MODEL ANSWER

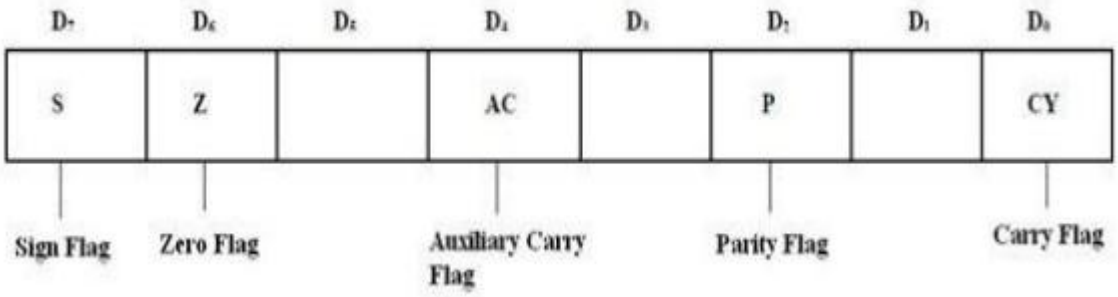
SUMMER– 17 EXAMINATION

Subject Title: Microprocessor and Programming. Subject Code: 17431

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 anyequivalent 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 judgement 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. No.	Sub Q.N.	Answer	Marking Scheme
Q.1		**Attempt any SIX of the following :**	**12-Total Marks**
	a)	**State the function of following pins of 8085 microprocessor :** **(i) INTR** **(ii) $\overline{\text{INTA}}$**	**2M**
	Ans:	**INTR**: - It is level triggered, non-vectored interrupt. When INTR occurs the microprocessor generates interrupt acknowledgement signal $\overline{\text{INTA}}$ **$\overline{\text{INTA}}$** It is an active low acknowledgement signal for INTR. This signal is used to get OPCODE & hence ISR address from external hardware.	**(1M each)**
	b)	**List any four features of 8086 microprocessor.**	**2M**
	Ans:	1) It is a 16 bit µp. 2) 8086 has a 20 bit address bus can access up to 2^{20} memory locations (1MB). 3) It can support up to 64K I/O ports. 4) It provides 16-bit registers. AX,BX,CX,DX,CS,SS,DS,ES,BP,SP,SI,DI,IP & FLAG REGISTER. 5) It has multiplexed address and data bus AD_0-AD_{15} and $A_{16} – A_{19}$. 6) 8086 is designed to operate in two modes, Minimum and Maximum. 7) It can prefetches up to 6 instruction bytes from memory and queues them in order to speed up instruction execution. 8) Interrupts:-8086 has 256 vectored interrupts. 9) Provides separate instructions for string manipulation. 10) Operating clock frequencies 5MHz, 8MHz, 10MHz.	**(Any Four Features [½ M each])**

c)	**Define immediate and direct addressing mode. Also give one example of each**	**2M**
Ans:	**1.Immediate addressing mode:**In this addressing mode, immediate data is a part of instruction, and appears in the form of successive byte or bytes **Example:** MOV AX,56D3H **2. Direct addressing mode:**In the direct addressing mode, a 16 bit address (offset) is directly specified in the instruction as a part of it. **Example:** MOV CL,[1000H]	**(Each Description ½ M and Each Example: ½ M) (Any example can be considered)**
d)	**List the program development steps for assembly language programming.**	**2M**
Ans:	1. Defining the problem 2. Algorithm 3. Flowchart 4. Initialization checklist 5. Choosing instructions 6. Converting algorithms to assembly language program	**(Correct steps : 2 M)**
e)	**Draw the format for flag register of 8085 microprocessor.**	**2M**
Ans:		**(Correct Format of flag register: 2 M)**
f)	**Give the steps in physical address generation in 8086 microprocessor.**	**2M**
Ans:	**Generation of 20 bit physical address in 8086** :- 1. Segment registers carry 16 bit data, which is also known as base address. 2. BIU appends four 0 bits to LSB of the base address. This address becomes 20-bit address. 3. Any base/pointer or index register carries 16 bit offset. 4. Offset address is added into 20-bit base address which finally forms 20 bit physical address of memory location.	**(Correct Steps : 2M)**
g)	**Give the syntax for defining a procedure.**	**2M**
Ans:	Procedure_Name PROC ---------- Procedure Statements Procedure_Name ENDP.	**(Correct Syntax : 2 M)**
h)	**Write assembly language instruction of 8086 microprocessor to :** **i. Copy 1000H to register BX**	**2M**

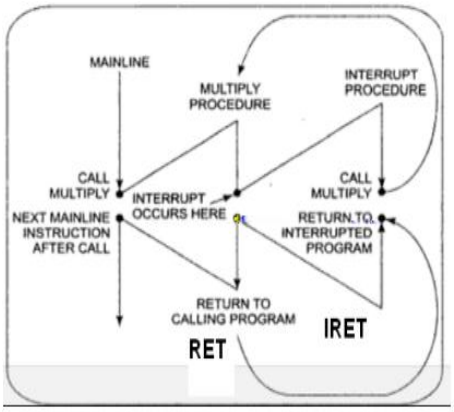
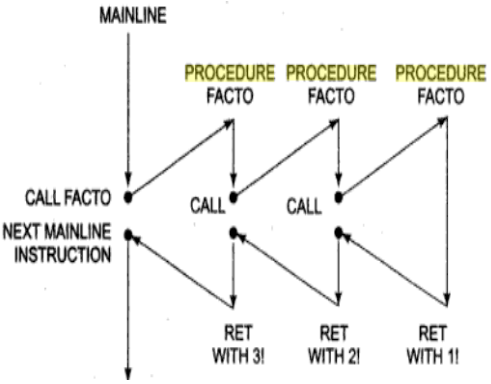
	ii. Rotate register BL left four times	
Ans:	i) MOV BX, 1000H ii) MOV CL, 04H RCL BL, CL Or MOV CL, 04H ROL BL, CL	**(1M each)**
B)	**Attempt any TWO of the following :**	**8M**
a)	**State the function of Assembler and Debugger**	**4M**
Ans:	**Assembler:-** 1. Assembler is a program that translates assembly language program to the correct binary code. 2. It also generates the file called as object file with extension .obj. 3. It also displays syntax errors in the program, if any. 4.It can be also be used to produce list(.lst) and .crf files **Debugger:** - 1. Debugger is a program that allows the execution of program in single step mode under the control of the user. 2. The errors in program can be located and corrected using a debugger. 3. Debugger generates .exe file.	**(Any two Functions of each :2M)**
b)	**Explain following assembler directives:** **(i) DB (ii) DW (iii) DD (iv) DQ**	**4M**
Ans:	**<u>DB (Define Byte)</u>** • This is used to define a byte type variable. • The range of values : 0 – 255 for unsigned numbers -128 to 127 for signed numbers • This can be used to define a single byte or multiple bytes **<u>DW (Define Word)</u>** • This is used to define a word (16-bit) type variable. • The range of values : 0 – 65535 for unsigned numbers -32768 to 32767 for signed numbers • This can be used to define a single word or multiple words **<u>DD (Define Double Word)</u>** • This is used to define a double word (32-bit) type variable. • This can be used to define a single double word or multiple double word **<u>DQ : Define Quad Word</u>**	**(Correct Explanation :1M each)**

• This is used to define a quad word (64-bit) type variable.

