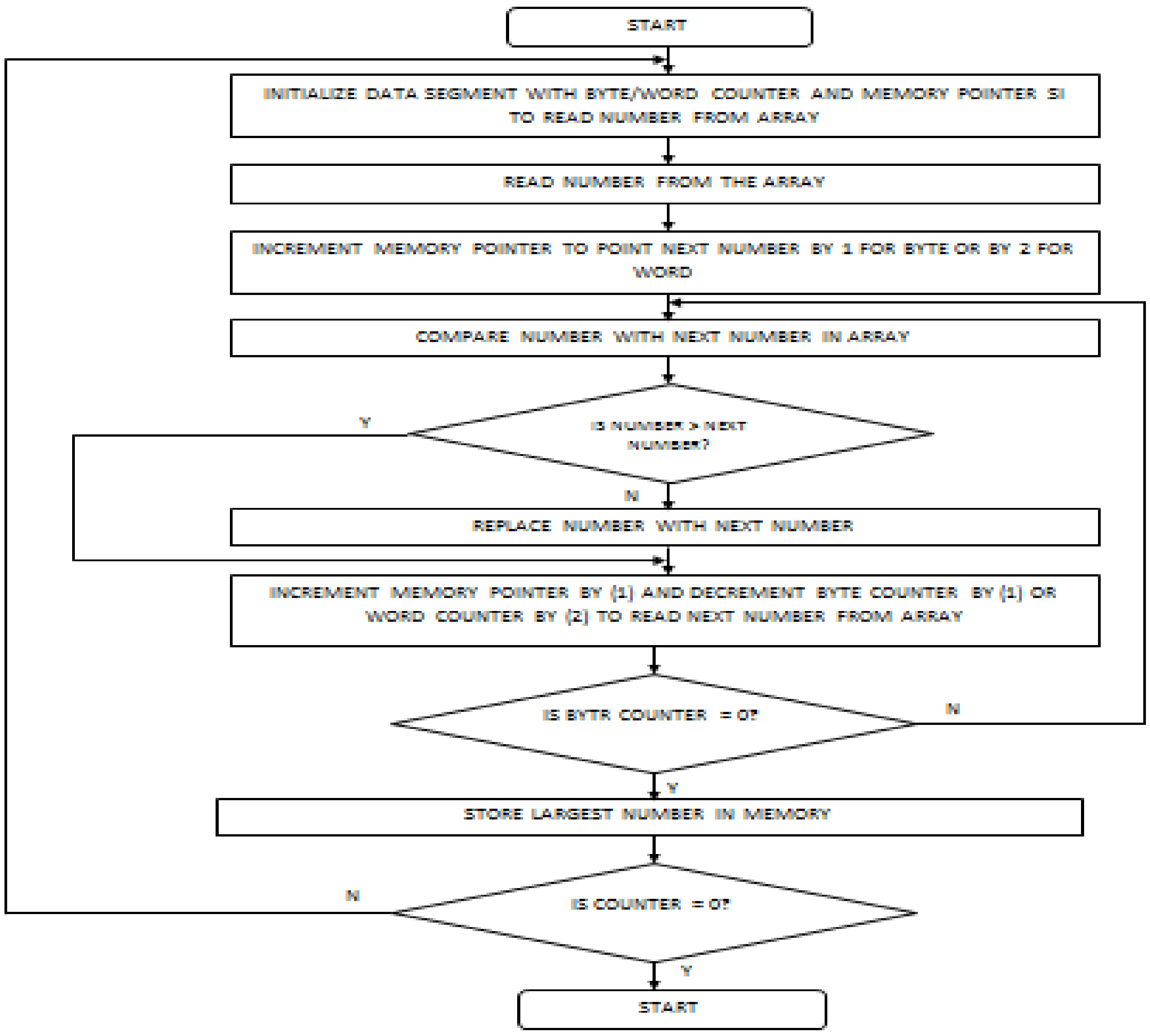
- The data remains on the bus until middle of T 4 state. The WR becomes active at the beginning of T 2 (unlike $\overline{RD}$ is somewhat delayed in T 2 to provide time for floating).
- The $\overline{BHE}$ and A0 signals are used to select the proper byte or bytes of memory or I/O word to be read or write.
- The M/$\overline{IO}$, $\overline{RD}$ and $\overline{WR}$ signals indicate the type of data transfer.



b) Write an assembly language program to sort l0 numbers in array in descending order. Draw the flow-chart for it.

