SUMMER – 2019 EXAMINATION
MODEL ANSWER

Subject: Digital Techniques and Microprocessor **Subject Code: 22323**

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 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.

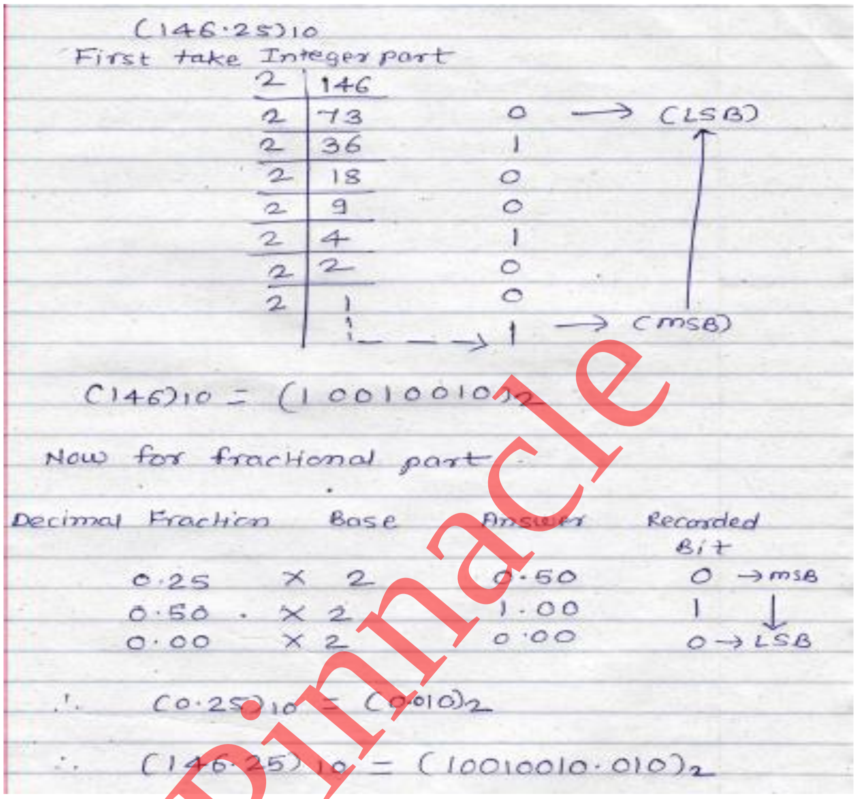
Q. No.	Sub Q.N.	Answer	Marking Scheme
1.	 **a)** **Ans.**	**Attempt any FIVE of the following:** **State the function of linker and debugger.** **Function of linker and debugger:** **Linker:** There are certain programs which are large in size and cannot be executed at one go simultaneously. Such programs are divided into sub programs also known as modules. The linker is used to link such small programs to form one large program. It also generates an executable file. **Debugger:** Debugger is used to test and debug programs. The debugger allows a user to test a program step by step, so that the problem points or steps can be identified and rectified. It allows the user to inspect the registers and memory locations after a program has been executed.	**10** **2M** ***Each function 1M***
	b) **Ans.**	**List any four addressing modes and give one example of each.** **Addressing Modes:** 1. Immediate Addressing Mode: Example: MOV CL, 03H ADD AX, 1234H	**2M**

		2. Register Addressing Mode: Example: MOV AL, BL ADD CL, DL MOV DS, AX 3. Direct Addressing Mode: Example: MOV AL, [2000H] MOV [1020], 5050H 4. Register Indirect Addressing Mode Example: MOV [DI], 1234H MOV AX, [BX] 5. Based Addressing with displacement Example: MOV AX, [BX+300H] MOV AX, [BX-2H] 6. Indexed Addressing Mode Example: MOV [DI + 2345H], 1234H MOV AX, [SI + 45H] 7. Based Indexed Addressing Mode Example: MOV [BX + DI], 1234H MOV AX, [SI + BX] 8. Based Indexed Addressing with Displacement Mode Example: MOV [DI + BX + 37H], AX MOV AL, [BX + SI + 278H] 9. Fixed or Direct Port Addressing: Example: OUT 06H, AL IN AX, 85H 10. Variable or Indirect Port Addressing Example: IN AL, DX OUT DX, AX 11. Implied (Implicit) Addressing Modes Example: CLC DAA	***Any four addressing modes with example $^{1/2}$M each***
	c) **Ans.**	**State any two Boolean laws with expression.** 1. A . 0 = 0 2. A . 1 = A And law 3. A . A = A 4. A . $\bar{A}$ = 0 5. Commutative Law A. B. = B. A. 6. Associative Law	**2M** ***Any 2 Boolean laws 1M each***

SUMMER – 2019 EXAMINATION
MODEL ANSWER

Subject: Digital Techniques and Microprocessor **Subject Code: 22323**

		A. (B.C) = (A.B)C 7. Distributive Law A.(B+C) = A.B + A.C. 8. A.(A+B) = A 9. A. ($\bar{A}$ + B) = AB 10. $\bar{\bar{A}}$ = A 11. De-Morgan's theorem $\overline{A.B} = \bar{A} + \bar{B}$ 12. A + 0 = A 13. A + 1 = 1 $\bar{A}$ + 1 =1 OR law 14. A + A = A 15. A + $\bar{A}$ = 1 16. A + B = B + A 17. A + (B + C) = (A + B) + C 18. A + (B. C) = (A + B) . (A + C) 19. A + AB = A 20. A + $\bar{A}$B = A + B 21. $\bar{A}$ + AB = $\bar{A}$+ B 22. $\bar{A}$ + A$\bar{B}$= $\bar{A}$ + $\bar{B}$ 23. $\overline{A + B} = \bar{A} . \bar{B}$	
	d) **Ans.**	**Define:** **i) Bit** **ii) Nibble** **i) Bit:** Bit is a Binary digit which is the smallest unit of data in digital systems. A bit has a single binary value, either 0 or 1. **ii) Nibble:** A group of 4 bits is referred as Nibble. Eg: 1011, 1001, 1100	**2M** ***Each definition 1M***
	e) **Ans.**	**Convert following number into its equivalent Binary Number $(146.25)_{10}$**	**2M**



$(146 \cdot 25)_{10}$

First take Integer part

2	146		
2	73	0	→ (LSB) ↑
2	36	1	
2	18	0	
2	9	0	
2	4	1	
2	2	0	
2	1	0	
	└ →	1	→ (MSB)

$(146)_{10} = (10010010)_2$ **1M**

Now for fractional part

Decimal Fraction	Base	Answer	Recorded Bit
0·25 ×	2	0·50	0 → MSB
0·50 ×	2	1·00	1 ↓
0·00 ×	2	0·00	0 → LSB

∴ $(0 \cdot 25)_{10} = (0010)_2$

∴ $(146 \cdot 25)_{10} = (10010010 \cdot 010)_2$ **1M**

f) Ans. **Define Minterm and Maxterm.** **2M**

Minterm:

Each individual term in the canonical SOP (Sum of Products) form is called as Minterm.