•This directive is used to tell the assembler to declare a variable 4 words in length or to reserve 4 words of storage.

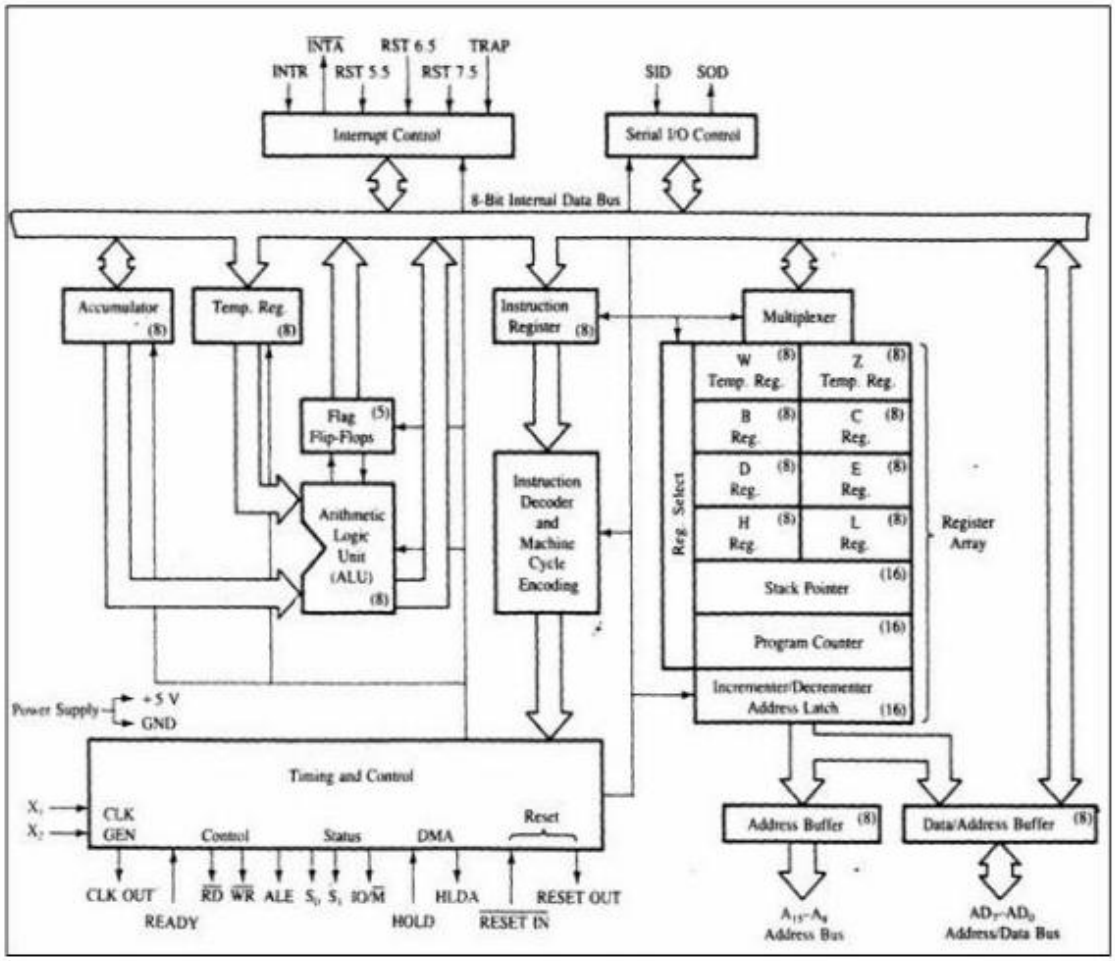
c) Differentiate between Re-entrant & Recursive procedure. **4M**

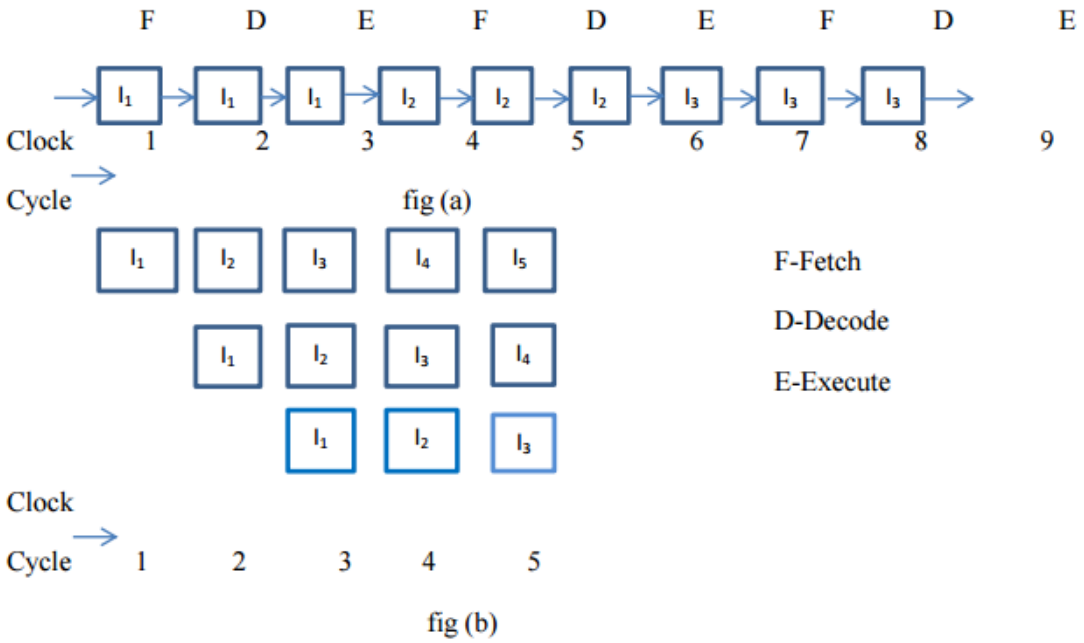
Ans: **(Any two points :2M each)**

Sr.No	Re-entrant procedure	Recursive procedure
1.	A procedure is said to be re-entrant, if it can be interrupted, used and re-entered without losing or writing over anything	A recursive procedure is a procedure which calls itself
2.	In Re-entrant Procedure must first push all the flags and registers used in the procedure .	In recursive procedure the program sets aside a few locations in stack for the storage of the parameters which are passed each time the computation is done and the value is returned
3.	To be a re-entrant, It should also use only registers or stack to pass parameters.	In recursive procedure Each value returned is then obtained by popping back from the stack at every RET instruction when executed at the end of the procedure.
4.	**Example** 	**Example** 