Ans: (Program – 4 Marks and Flow Chart – 4 Marks)

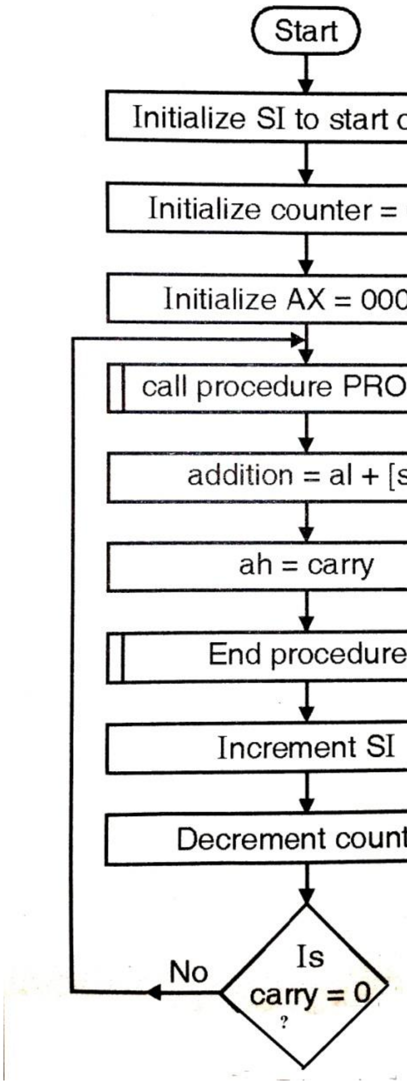
```
        DATA SEGMENT
              ARRAY DB 06H,09H,22H,02H,07H,10H,11H,12H,13H,14H
        DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
          MOV DS,AX
           MOV BL,0AH
STEP1: MOV SI, OFFSET ARRAY
          MOV CL,09H
STEP2: MOV AL,[SI]
         CMP AL,[SI+1]
          JNC DOWN
          XCHG AL, [SI+1]
          XCHG AL,[SI]
DOWN: ADD SI, 01
           LOOP STEP2
           DEC BL
           JNZ STEP1
          MOV AH,4CH
           INT 21H
             CODE ENDS
             END START
```


START
INITIALIZE DATA SEGMENT WITH BYTE/WORD COUNTER AND MEMORY POINTER SI TO READ NUMBER FROM ARRAY
READ NUMBER FROM THE ARRAY
INCREMENT MEMORY POINTER TO POINT NEXT NUMBER BY 1 FOR BYTE OR BY 2 FOR WORD
COMPARE NUMBER WITH NEXT NUMBER IN ARRAY
IS NUMBER > NEXT NUMBER?
Y
N
REPLACE NUMBER WITH NEXT NUMBER
INCREMENT MEMORY POINTER BY (1) AND DECREMENT BYTE COUNTER BY (1) OR WORD COUNTER BY (2) TO READ NEXT NUMBER FROM ARRAY
IS BYTR COUNTER = 0?
N
Y
STORE LARGEST NUMBER IN MEMORY
IS COUNTER = 0?
N
Y
START

Q6 C: Write an ALP for sum of series of 10 numbers using procedure. Also draw the flowchart.

```
      .model small
     .data
     Arr 1 db 01,02,03,04,05,06,07,08,09,0AH
     Sum db?
     Count dw 0AH
     .code
     mov ax, @data
     mov ds, ax,
     mov si, offset arr1
     mov cx,count
next: call proadd
     inc si
     loop next
     mov sum, al
     mov ah, 4CH
     int 21H
```

```
proadd proc near
add al [si]
adc ah, 00H
ret
proadd endp
end.
```


Start
Initialize SI to start of arr1
Initialize counter = 0AH
Initialize AX = 0000H
call procedure PROADD
addition = al + [si]
ah = carry
End procedure
Increment SI
Decrement counter
Is carry = 0 ?
No