Example:

$$\text{Canonical SOP } Y = \underbrace{ABC} + \underbrace{A\bar{B}\bar{C}} + \underbrace{\bar{A}B\bar{C}}$$

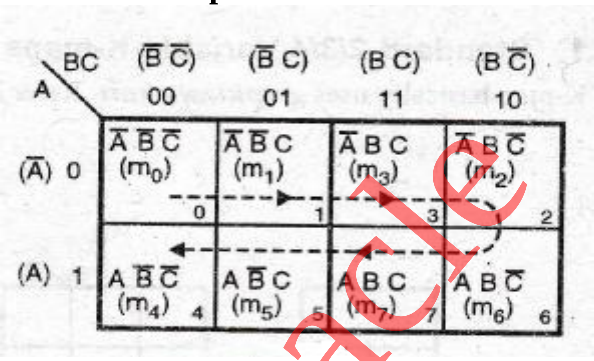
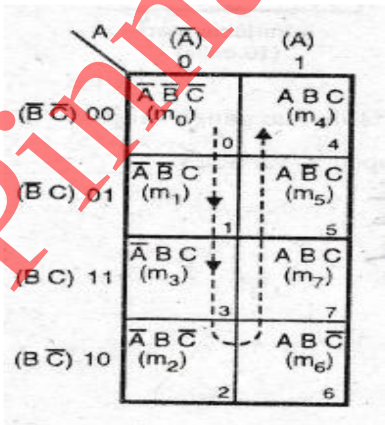
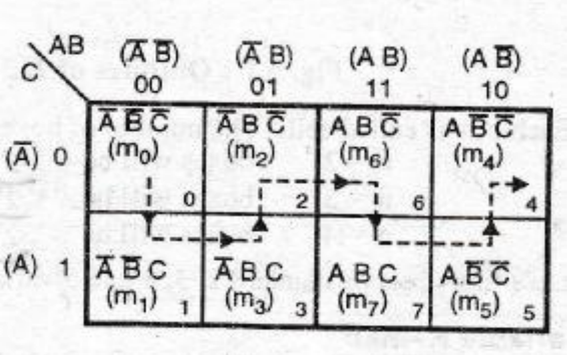
Each individual term is called minterm

Each definition 1M

Maxterm:

Each individual term in the canonical POS (Product of Sums) form is called as Maxterm.

Example:

		Canonical POS $Y = \underbrace{(A+B)}\cdot\underbrace{(A+\bar{B})}$ Each individual term is called maxterm	
	g) Ans.	**Draw three variable K-map format.**  **OR**  **OR** 	2M *Correct diagram 2M*

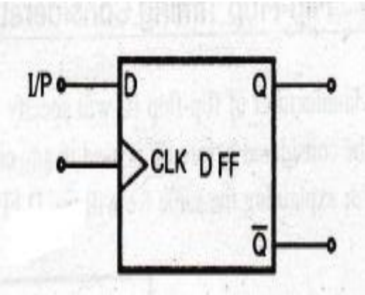
SUMMER – 2019 EXAMINATION
MODEL ANSWER

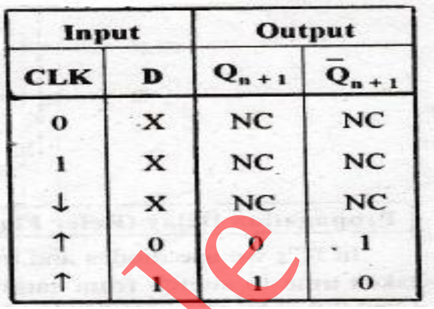
Subject: Digital Techniques and Microprocessor **Subject Code:** 22323

2 **Attempt any THREE of the following:** **12**

a) **Draw symbol and truth table of D and T flip flop. State theie applications.** **4M**

Ans. **D flip flop:**





Input		Output	
CLK	D	Q_{n+1}	$\bar{Q}_{n+1}$
0	X	NC	NC
1	X	NC	NC
↓	X	NC	NC
↑	0	0	1
↑	1	1	0

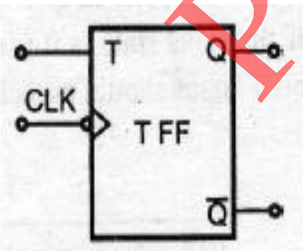
Symbol **Truth Table**

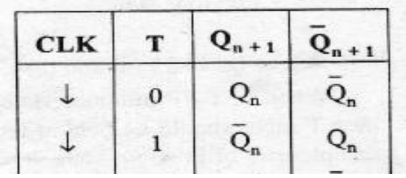
D flip flop Symbol - ½M; Truth table-1M; One application -½M

Applications of D flip flop:

1. used as a Latch
2. Divide - by - 4 Ripple Counter
3. Ring Counter
4. Johnson Counter
5. Used in registers

T flip flop:





CLK	T	Q_{n+1}	$\bar{Q}_{n+1}$
↓	0	Q_n	$\bar{Q}_n$
↓	1	$\bar{Q}_n$	Q_n

Symbol **Truth Table**

T flip flop Symbol - ½M; Truth table-1M; One application -½M

Applications of T flip flop:

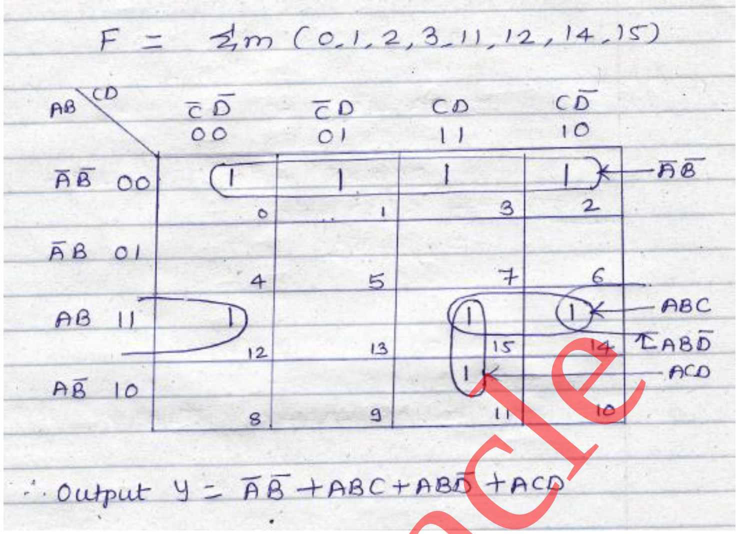
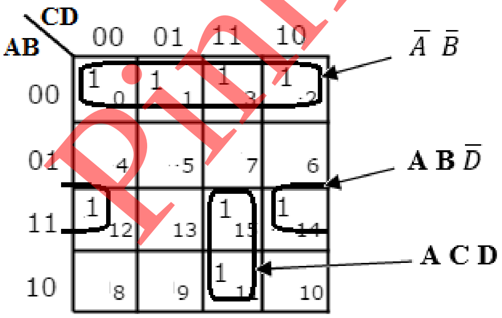
1. As the basic building block of counter.
2. In frequency divider circuits.
3. Used in D to A converter (DAC)

b) **Minimize the following function using K-map.** **4M**

F = Σm (0,1,2,3,11,12,14,15).

(Note: Any other equations shall be considered).

Ans.