Q 2 **Attempt any FOUR of the following :** **16M**

a) Draw the Architecture of 8085 microprocessor. **4M**

	Ans:		**(Correct Diagram:4 M)**
	b)	**Explain the function of following pins of 8086 microprocessor :** • **MN/ MX** • **READY** • **ALE** • **DT/$\overline{R}$**	**4M**
	Ans:	**MN/ $\overline{MX}$:** The MN/$\overline{MX}$ pin is used to select either the minimum mode or maximum mode operation of the 8086. This is achieved by connecting this pin to either +5V directly (for minimum mode) or to the ground (for maximum mode). **READY:** This input is used to insert wait state into the timing cycle of the 8086. If the ready pin is at logic 1, it has no effect on the operation of the microprocessor. If it is logic 0, the 8086 enters the waits state and remains the idle. This pin is used to interface the operating peripherals with the 8086. **ALE :**This active high ,output signal used to indicate availability of valid address on address/data lines and is connected to latch enable input of latches (8282 or 74LS373) . **DT/$\overline{R}$-** This output line is used to decide the direction of data flow through the transceivers(bidirectional buffer). When the processor sends the data, this signal is high and when the processor is receiving data, this signal is low.	**(Correct Function of each :1M each)**

c)	**Explain the concept of pipelining in 8086 microprocessor with diagram.**	**4M**
Ans:	**Description:** Process of fetching the next instruction while the current instruction is executing is called pipelining which will reduce the execution time. The technique used to enable an instruction to complete with each clock cycle. Normally, on a non – pipelined processor, nine clock cycles are required for fetch, decode and execute cycles for the three instructions as shown in Fig (a). This takes longer time when compared to pipelined processor. In this ,the fetch, decode and execute operations are performed in parallel, so only five clock cycles are required to execute the same three instructions as shown Fig(b). In 8086, pipelining is implemented by providing 6 byte queue where as long as 6 one byte instructions can be stored well in advance and then one by one instruction goes for decoding and executions. So, while executing first instruction in a queue, processor decodes second instruction and fetches 3rd instruction from the memory In this way, 8086 perform fetch, decode and execute operation in parallel i.e. in single clock cycle as shown in above fig (b)  fig (b)	**(Description :2M, Diagram:2 M)**
d)	**List any four features and four limitation of 8085 microprocessor.**	**4M**
Ans:	**Features of 8085Microprocessor** 1. 8085 is 8 bit microprocessor. 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. 16 address line so 2^{16}=64 Kbytes of memory can be addressed. 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	**Any Four Features [½ M each] ,**

		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 **<u>Limitation of 8085Microprocessor</u>** 1.In 8085 microprocessor, microprocessor can perform any arithmetic and logical operation only on 8 bit data at a time. 2. In 8085 microprocessor, only 16 bit address lines, we can address only up to 64 KB of memory. 3.8085 microprocessor has multiplexed address and data bus, so extra hardware is required to separate address signals from the data signals. 4. Flags register has limited flags. 5. Interrupts are very limited in 8085. 6. Operating frequency is less in 8085 microprocessor, so the speed of execution is slow. 7. In 8085 microprocessor, we cannot design multi-processor system. 8.In 8085 microprocessor due to limited 8 bit size of the all registers, we can store limited data bytes in the microprocessor memory.	**Any Four Limitation [½ M each]**
	e)	**What will be the content of register AL after the execution of last instruction?** **MOV AL, 02H** **MOV BL, 02H** **SUB AL, BL** **MUL 08H**	**4M**

	Ans:	**[Note: If the student corrects MUL instructions and writes the output, marks can be given,**
		
		The above program segment gives an error at MUL 08H instructions hence it gives no output.
		
		After correction;
		MUL instruction will be MUL BL or MUL AL
		
		MOV AL, 02H ; AL=02H
		MOV BL, 02H ; BL=02H
		SUB AL,BL ; AL=00H
		MUL BL ; AX=0000H ;ANS: AL=00H
		(OR)
		MOV AL, 02H ; AL=02H
		MOV BL, 02H ; BL=02H
		SUB AL,BL ; AL=00H
		MUL AL ; AX=0000H ; ANS: AL=00H
		(OR)
		MOV AL, 02H ; AL=02H
		MOV BL, 02H ; BL=02H
		SUB AL,BL ; AL=00H
		MUL 08H ; Gives Error
			(Correct Answer : 4 M)

	f)	**Calculate the physical address for given :** **(i) DS = 73A2H SI =3216H** **(ii) CS = 7370H IP = 561EH**	**4M**
	Ans:	**(i)** DS = 73A2H SI =3216H DS 73A20H ……..... 0 is appended by BIU (or Hardwired zero) SI + 3 216 H ---------- 76C36H (ii) CS = 7370H IP = 561EH CS 73700H ……..... 0 is appended by BIU (or Hardwired zero) IP + 561EH ---------- 78D1EH	**(2M each)**
Q. 3		**Attempt any FOUR of the following :**	**16M**
	a)	**Write any two conditional and two unconditional branching instruction with their function. Give the syntax with one example each.**	**4M**
	Ans:	**<u>Unconditional Branch Instructions :</u>** In Unconditional control transfer instructions, the execution control is transferred to the specified location independent of any status or condition. The CS and IP are unconditionally modified to the new CS and IP. **1. CALL : Unconditional Call** The CALL instruction is used to transfer execution to a subprogram or procedure by storing return address on stack There are two types of calls-NEAR (Inter-segment) and FAR(Intra-segment call). Near call refers to a procedure call which is in the same code segment as the call instruction and far call refers to a procedure call which is in different code segment from that of the call instruction. **Syntax: CALL procedure_name** **2. RET: Return from the Procedure.** At the end of the procedure, the RET instruction must be executed. When it is executed, the previously stored content of IP and CS along with Flags are retrieved into the CS, IP and Flag registers from the stack and execution of the main program continues further. **Syntax :RET** **3. INT N: Interrupt Type N.** In the interrupt structure of 8086, 256 interrupts are defined corresponding to the types from 00H to FFH. When INT N instruction is executed, the type byte N is multiplied by 4 and the contents of IP and CS of the interrupt service routine will be taken from	**(Any 2 unconditional instruction explanation: 1M each)**

memory block in 0000 segment.

Syntax : **INT N**

4. INTO: Interrupt on Overflow

This instruction is executed, when the overflow flag OF is set. This is equivalent to a Type 4 Interrupt instruction.

Syntax : INTO

5. JMP: Unconditional Jump

This instruction unconditionally transfers the control of execution to the specified address using an 8-bit or 16-bit displacement. No Flags are affected by this instruction.

Syntax : JMP Label

6. IRET: Return from ISR

When it is executed, the values of IP, CS and Flags are retrieved from the stack to continue the execution of the main program.