		$F = \sum m(0,1,2,3,11,12,14,15)$  AB CD $\bar{C}\bar{D}$ 00 $\bar{C}D$ 01 CD 11 $C\bar{D}$ 10 $\bar{A}\bar{B}$ 00 $\bar{A}B$ 01 AB 11 $A\bar{B}$ 10 1 1 1 1 0 1 3 2 4 5 7 6 1 1 12 13 15 14 1 8 9 11 10 $\bar{A}\bar{B}$ ABC $AB\bar{D}$ ACD ∴ Output $Y = \bar{A}\bar{B} + ABC + AB\bar{D} + ACD$ **OR**  CD AB 00 01 11 10 00 01 11 10 1 1 1 1 0 1 3 2 4 5 7 6 1 1 1 12 13 15 14 1 8 9 11 10 $\bar{A}$ $\bar{B}$ A B $\bar{D}$ A C D Minimized Expression F = $\bar{A}\bar{B}$ + **A B$\bar{D}$** + **A C D**	*Four variable K-map 2M* *Final equation 2M*
	c) Ans.	**Perform binary subtraction using 2's complement of the following:** **i) $(63)_{10}$ - $(20)_{10}$ = ?** **ii) $(34)_{10}$ - $(48)_{10}$ = ?** **i) $(63)_{10}$ - $(20)_{10}$ = ?**	4M

i) $(63)_{10} - (20)_{10} = ?$

$\Rightarrow (63)_{10} - (20)_{10} = (63)_{10} + (-20)_{10}$

$(63)_{10} = (?)_2$

$$\begin{array}{r|l l}
2 & 63 & \\
\hline
2 & 31 & 1 \\
\hline
2 & 15 & 1 \\
\hline
2 & 7 & 1 \\
\hline
2 & 3 & 1 \\
\hline
2 & 1 & 1 \\
 & & \rightarrow 1
\end{array}$$

(LSB) ↑ (MSB)

$$\begin{array}{r|l l}
2 & 20 & \\
\hline
2 & 10 & 0 \\
\hline
2 & 5 & 0 \\
\hline
2 & 2 & 1 \\
\hline
2 & 1 & 0 \\
 & & \rightarrow 1
\end{array}$$

LSB ↑ MSB

$\therefore (63)_{10} = (111111)_2$ $\quad \therefore (20)_{10} = (010100)_2$

For finding 2's complement of $(20)_{10}$

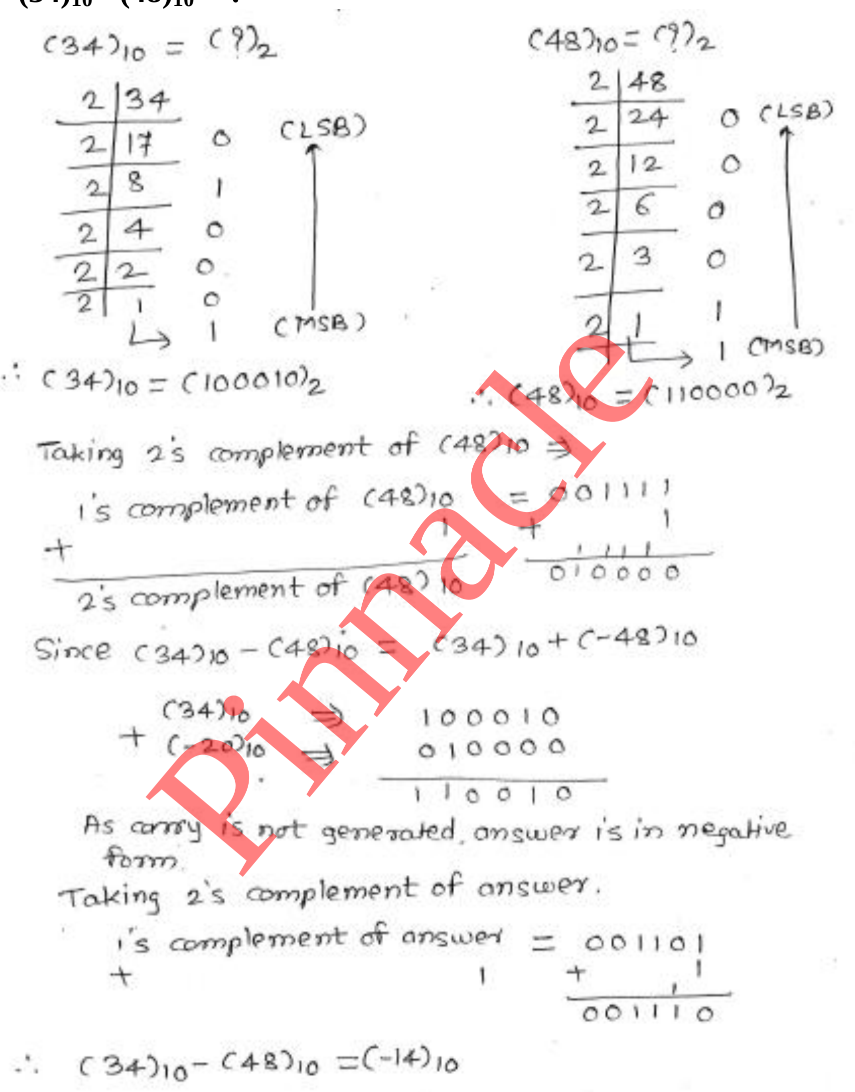
$$\begin{array}{lcr}
\text{1's complement of } (20)_{10} & & 101011 \\
+ \qquad\qquad\qquad 1 & \Rightarrow & + \qquad 1 \\
\hline
\text{2's complement of } (20)_{10} & & 101100
\end{array}$$

$$\begin{array}{lcr}
(63)_{10} & \Rightarrow & 111111 \\
+ \; (-20)_{10} & \Rightarrow & 101100 \\
\hline
 & & \boxed{1}\,101011 \Rightarrow (43)_{10}
\end{array}$$

Discard Carry

As carry is generated, result is positive and in its true form.

2M

		ii) $(34)_{10}$ - $(48)_{10}$ = ?  $(34)_{10} = (?)_2$ 2 | 34 2 | 17 0 (LSB) 2 | 8 1 2 | 4 0 2 | 2 0 2 | 1 0 ↳ 1 (MSB) $\therefore (34)_{10} = (100010)_2$ $(48)_{10} = (?)_2$ 2 | 48 2 | 24 0 (LSB) 2 | 12 0 2 | 6 0 2 | 3 0 1 2 | 1 ↳ 1 (MSB) $\therefore (48)_{10} = (110000)_2$ Taking 2's complement of $(48)_{10}$ ⇒ 1's complement of $(48)_{10}$ = 001111 + 1 + 1 2's complement of $(48)_{10}$ 010000 Since $(34)_{10} - (48)_{10} = (34)_{10} + (-48)_{10}$ $(34)_{10}$ ⇒ 100010 + $(-20)_{10}$ ⇒ 010000 110010 As carry is not generated, answer is in negative form. Taking 2's complement of answer. 1's complement of answer = 001101 + 1 + 1 001110 $\therefore (34)_{10} - (48)_{10} = (-14)_{10}$	*2M*
	d)	**Simplify the following Boolean expression** **i) $Y = AB + ABC + \overline{A}B + \overline{A}\overline{B}C$** **ii) $Y = (A + B)(A + \overline{B})(\overline{A} + B)$** **Note: Any other method of simplifying using the Boolean laws shall also be considered.**	**4M**
	Ans.	**i) $Y = AB + ABC + \overline{A}B + \overline{A}\overline{B}C$** $= AB(1 + C) + \overline{A}(B + \overline{B}C)$	

SUMMER – 2019 EXAMINATION
MODEL ANSWER

Subject: Digital Techniques and Microprocessor **Subject Code: 22323**

<table>
<tr><td></td><td></td><td>
= AB + $\bar{A}$(B + C) ∵ 1 + C = C, B + $\bar{B}$C = B + C
= AB + $\bar{A}$B + $\bar{A}$C
= B (A + $\bar{A}$) + $\bar{A}$C ∵ A + $\bar{A}$ = 1
= B (1) + $\bar{A}$C
= **B + $\bar{A}$C**

ii) Y = (A + B) (A + $\bar{B}$) ($\bar{A}$ + B)
=(A.A + A $\bar{B}$ + AB + B $\bar{B}$) ($\bar{A}$ + B)
= (A + A $\bar{B}$ + AB + 0) ($\bar{A}$+ B) (∵ A.A = A, B $\bar{B}$ =0)
= A (1 + $\bar{B}$ + B) ($\bar{A}$ + B)
= A (1) ($\bar{A}$ + B) (∵ B+$\bar{B}$=1, 1+A=1)
= A ($\bar{A}$ + B)
= A $\bar{A}$ + AB
= 0 + AB (∵ A $\bar{A}$ = 0)
= AB
</td><td>*2M*