Syntax: IRET

Example of unconditional CALL and RET ,INT instruction:

```
DATA SEGMENT
      NUM1 DB 10h
     NUM2 DB 20h
DATA ENDS
CODE SEGMENT
START:ASSUME CS:CODE,DS:DATA
      MOV DX,DATA
      MOV DS,DX
           CALL ADD_PROC
     MOV AX,4C00H
     INT 21H

 ADD_PROC  PROC
      MOV AL, NUM1
      MOV BL,NUM2
      ADD AL,BL
     RET
ADD_PROC ENDP
CODE ENDS
END START
```

Conditional Branch Instructions

When this instruction is executed, execution control is transferred to the address specified relatively in the instruction

1. JZ/JE Label

Transfer execution control to address 'Label', if ZF=1.

2. JNZ/JNE Label

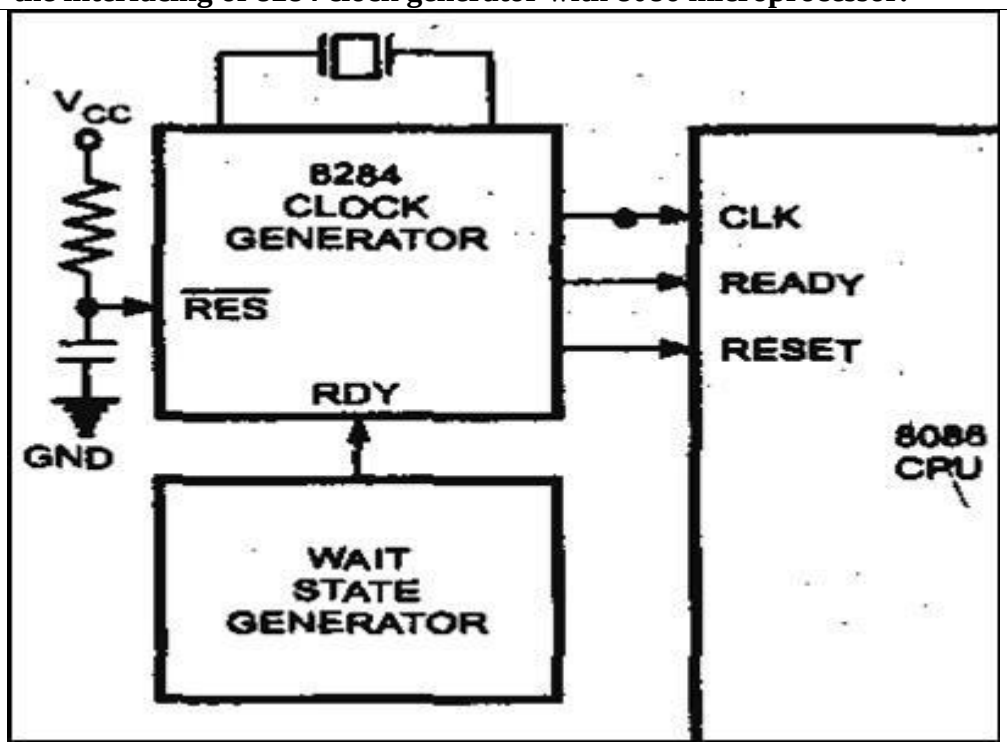
Transfer execution control to address 'Label', if ZF=0

3. JS Label

(Any 2 conditional instruction explanation: 1M each)

		Transfer execution control to address 'Label', if SF=1.
		4. JNS Label
		Transfer execution control to address 'Label', if SF=0.
		5. JO Label
		Transfer execution control to address 'Label', if OF=1.
		6. JNO Label
		Transfer execution control to address 'Label', if OF=0.
		7. JNP Label
		Transfer execution control to address 'Label', if PF=0.
		8. JP Label
		Transfer execution control to address 'Label', if PF=1.
		9. JB Label
		Transfer execution control to address 'Label', if CF=1.
		10. JNB Label
		Transfer execution control to address 'Label', if CF=0.
		11. JCXZ Label
		Transfer execution control to address 'Label', if CX=0
		Conditional LOOP Instructions.
		12. LOOPZ / LOOPE Label
		Loop through a sequence of instructions from label while ZF=1 and CX=0.
		13. LOOPNZ / LOOPENE Label
		Loop through a sequence of instructions from label while ZF=1 and CX=0.
		
		EXAMPLE OF JC AND LOOP Instruction:
		MOV CX,08H
		MOV AL,05H
		UP:ROR AL,1
		JC DN
		INC BL
		DN:**LOOP UP**	

b)	**State the function of following registers of 8086 microprocessor :**	**4M**
Ans:	**(i) <u>General Purpose Registers of 8086</u>** 1. AX (Accumulator) – Used to store the result for arithmetic / logical operations All I/O data transfer using IN & OUT instructions use "A" register(AH / AL or AX). 2. BX – Base – used to hold the offset address or data in indirect addressing mode. 3. CX – acts as a counter for repeating or looping instructions. 4. DX –Used with AX to hold 32 bit values during multiplication and division. Used to hold address of I/O port in indirect addressing mode. 5. BP – Base Pointer BP can hold offset address of any location in the stack segment. It is used to access random locations of stack. 6. SP –Stack Pointer – Contains the offset of the top of the stack. SP is used with SS register to calculate 20-bit physical address. Used during instructions like PUSH,POP,CALL,RET etc. 7. SI – Source Index – Used in string movement instructions. Holds offset address of source data in Data segment during string operations. Used to hold offset address of data segment. 8.DI – Destination Index – acts as the destination for string movement instructions Used to hold offset address of Extra segment.	**(Any 4 General Purpose Register :1/2 M each)**

	9. IP – Instruction Pointer – Contains 16 bit offset address of instruction that is to Be executed in code segment **(ii) Segment Register:** 1. CS – Code Segment – holds base address for all executable instructions in a program 2. SS - Stack Segment- holds the Base address of the stack **3.** DS – Data Segment – default base address for variables. **4.** ES – Extra Segment – additional base address for memory variables in extra segment	**(4 Segment Register :1/2 M each)**
c)	**Draw the interfacing of 8284 clock generator with 8086 microprocessor.**	**4M**
Ans:		**Correct Diagram :4 M (Any other diagram also considered)**
d)	**List and explain any four process control instruction with their function.**	**4M**
Ans:	**List of process control instructions:** 1. CLC – Clear carry flag 2. CMC- Complement carry flag 3. STC- Set carry flag 4. CLD- Clear direction flag 5. STD- Set direction flag 6. CLI- Clear interrupt flag 7. STI- Set Interrupt flag 8. WAIT- wait for test input pin to go low 9. HLT- Halt the processor 10. NOP- No operation 11. ESC- Escape to external device like NDP (numeric co-processor) 12. LOCK- Bus lock instruction prefix. **Explanation:** 1. CLC – This instruction Clear Carry Flag. CF← 0 2. CMC – This instruction Complement Carry Flag. CF←~CF 3. STC – This instruction Set Carry Flag. CF←1 4. CLD – This instruction Clear Direction Flag. DF←0	**(List any 4 instructions : 2M, Explanation of any 4 instructions : 2M)**

5. STD – This instruction Set Direction Flag. DF←1
6. CLI – This instruction Clear Interrupt Flag. IF←0
7. STI – This instruction Set Interrupt Flag. IF←1

8)HLT

- This instruction causes processor to enter the halt state.
- CPU stop fetching and executing instructions.

9)NOP

- Used to add wait state of three clock cycles and during these clock cycles CPU does not perform any operation. .
- This instruction is Used to add delay loop in program

10)WAIT

- It causes processor to enter into an idle state or a wait state and continue to remain in that the processor receives state until one of the following signal.
  - Signal on processor TEST pin
  - Valid interrupt on INTR
  - Valid interrupt on NMI
- Used to synchronize other external hardware such as math co-processor.

11**)LOCK**

- Prevent other processor to take the control of shared resources.
- Lock the bus attached to lock pin of device while a multicycle instruction completes.
- The lock prefix this allows a microprocessor to make sure that another processor does not take control of system bus while it is in the middle of a critical instruction.

12)ESC:

- This instruction is used to pass instructions to a coprocessor, such as the 8087 Math coprocessor, which shares the address and data bus with 8086. Instructions for the coprocessor are represented by a 6-bit code embedded in the ESC instruction.

e)	**Write an assembly language program to add two BCD numbers.**	**4M**

Ans:	DATA SEGMENT
	NUM1 DB 09H
	NUM2 DB 09H
	SUM DB ?
	DATA ENDS
	CODE SEGMENT
	START: ASSUME CS:CODE,DS:DATA
	MOV AX,DATA
	MOV DS,AX
	MOV AL,NUM1
	ADD AL,NUM2
	DAA ;Decimal adjust for addition
	MOV SUM,AL
	MOV AH,4CH
	INT 21H
	CODE ENDS
	END START
		Correct Program :4 M (Any Other logic can be considered)

(OR)

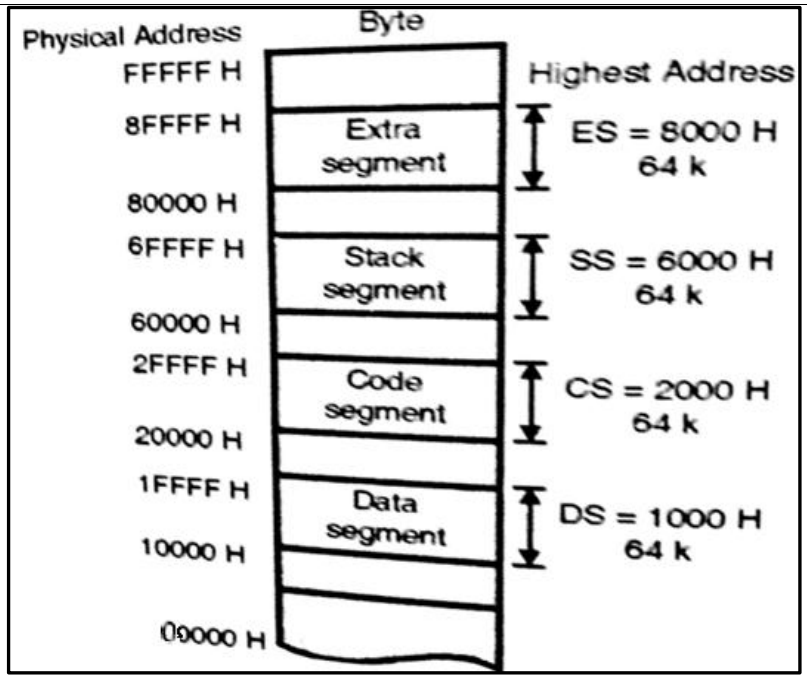
```
.MODEL SMALL
.DATA
NUM1 DB 84H
NUM2 DB 28H
RES_LSB DB ?
RES_MSB DB ?

.CODE
      MOV AX,@DATA
      MOV DS,AX

      MOV AL,NUM1                ;
      MOV BL,NUM2
      ADD AL,BL    ;Ans ACH
      DAA
      JNC DN
      INC RES_MSB

   DN:MOV RES_LSB,AL
      MOV AH,4CH
      INT 21H
     END
```

f)	**Explain concept of segmentation with diagram.**	**4M**
Ans:	**Memory Segmentation**: The memory in an 8086 microprocessor is organized as a segmented memory. The physical memory is divided into 4 segments namely,- Data segment, Code Segment, Stack Segment and Extra Segment. **Description**: • Data segment is used to hold data, Code segment for the executable program, Extra segment also holds data specifically in strings and stack segment is used to store stack data. • Each segment is 64Kbytes & addressed by one segment register. i.e CS,DS,ES or SS • The 16 bit segment register holds the starting address of the segment • The offset address to this segment address is specified as a 16-bit displacement (offset) between 0000 to FFFFH. Hence maximum size of any segment is 2^{16}=64K locations. • Since the memory size of 8086 is 1Mbytes, total 16 segments are possible with each having 64Kbytes. • The offset address values are from 0000H to FFFFH so the physical address range from 00000H to FFFFFH.	**(Explanation: 2M, Diagram : 2 M)**

			
Q. 4		**Attempt any FOUR of the following :**	**16M**
	a)	**Identify the addressing modes for following instructions:** **(i) MOV AX, 2034H** **(ii) MOV AL,[6000H]** **(iii) ADD AL, CL** **(iv) MOV AX, 50H [BX] [SI]**	**4M**
	Ans:	(i) MOV AX, 2034H : Immediate addressing mode (ii) MOV AL,[6000H] :Direct addressing mode (iii) ADD AL, CL :Resister addressing mode (iv) MOV AX, 50H [BX][SI] : Relative Base Index addressing mode	**Correct Addressing Mode 1M each**
	b)	**Explain the following instruction of 8086 :** **(i) XLAT** **(ii) XCHG**	**4M**
	Ans:	**(i)XLAT** • XLAT replaces a byte in AL register with a byte from 256 byte lookup table beginning at [BX] . • AL is used as offset into this table. • Flags are not affected • operation :- AL←[BX+AL] Example : data segment Table db '0123456789ABCDEF' CODE DB 11 data ends Code segment - - - MOV BX,offset Table	**(Each Instruction Correct operation : 2M)**

		MOV al,CODE XLAT ;AL will output code 0BH Code ends **(ii) XCHG Destination, Source** • This instruction exchanges the contents of a register with the contents of another register or memory location. • **Operation performed:** Destination ⟷ Source None of flag affected **Example:** XCHG BL, CL ; Exchange the byte in BL with byte in CL.	
c)	**Write an ALP to count of zero's in BL register.**		**4M**