2M</td></tr>
<tr><td>**3.**</td><td>**a)**

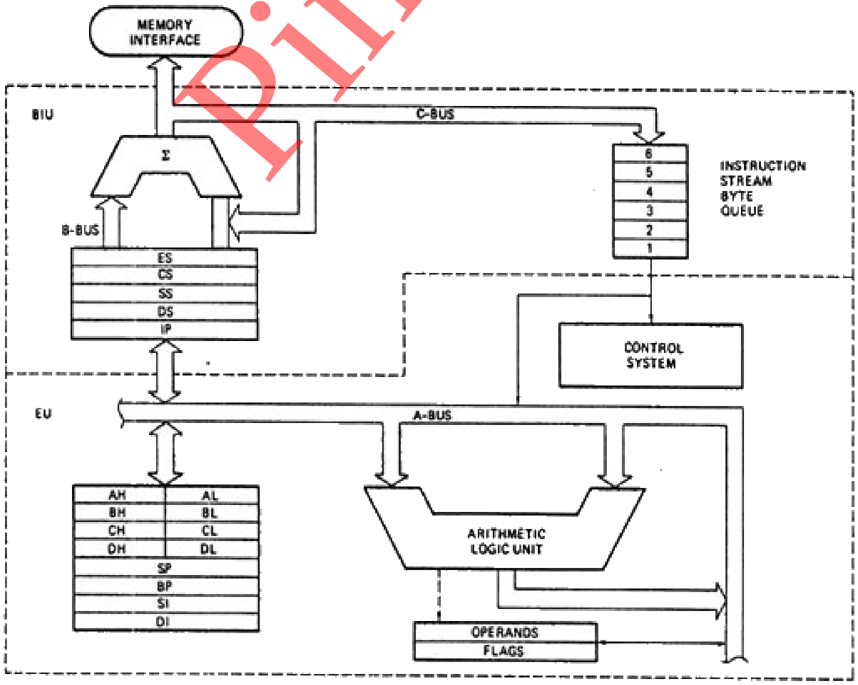
Ans.</td><td>
Attempt any <u>THREE</u> of the following:
Draw 8086 architecture block diagram and state the functions of EV and B/V.
(Note: EV and B/V are considered as EU and BIU).


Fig: Functional Block Diagram of Intel 8086 microprocessor
</td><td>**12**
4M

Diagram 2M</td></tr>
</table>

		BIU: It handles all transfers of data and addresses on the buses for the execution unit. • Sends out addresses • Fetches instructions from memory. • Read / write data from/to ports and memory i.e. handles all transfers of data and addresses on the busses **EU:** • Tells BIU where to fetch instructions or data from • Decodes instructions • Executes instructions **OR** **The functions performed by the Bus interface unit are:** - The BIU is responsible for the external bus operations. - It performs fetching, reading, writing for memory as well as I/O of data for peripheral devices. - The BIU also performs address generation and the population of the instruction queue. **The Execution unit is responsible for the following work:** - The instructions are decoded and executed by it. - The EU accepts instructions from the instruction queue and from the general purpose registers it takes data. - It has no relation with the system buses.	*1M for BIU* *1M for EU*
	b) **Ans.**	**Design half adder using K-map and realize it using basic gate.** **Half Adder:** Half adder is a combinational circuit that performs simple addition of two binary digits. **Half Adder Truth Table:** If we assume A and B as the two bits whose addition is to be performed, a truth table for half adder with **A, B** as inputs and **Sum, Carry** as outputs can be tabulated as follows.	**4M** *1M for Truth Table*

Truth Table			
Input		Output	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

1M each for K map of sum and carry

K map for sum

A\B	0	1
0	0	1
1	1	0

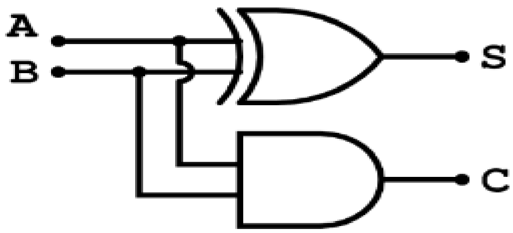
Sum= $A\overline{B}+ \overline{A}B$

K map for Carry

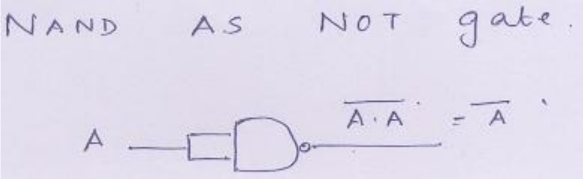
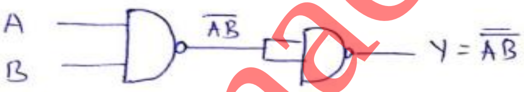
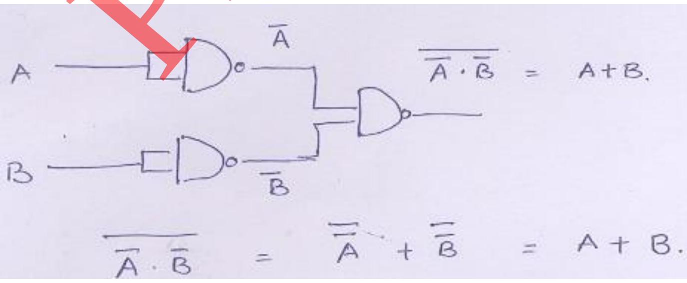
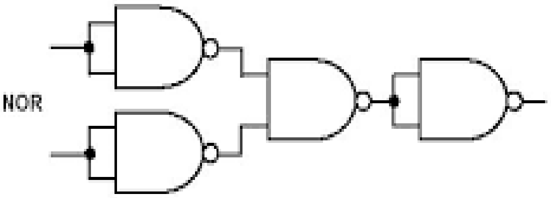
A\B	0	1
0	0	0
1	0	1

Carry=A.B

Logic Diagram for Half Adder:



1M for Logic Diagram

	c) Ans.	**Construct NOT, AND, OR, NOR gates using NAND gate.** **NAND as NOT gate:**  NAND AS NOT gate. A $\overline{A \cdot A} = \overline{A}$ **AND using NAND:** $Y = AB$ $Y = \overline{\overline{AB}}$ (∵ $\overline{\overline{A}} = A$) ∴ $\boxed{Y = AB}$  Fig:- AND gate using NAND **OR using NAND:**  A B $\overline{A}$ $\overline{B}$ $\overline{\overline{A} \cdot \overline{B}} = A+B.$ $\overline{\overline{A} \cdot \overline{B}} = \overline{\overline{A}} + \overline{\overline{B}} = A + B.$ **NOR using NAND:**  	4M *1M for each conversion*

d) **Interpret the given program and specify the output for the following situation.** **4M**

MOV AX, 34F9H
MOV BX, 3A69H.

(i) Masking of lower nibble of AX.
(ii) Rotate right through carry contents of BX by 4 positions.
(iii) Shift left contents of BX by 6 positions..
(iv) XOR AX, BX

(Note: If the outputs are written correctly according to the sequence also, marks shall be given. Weightage shall be given to the output need not consider the steps).