Ans:	DATA SEGMENT
	NUM DB 0F3H ;BINARY{ 1111 0011}
	ZEROS DB 0
	DATA ENDS
	CODE SEGMENT
	START: ASSUME CS:CODE,DS:DATA
	MOV AX,DATA
	MOV DS,AX
	
	MOV CX,8 ;rotation counter
	MOV BL,NUM
	UP:
	ROR BL,1 ; RCR,ROL , RCL can be used
	JC DN ;IF CARRY loop
	INC ZEROS ; else increment 0's count ;ANSWER 02
	DN:LOOP UP ;decrement rotation counter
	
	EXIT: MOV AH,4CH
	INT 21H
	CODE ENDS
	END START
			Correct Program :4M (Any Other logic can be considered)

d)	**Write an ALP to subtract two 8 bit numbers.**		**4M**

Ans:	DATA SEGMENT
	NUM1 DB 10H
	NUM2 DB 20H
	DIFFDB ?
	DATA ENDS
	CODE SEGMENT
	START: ASSUME CS:CODE,DS:DATA
	MOV AX,DATA
	MOV DS,AX
	
	MOV AL,NUM1
			Correct Program :4M (Any other logic also considered)

```
        MOV BL,NUM2
        SUB AL,BL
        MOV DIFF,AL

        MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

(OR)

```
DATA SEGMENT
    NUM1    DB 85H
    NUM2    DB 92H
    DIFFERENCE   DB  1 DUP(0)
DATA ENDS
CODE SEGMENT
      ASSUME CS:CODE,DS:DATA
      START:
      MOV DX,DATA
      MOV DS,DX
      MOV AL,NUM1
      MOV BL,NUM2
      SUB AL,BL
      MOV DIFFERENCE,AL
      JNC EXIT
      MOV DIFFERENCE+1,01
  EXIT:MOV AH,4CH
      INT 21H
CODE ENDS
END START
```

e) Write an ALP to add two 16 bit numbers. — **4M**

Ans:

```
DATA SEGMENT
    NUMBER1    DW 5522 H
    NUMBER2    DW  3311H
    SUM   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
      ADD AX,BX
      MOV SUM,AX

MOV AH,4CH
```

Correct Program :4M (Any other logic also considered)

```
      INT 21H
CODE ENDS
END START
```

(OR)

```
DATA SEGMENT
    NUMBER1   DW 5522 H
    NUMBER2   DW  8311H
    SUM   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
      ADD AX,BX
      MOV SUM,AX
      JNC EXIT  ;EXIT IF CARRY
      MOV SUM+2,01  ;STORE CARRY BIT IN MS DIGIT

   EXIT:MOV AH,4CH
      INT 21H
CODE ENDS
END START
```

	f)	**Define MACRO with its syntax. Also give two advantages of it.**	**4M**
	Ans:	**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. **1) Macro Syntax:** Macro_name MACRO[arg1,arg2,…..argN) …… …… ENDM **Advantages of Macro:** • The speed of the execution of the program is increased. • It saves a lot of time that is spent by the compiler for invoking / calling the functions. • It reduces the length of the program.	**(Definition : 1M)** **(Syntax :1 M)** **(Any 2 advantages : 2M)**
Q.5		**Attempt any FOUR of the following ;**	**16M**
	a)	**Write an ALP to find sum of series 0BH, 05H, 07H, 0AH,01H.**	**4M**
	Ans:	**DATA SEGMENT** NUM1 DB 0BH,05H,07H,0AH,01H	**(Correct program- 4**

		RESULT DB 1 DUP(0)
		CARRY DB 0H
		DATA ENDS
		CODE SEGMENT
		START:ASSUME CS:CODE,DS:DATA
		MOV DX,DATA
		MOV DS,DX
		MOV CL,05H
		MOV SI,OFFSET NUM1
		UP:MOV AL,[SI]
		ADD RESULT,AL ;Answer : AL : 22H
		JNC NEXT
		INC CARRY
		NEXT:INC SI
		LOOP UP
		MOV AX,4C00H
		INT 21H
		CODE ENDS
		END START
			M, Any other logic may be considered)

	b)	**Write ALP to compute, whether the number in BL register is even or odd.**	**4M**

	Ans:	DATA SEGMENT
		NUM DB 9H
		ODD DB 0
		EVEN_NO DB 0
		DATA ENDS
		CODE SEGMENT
		START: ASSUME CS:CODE,DS:DATA
		MOV AX,DATA
		MOV DS,AX
		MOV BL,NUM
		ROR BL,1 ;or RCR
		JNC DN ; check ENEN or ODD
		ROL BL,1 ; restore number
		MOV ODD,BL ; odd
		JMP EXIT
		DN: ROL BL,1
		MOV EVEN_NO,BL ; even no
		
		EXIT: MOV AH,4CH
		INT 21H
		CODE ENDS
		END START
			(Correct program-4M, Any other logic may be considered)

	c)	**Write an ALP to reverse the string.**	**4M**
	Ans:	DATA SEGMENT STRING DB 'GOOD MORNING$' REV DB 0FH DUP(?) DATA ENDS CODE SEGMENT	**(Correct program-4M, Any other logic may be**

		START:ASSUME CS:CODE,DS:DATA
		MOV DX,DATA
		MOV DS,DX
		LEA SI,STRING
		MOV CL,0FH
		LEA DI,REV
		ADD DI,0FH
		UP:MOV AL,[SI]
		MOV [DI],AL
		INC SI
		DEC DI
		LOOP UP
		MOV AH,4CH
		INT 21H
		CODE ENDS
		END START
			considered)

	d)	**Write an appropriate 8086 instruction to perform following operation** **(i) Initialize stock of 4200H** **(ii) Multiply AL by 05H**	**4M**

	Ans:	**I)** MOV AX,,4200h
		MOV SS,AX
		MOV SP,0000H
		
		II) MOV BL,05h
		MUL BL
			(Instruction to initialize stack segment 1 M, instruction to initialize stack pointer 1 M)

	e)	**Explain NEAR and FAR procedure.**	**4M**
	Ans:		**(Any 4 points , 1M 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

f) Explain the directives used for defining MACRO. Give an example. **4M**

Ans:

1)Macro definition or (Macro directive):
The directive MACRO informs the assembler the beginning of MACRO.
It consist of name of macro followed by keyword MACRO and MACRO arguments if any.

Syntax:

```
Macro_name      MACRO[arg1,arg2,…..argN)
        …..
 Endm
```

2)ENDM Directive :END OF MACRO
The directive ENDM informs the assembler the end of macro.
Syntax: ENDM

3)LOCAL

- Macros are expanded directly in code, therefore if there are labels inside the macro definition you may get "Duplicate declaration" error when macro is used for twice or more. To avoid such problem, use **LOCAL** directive followed by names of variables, labels or procedure names.

Syntax: LOCAL <label>

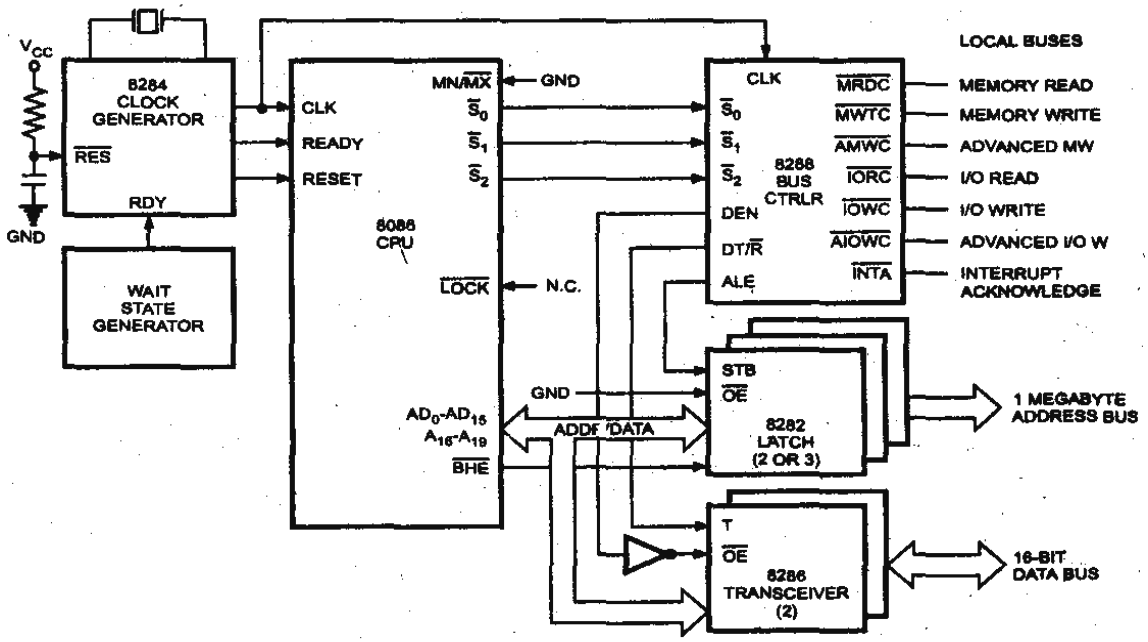
Example with MACRO ,ENDM,LOCAL Directive

```
MyMacro2    MACRO
            LOCAL label1, label2
            CMP  AX, 2
            JE label1
            CMP  AX, 3
            JE label2
            label1:
                        INC  AX
            label2:
                        ADD  AX, 2
ENDM
data segment
data ends
code segment
start:    assume cs:code,ds:data
          mov ax,data
```

(Any 2-Directives 1M each – Example :2 M)

(Any other example also considered)