Ans. **(i) Masking of lower nibble of AX:**

AND AL,0F0H

After the execution of this instruction the content of AX register will be **34F0H.**

AX= 34F0H

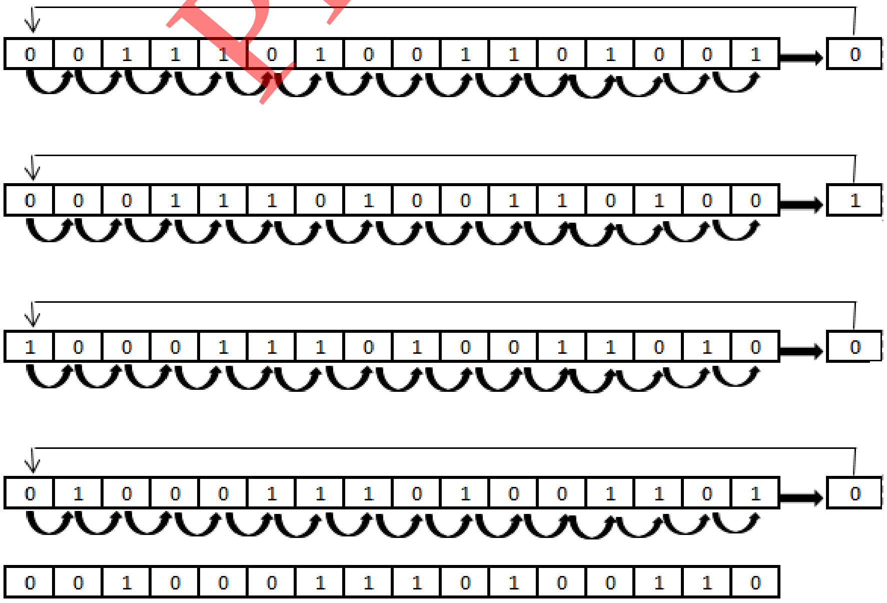
(ii) Rotate right through carry contents of BX by 4 positions:

The instruction will be

MOV CL,04H

RCR BL,CL

1M for the output of each instruction



0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 → 0

0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 → 1

1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 → 0

0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 → 0

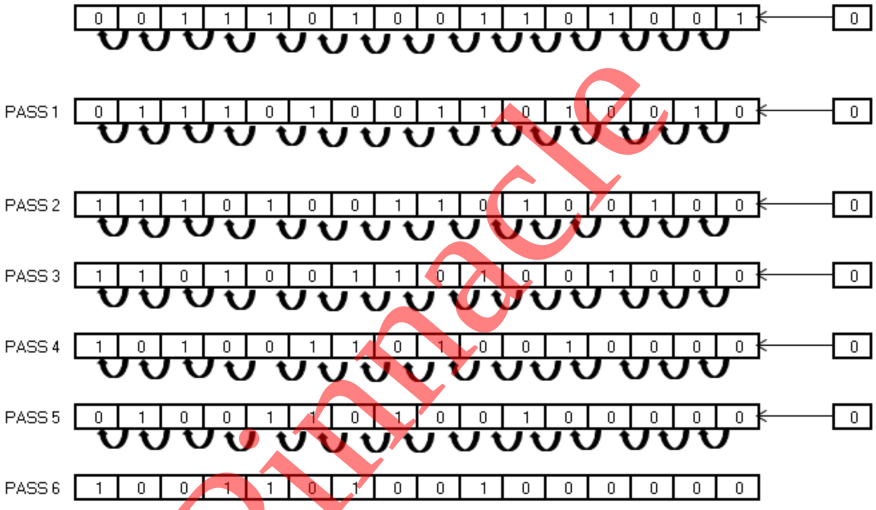
0 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0

After the Execution of the instruction the data will be **23A6H.**

BX= 23A6H

(iii) Shift left contents of BX by 6 positions:

Register BX is 3A69H, after shifting it by 6 positions, using SHL BX, CL instruction, where CL=06



	0	0	1	1	1	0	1	0	0	1	1	0	1	0	0	1	0
PASS 1	0	1	1	1	0	1	0	0	1	1	0	1	0	0	1	0	0
PASS 2	1	1	1	0	1	0	0	1	1	0	1	0	0	1	0	0	0
PASS 3	1	1	0	1	0	0	1	1	0	1	0	0	1	0	0	0	0
PASS 4	1	0	1	0	0	1	1	0	1	0	0	1	0	0	0	0	0
PASS 5	0	1	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0
PASS 6	1	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	

After the execution the content of regBx will be **9A40H**

BX= 9A40H

(iv) XOR AX, BX:

AX	34F9	0	0	1	1	0	1	0	0	1	1	1	1	1	0	0	1
BX	3A69	0	0	1	1	1	0	1	0	0	1	1	0	1	0	0	1
	XORing	0	0	0	0	1	1	1	0	1	0	0	1	0	0	0	0

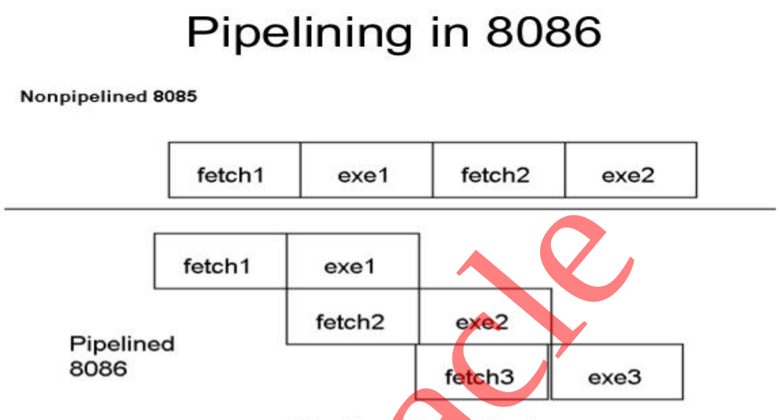
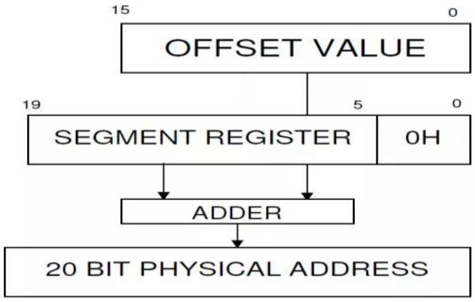
After the Execution of the instruction Register AX will contain data **0E90H**

AX= 0E90H

4. **Attempt any <u>THREE</u> of the following:** **12**

a) **Explain the concept of pipelining.** **4M**

Ans. In pipelined processor, fetch, decode and execute operation are performed simultaneously or in parallel. When first instruction is being decoded, same time code of the next instruction is fetched. ***Explanation 2M***

		• When first instruction is getting executed, second one's is decoded and third instruction code is fetched from memory. This process is known as pipelining. It improves speed of operation to great extent.  Pipelining in 8086 Nonpipelined 8085 fetch1 exe1 fetch2 exe2 fetch1 exe1 fetch2 exe2 Pipelined 8086 fetch3 exe3 Pipelined in 8086 microprocessor	*Diagram 2M*
	b) Ans.	**Explain concept of physical address calculation with suitable diagram and examples.** The 8086 addresses a segmented memory. The complete physical address which is 20-bits long is generated using segment and offset registers each of the size 16-bit.The content of a segment register also called as segment address, and content of an offset register also called as offset address. To get total physical address, put the lower nibble 0H to segment address and add offset address. The figure shows formation of 20-bit physical address.  15 0 OFFSET VALUE 19 5 0 SEGMENT REGISTER 0H ADDER 20 BIT PHYSICAL ADDRESS **Fig: Physical address formation** Calculate the physical address for the given CS=3420H, IP=689AH. CS=3420H	4M *2M for explaination* *1M diagram*

IP=689AH
Zero is inserted
3 4 2 0 0
+ 6 8 9 A
= 3 A A 9 A

1M for example

c) Ans.