```
        mov ds,ax
                mov ax,02h
                MyMacro2
                MyMacro2
            mov ah,4ch
        int 21h
code ends
end start
```

Q.6		Attempt any two of the following	16M
	a)	Draw and explain the working of 8086 in maximum mode.	8M
	Ans:		(Diagram : 4M, Explanation : 4M) *Maximum mode pin description optional*

1.In the maximum mode, the 8086 is operated by strapping the MN/MX pin to ground.
2.In this mode, the processor derives the status signal S2, S1, S0.
3.Another chip called bus controller derives the control signal using this status information
4.In the maximum mode, there may be more than one microprocessor in the system configuration. The components in the system are same as in the minimum mode system.
5.The basic function of the bus controller chip IC8288, is to derive control signals like RD and WR (for memory and I/O devices), DEN, DT/R, ALE etc. using the information by the processor on the status lines.
6.The bus controller chip has input lines S2, S1, S0 and CLK. These inputs to 8288 are driven by CPU.
It derives the outputs ALE, DEN, DT/R, MRDC, MWTC, AMWC, IORC, IOWC and AIOWC. The AEN, IOB and CEN pins are specially useful for multiprocessor systems. AEN and IOB are generally grounded. CEN pin is usually tied to +5V.
7.The significance of the MCE/PDEN output depends upon the status of the IOB pin.
8.INTA pin used to issue two interrupt acknowledge pulses to the interrupt controller or to an interrupting device.
9.IORC, IOWC are I/O read command and I/O write command signals respectively . These signals enable an IO interface to read or write the data from or to the address port.
10.The MRDC, MWTC are memory read command and memory write command signals respectively and may be used as memory read or write signals.

11.All these command signals instructs the memory to accept or send data from or to the bus.

Pin Discription:

AD15-AD0:-These pins acts as multiplexed address and data bus of the microprocessor. Whenever the ALE pin is high these pins carry the address, when the ALE pin is low it carry the data.

A19/S6-A16/S3:- These pins are multiplexed to provide the address signals A19-A16 and the status bits S6-S3. When ALE=1 these pins carry the address and when ALE=0, they carry the status lines.

S4	S3	Segment Accessed
0	**0**	Extra Segment
0	**1**	Stack Segment
1	**0**	Code Segment
1	**1**	Data Segment

NMI: - The non-mask able interrupt input is a hardware interrupt. It can not be disable by software.

INTR: - The interrupt request is a level-triggered hardware interrupt, which depends on the status of IF. When IF=1, INTR is held high then 8086 get interrupted. When IF=0, INTR is disabled.

CLK: The clock signal must have a duty cycle of 33% to provide proper internal timing for the 8086. Its maximum frequency can be 5, 8 and 10 MHz for different version of microprocessor.

Vcc: This is power supply pin and provide +5V signals to 8086.

$\overline{BHE}$/S7: The bus high enable pin used in 8086 to enable the most significant data bus (D15-D8) during a read/write operation. The state of the states line S7 is always logic 1 or high.

MN/$\overline{MX}$: The MN/$\overline{MX}$ pin is used to select either the minimum mode or maximum mode operation of the 8086. This is achieved by connecting this pin to either +5V directly (for minimum mode) or to the ground (for maximum mode).

$\overline{RD}$: Whenever the Read Signals is at logic o, the 8086 reads the data from the memory or I/O device through the data bus.

$\overline{TEST}$: The $\overline{TEST}$ pin is an input that is tested by the WAIT instruction. If the $\overline{TEST}$ pin is at logic 0, the WAIT instruction functions as NOP instruction. If the $\overline{TEST}$ pin is at logic 1, the WAIT instruction waits for the $\overline{TEST}$ pin to become logic 0.

READY: This input is used to insert wait state into the timing cycle of the 8086. If the ready pin is at logic 1, it has no effect on the operation of the microprocessor. If it is logic 0, the 8086 enters the waits state and remains the idle. This pin is used to interface the operating peripherals with the 8086.

RESET: This input is used to reset the 8086.

GND: The 8086 has two GND pins and both must connected to ground for proper operations.

The pins that have a function in maximum mode are as given follows.

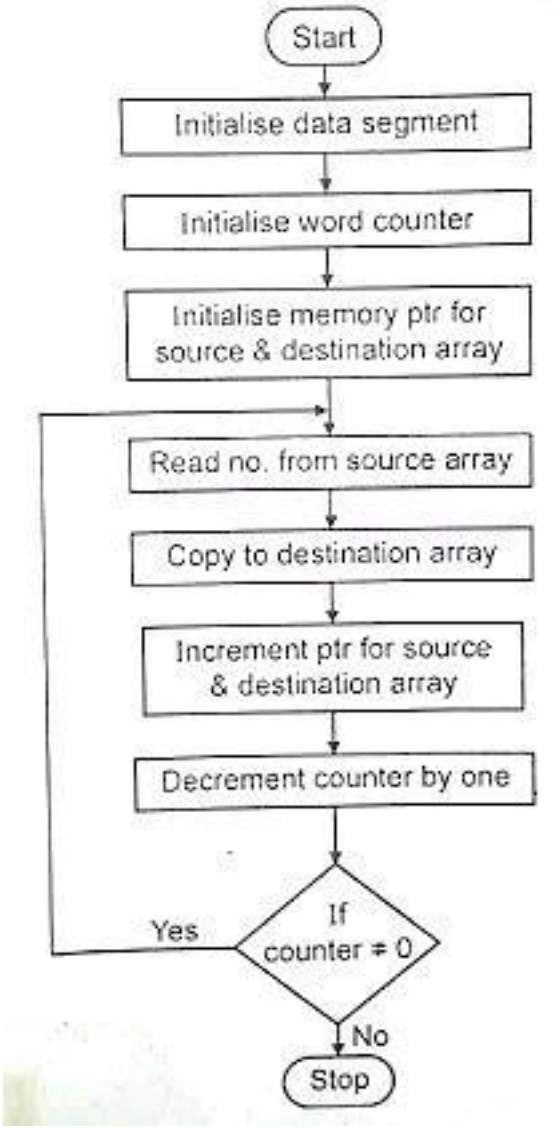
$\overline{S2}$, $\overline{S1}$& $\overline{S0}$: The states bits indicate the function of current cycle. These signals are normally decoded by the 8288.

		S2 S1 S0 Function 0 0 0 INTR 0 0 1 I/O Read 0 1 0 I/O Write 0 1 1 Halt 1 0 0 Op-code Fetch 1 0 1 Memory Read 1 1 0 Memory Write 1 1 1 Passive $\overline{LOCK}$: The $\overline{LOCK}$ output is used to look peripheral off the system. This pin is activated by using $\overline{LOCK}$ prefix on any instruction. $\overline{RQ}/\overline{GT0}$&$\overline{RQ}/\overline{GT1}$:- The request/grant pins request DMA during the maximum mode operations of 8086. These lines are bi-directional and are used to request and grant a DMA operation. QS1 & QS0:- The queue states bit show the states of the internal instruction queue in 8086.	
b)	**Write an ALP to transfer 10 bytes of data from one memory location to another Also draw the flow chart for the same.**	**8M**

Ans:	DATA SEGMENT
	block1 db 10 dup(10h)
	block2 db 10 dup(0)
	DATA ENDS
	CODE SEGMENT
	ASSUME CS:CODE,DS:DATA ,ES: EXTRA
	START:MOV DX,DATA ;initialize data seg
	MOV DS,DX
	MOV DX, EXTRA
	MOV ES,DX
	
	LEA SI,BLOCK1
	LEA DI,BLOCK2
	MOV CX,000AH
	CLD
	REP MOVSB
	MOV AH,4CH
	INT 21H
	CODE ENDS
	END START
	
	(OR)
	
	DATA SEGMENT
	block1 db 10 dup(10h)
	block2 db 10 dup(0)
	DATA ENDS
		(Correct Program : 5M, Flowchart : 3M)

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV DX,DATA   ;initialize data seg
      MOV DS,DX
      MOV ES,DX

      LEA SI,BLOCK1
      LEA DI,BLOCK2
      MOV CX,000AH
      CLD
   BACK:MOV AL,[SI]              ;     REP MOVSB
       MOV [DI],AL
       INC SI
       INC DI
       DEC CX
       JNZ BACK
MOV AH,4CH
  INT 21H
 CODE ENDS
END START
```



	c)	**(i) Write an ALP to find smallest number using procedure.** **(ii) Write a procedure to find the factorial.**	**8M**
	Ans:	**<u>i) An ALP to find smallest number using procedure.</u>** **DATA SEGMENT** **ARRAY DB 15H,45H,08H,56H,78H** **SMALLEST DB 00H** **DATA ENDS** **CODE SEGMENT** **START:ASSUME CS:CODE,DS:DATA** **MOV DX,DATA** **MOV DS,DX** CALL SMALLEST_NO **MOV AX,4C00H** **INT 21H**	**(Correct Program : 4M)**

```
    SMALLEST_NO PROC
      MOV CX,04H
      MOV SI ,OFFSET ARRAY
      MOV AL,[SI]
      UP:INC SI
      CMP AL,[SI]
      JC NEXT
      MOV AL,[SI]
   NEXT:DEC CX
      JNZ UP
      MOV SMALLEST,AL   ;AL=08H
      RET
     SMALLEST_NO ENDP

CODE ENDS
END START
```

ii) A procedure to find the factorial.

```
DATA SEGMENT
      NUM DB 04H
DATA ENDS

CODE SEGMENT
      START: ASSUME CS:CODE, DS:DATA
      MOV AX,DATA
      MOV DS,AX
                  CALL FACTORIAL
      MOV AH,4CH
      INT 21H

      PROC FACTORIAL
      MOV BL,NUM    ; TAKE NO IN BL REGISTER
      MOV CL,BL      ;TAKE CL AS COUNTER

      DEC CL  ;DECREMENT CL BY 1
      MOV AL,BL
     UP: DEC BL       ;DECREMENT BL TO GET N-1
     MUL BL  ;MULTIPLY CONTENT OF N BY N-1
     DEC CL            ;DECREMENT COUNTER
     JNZ UP            ;REPEAT TILL ZERO
     RET
      FACTORIAL ENDP
CODE ENDS
END START
```

(OR)

```
DATA SEGMENT
```

(Correct Program : 4M)

```
        A DW 0005H
        FACT_LSB DW ?
        FACT_MSB DW ?
DATA ENDS
CODE SEGMENT
        ASSUME DS:DATA,CS:CODE
START:MOV AX,DATA
      MOV DS,AX
     CALL FACTORIAL
      MOV AH,4CH
      INT 21H
       FACTORIAL PROC
       MOV AX,A
       MOV BX,AX
       DEC BX
  UP: MUL BX        ; MULTIPLY AX * BX
       MOV FACT_LSB,AX    ;ANS DX:AX PAIR
       MOV FACT_MSB,DX
       DEC BX
       CMP BX,0
       JNZ UP
       RET
       FACTORIAL ENDP
CODE ENDS
END START
```