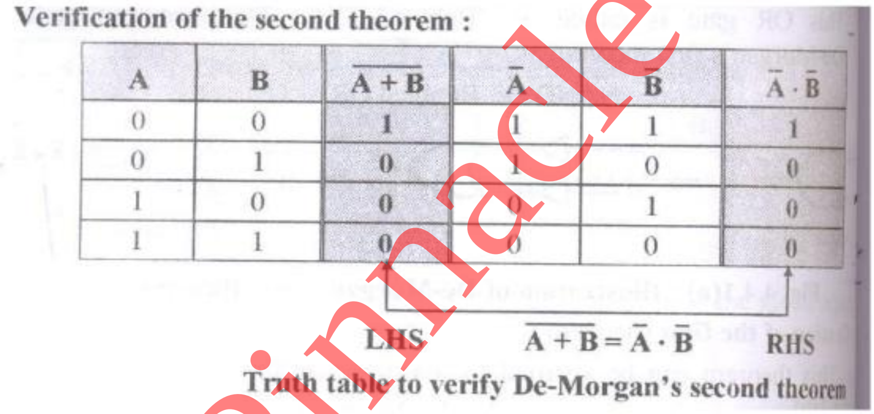
State and prove De-Morgan's Theorems. **4M**

Theorem no 1:

It states that the, complement of a sum is equal to product of their complements

For each theorem 2M

Verification of the second theorem :



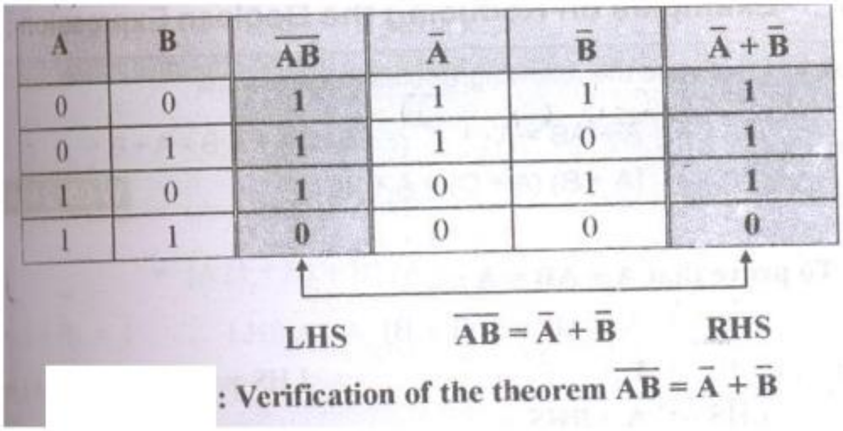
A	B	$\overline{A+B}$	$\bar{A}$	$\bar{B}$	$\bar{A}\cdot\bar{B}$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	0

LHS $\overline{A+B} = \bar{A}\cdot\bar{B}$ RHS

Truth table to verify De-Morgan's second theorem

Theorem no 2:

It states that, the complement of a product is equal to sum of the complements.



A	B	$\overline{AB}$	$\bar{A}$	$\bar{B}$	$\bar{A}+\bar{B}$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

LHS $\overline{AB} = \bar{A}+\bar{B}$ RHS

: Verification of the theorem $\overline{AB} = \bar{A}+\bar{B}$

d) **Describe race-around condition in JK flip flop and suggest ways to overcome it.** **4M**

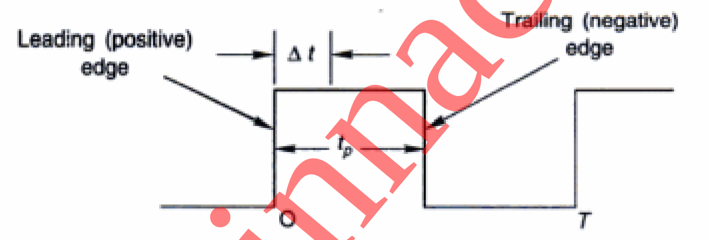
Ans. **Race around condition in JK flip-flop:**

In a J-K Flip-flop, when J=K=1, the output toggles.

If the clock pulse as shown below is applied at the clock input, for a level triggered J-K flip-flop, after a time interval Δt equal to the propagation delay through two NAND gates, the output again toggles. After another time interval Δt, the output changes again. Hence during t_p of the clock pulse, the output will oscillate back and forth between 0 and 1. At the end of the clock pulse, the value of Q is uncertain. This situation is referred as race -around condition.

2M for description

This can be avoided if $t_p < \Delta t < T$. A practical method of overcoming this difficulty is the use of the master-slave (MS) configuration. It can also be achieved through edge triggering.



2M for suggestion

e) **Compare combinational and sequential circuits (four points).** **4M**

Ans.

Sr. No.	Combinational circuits	Sequential circuits
1	Output depends on inputs present at that time	Output depends on present inputs and past inputs/ outputs
2	Memory is not necessary	Memory is necessary
3	Clock input is not necessary	Clock input is necessary
4	Design is simple	Design is complex
5	For e.g. Adders, Subtractors	For e.g. Shift registers, Counters

Any four points 1M each

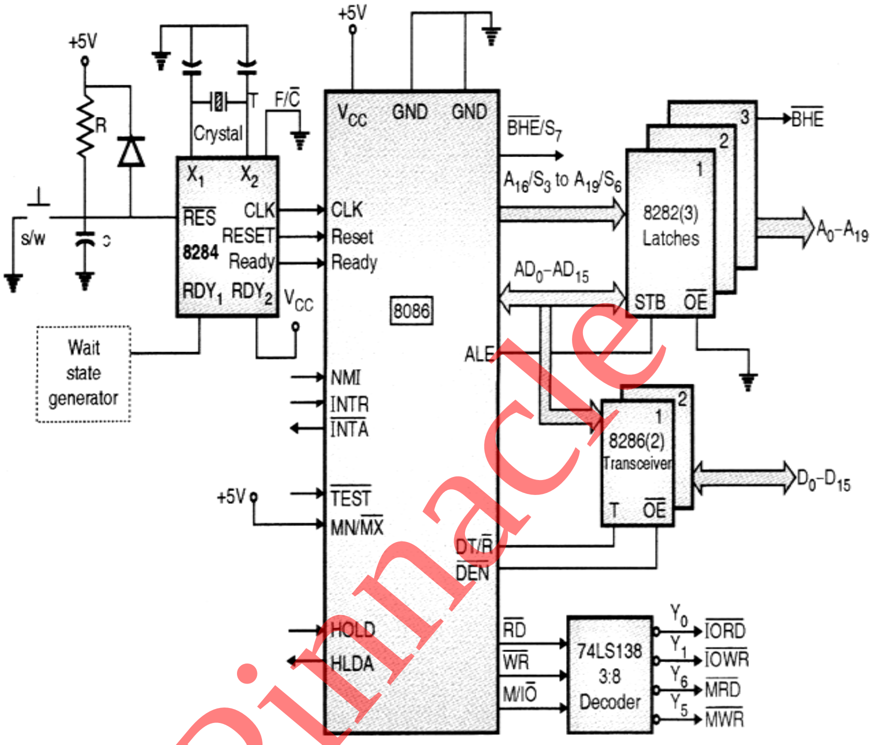
5. **Attempt any TWO of the following:** **12**

a) **Write an assembly language program with algorithm for finding smallest number from the array of 10 numbers (Assume suitable data).** **6M**

(Note: Any other logic shall be considered).

Ans.

		Algorithm: 1. Start 2. Load the array offset in BX 3. Initialize the CX with count value. 4. Initialize AL with FFh. 5. Compare the first number in BL with AL 6. Compare and transfer the smallest number in AL. 7. Decrement counter and if it is not zero then repeat the loop from step 5. 8. Store the smallest number in the defined destination location. 9. Stop the process.	***Algorithm 2M***
		Program: data segment STRING1 DB 08h,14h,05h,0Fh,09h, 01h, 05h, 18h, 2Ah, 0ACh res db ? data ends code segment assume cs:code, ds:data start: mov ax, data mov ds, ax mov al, 0ffh mov cx, 0ah mov bx, offset STRING1 again: cmp al, [bx] jc skip mov al, [bx] skip: inc bx loop again mov res, al int 3 code ends end start	***Correct Program 4M***
	b) **Ans.**	**Draw minimum mode configuration of 8086 and explain the function of any four control signals.**	**6M**

		 +5V R s/w C Crystal T F/C̄ X1 X2 RES CLK RESET 8284 Ready RDY1 RDY2 VCC Wait state generator +5V VCC GND GND CLK Reset Ready 8086 NMI INTR INTA +5V TEST MN/MX HOLD HLDA BHE/S7 A16/S3 to A19/S6 AD0–AD15 ALE DT/R DEN RD WR M/IO 8282(3) Latches 1 2 3 STB OE BHE A0–A19 8286(2) Transceiver 1 2 T OE D0–D15 74LS138 3:8 Decoder Y0 Y1 Y6 Y5 IORD IOWR MRD MWR	*Diagram 4M*
		1. **$\overline{\text{INTA}}$**: This is related to the non-vectored interrupt. It indicates that the processor has accepted INTR interrupt. 2. **ALE:** (Address Latch Enable): This signal is used to demultiplex the multiplexed the address and data at the falling edge of the ALE. i. If ALE = 1 => AD0-AD15 will form A0-A15 ii. If ALE =0 => AD0-AD15 will form D0-D15. 3. **$\overline{\text{DEN}}$ (Data Enable)**: It provides an output enable for the 8286 in a minimum mode which uses a transceiver. It is active LOW during each memory and I/O access and for INTA cycle. 4. **DT/$\overline{\text{R}}$ (Data Transmit / Receive):** It is an output signal which controls the direction of data flow through the transceivers. If it is at logic 1 the buffers are enabled to transmit data from the 8086. If it is at logic 0 the buffers are enabled to receive data. 5. **M/$\overline{\text{IO}}$:** It is used to distinguish a memory transfer or I/O transfer. For memory operation M/IO=1 and for I/O operation M/IO=0.	*Function of any 4 control signals 2M*

		6. **$\overline{WR}$:** It is used by the 8086 for outputting a low to indicate that the processor is performing a write memory or write I/O operation depending on the $M/\overline{IO}$ signal. 7. **HOLD:**This is s request signal which is given by peripheral device to the microprocessor to have control over address and data lines. 8. **HLDA:** If the microprocessor is ready to give the control of address and data lines to external device then it provides Hold Acknowledge.	
	c) **Ans.**	**List the addressing modes of 8086 and describe them with an example.** **Addressing Modes:** 1. Immediate Addressing Mode 2. Register Addressing Mode 3. Direct Addressing Mode 4. Indirect Addressing mode 5. Register Indirect Addressing Mode 6. Based Addressing with displacement 7. Indexed Addressing Mode 8. Based Indexed Addressing Mode 9. Based Indexed Addressing with Displacement Mode 10. Fixed or Direct Port Addressing 11. Variable or Indirect Port Addressing 12. Implied (Implicit) Addressing Modes 1. Immediate Addressing Mode: In immediate addressing 8/16 bit data is specified as a part of instruction or specified in the instruction itself. The immediate operand can be only source operand. Ex: MOV CL, 03H ADD AX, 1234H. 2. Register Addressing Mode: In this addressing mode the source and destination operand are specified in a register. The operand can be 8/16 bit wide. The 8 bit operand can be any one of the register: AL, AH, BH, BL, CH, CL, DH, DL and the 16-bit operand can be AX, BX, CX, DX, SI, DI, SP. The 16-bit operand can be also be either of the segment registers. Ex: MOV AL, BL ADD CL, DL	**6M** ***List (any 4) -2M*** ***Any 4 description – 1M each***

MOV DS, AX

3. Memory Addressing Mode: The memory addressing mode is classified under two categories:

- Direct Addressing Mode: In this 16-bit offset address is provided in the instruction itself. Here [ ] refers the contents of the offset address.
  Ex: MOV AL, [2000H]; MOV [1020], 5050H
- Indirect Addressing mode: In this mode the Effective address is calculated from the contents of one or two registers along with the displacement value. The indirect addressing mode is classified in five categories:

i. Register Indirect Addressing Mode: In this mode EA is provided in an index register or base register. The index register can be SI or DI and the base register can be BX.
EA= [BX, SI, DI]
Ex: MOV [DI], 1234H; MOV AX, [BX]

ii. Based Addressing with displacement: In this mode EA is sum of an 8/16 bit displacement and the contents of base register (BX or BP).
Ex: MOV AX, [BX+300H]; MOV AX, [BX-2H]

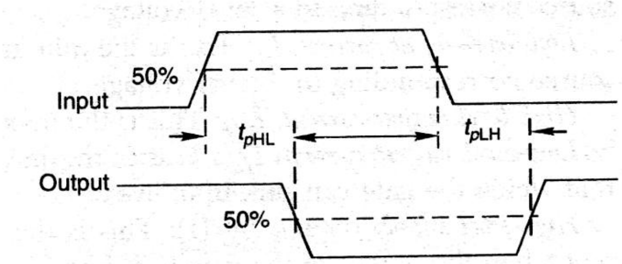
iii. Indexed Addressing Mode: In this EA is the sum of the 8/16 bit displacement plus the contents of the index registers SI or DI.
Ex: MOV [DI + 2345H], 1234H; MOV AX, [SI + 45H]

iv. Based Indexed Addressing Mode: In this EA is the sum of base registers (BX or BP) and the indexed register (SI or DI) both which are specified in the instruction.
Ex: MOV [BX + DI], 1234H; MOV AX, [SI + BX]

v. Based Indexed Addressing with Displacement Mode: In this EA is the sum of base registers (BX or BP) and the indexed register (SI or DI) along with the 8/16 bit displacement.
Ex: MOV [DI + BX + 37H], AX; MOV AL, [BX + SI + 278H]

4. I/O Port addressing: There are two types of I/O port addressing:

i. Fixed or Direct Port Addressing: In this case a one byte port

		address will be provided in the instruction. This allows fixed access to ports numbered 0 to 255 (00-FFH). Ex: OUT 06H, AL; IN AX, 85H ii. Variable or Indirect Port Addressing: In this case port address will not be explicitly in the instruction. The address of port number is taken from DX allowing 64K 8 bit ports or 32K 16 bit ports. This mode is known as variable or indirect port address. The 8 and 16 bit I/O data transfers should take place only through AL or AX. Ex: IN AL, DX; OUT DX, AX. 5. Implied (Implicit) Addressing Modes: In this the instructions does not have any operand. Ex: CLC, DAA	
6.	 **a)** **Ans.**	**Attempt any <u>TWO</u> of the following:** **Define the following term with respect the digital IC's:** **(i) Propagation delay** **(ii) Fan in** **(iii) Fan out** **(iv) Power Dissipation** **(v) Noise Margin** **(vi) Threshold Voltage.** **(i) Propagation delay:** Propagation delay is defined as the time taken to obtain the O/P when the I/P is applied. It is given in nano seconds. (1 ns=10^{-9} sec). The I/P and O/P wave forms of a logic gate are as follows:  The delay times are measured between 50% voltage levels of I/P & O/P wave forms. There are 2 delay times t_{PHL} when O/P goes from high to low & t_{PLH} when it goes from low to high. The propagation delay time of the logic gate is taken as the average of these 2 delay	**12** **6M** ***Each definition 1M***

times.

(ii) Fan in: Fan-In is defined as the number of inputs the gate has. For e.g. a two input gate will have fan-in equal to 2.

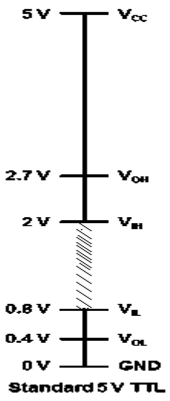
(iii) Fan out: Fan-out is the no. of similar gates which can be driven by the gate. High fan out is better as it reduces need for additional drivers to drive more gates

(iv) Power dissipation: Power dissipation is the power required in mW in an IC. Low power requirement indicates low speed of operation & vice versa. Hence, to select an IC, figure of merit is considered. It is the product of propagation delay & power, i.e. ns x mw = pJ. The gate of the lowest fig. of merit is selected.

(v) Noise margin: Some electric & magnetic fields can induce unwanted voltages on the wires between logic circuits. They are called ‘Noise Signals’. They may cause a change in VIH or VIL & may produce undesired operation. The ability of circuit to tolerate these noise signals is called as Noise immunity. These are indicated by noise margins. If they are defined above, they are called DC noise margins. If the noise pulse width is less & is approaching the propagation delay of circuit, it is called AC noise margin.

(vi) Threshold voltage: For any logic family, there are a number of threshold voltage levels to know:

1. V_{OH} -- Minimum OUTPUT Voltage level a TTL device will provide for a HIGH signal.
2. V_{IH} -- Minimum INPUT Voltage level to be considered a HIGH.
3. V_{OL} -- Maximum OUTPUT Voltage level a device will provide for a LOW signal.
4. V_{IL} -- Maximum INPUT Voltage level to still be considered a LOW.



	b)	**Write an assembly language program to arrange any array of 10 bytes in ascending order. Draw flowchart for the same.** ***(Note: Any other logic shall also be considered).***	**6M**

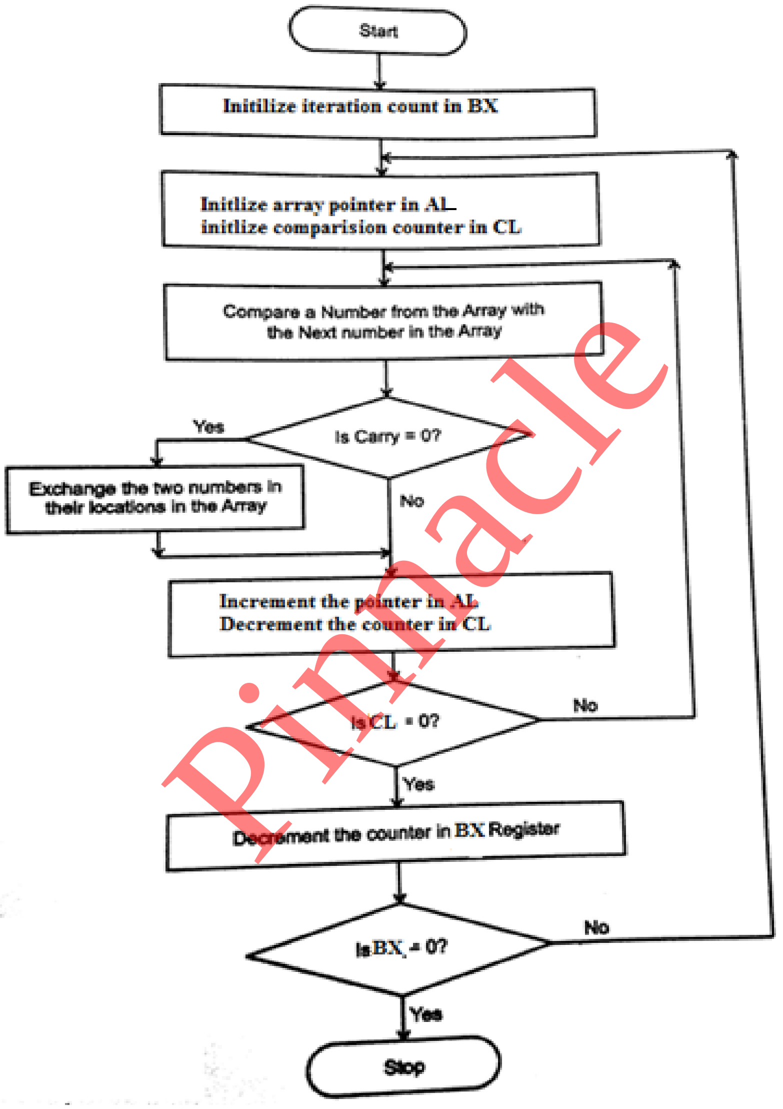
Ans.

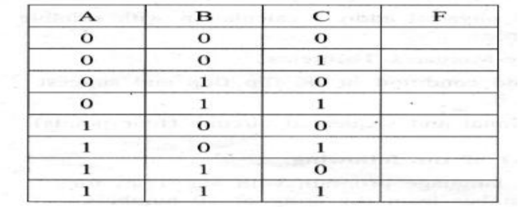
Program:

```
DATA SEGMENT
ARRAY DB 15h,05h,08h,78h,56h, 60h, 54h, 35h, 24h, 67h
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS:DATA
START:MOV DX, DATA
        MOV DS, DX
        MOV BL,0AH
step1: MOV SI,OFFSET ARRAY
MOV CL,09H
step:   MOV AL,[SI]
CMP AL,[SI+1]
JC Down
XCHG AL,[SI+1]
XCHG AL,[SI]
Down : ADD SI,1
       LOOP step
       DEC BL
       JNZ step1
       MOV AH,4CH
       INT 21H
       CODE ENDS
       END START
```

Flowchart:

Correct Program 4M

		 Start Initilize iteration count in BX Initlize array pointer in AL initlize comparision counter in CL Compare a Number from the Array with the Next number in the Array Is Carry = 0? Yes Exchange the two numbers in their locations in the Array No Increment the pointer in AL Decrement the counter in CL Is CL = 0? No Yes Decrement the counter in BX Register Is BX = 0? No Yes Stop	*Flowchart 2M*
	c)	**Refer given Fig. No.1 and write the outputs for each of the following input:**	6M



A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

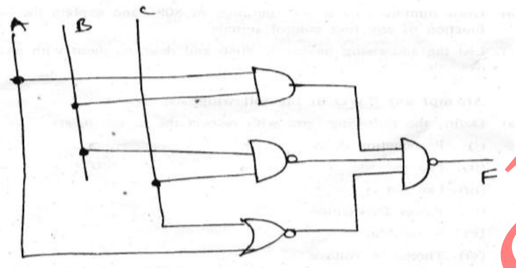


Fig. No. 1

Ans.

(Note: Writing Boolean expression shall be considered as option. Any four correct output shall be given 3M).

$$F=\overline{(AB).(\overline{BC}).(\overline{A+C})}$$

$$F=\overline{AB}+\overline{\overline{BC}}+(A+C)$$

$$F=\overline{A}+\overline{B}+BC+A+C$$

$$F=A+\overline{A}+\overline{B}+BC+C$$

$$F=1+\overline{B}+C$$

$$F=1+C$$

$$F=1$$

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Correct outputs 6M