V2V EDTECH LLP

Online Coaching at an Affordable Price.

OUR SERVICES:

- Diploma in All Branches, All Subjects
- Degree in All Branches, All Subjects
- BSCIT / CS
- Professional Courses

+91 93260 50669

v2vedtech.com

V2V EdTech LLP

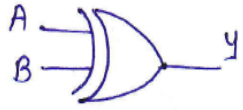
v2vedtech

WINTER – 2018 EXAMINATION
MODEL ANSWER

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

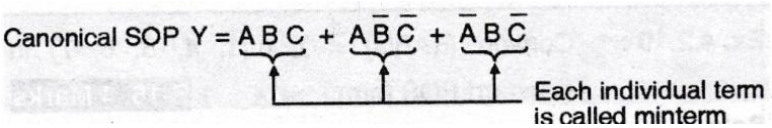
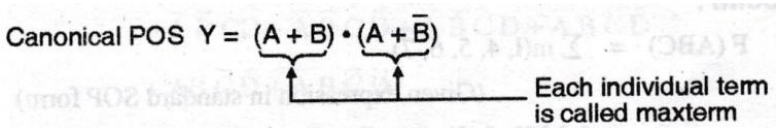
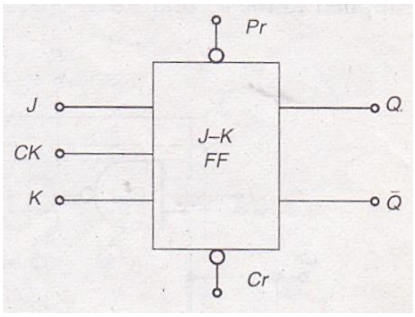
Important Instructions to examiners:

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.

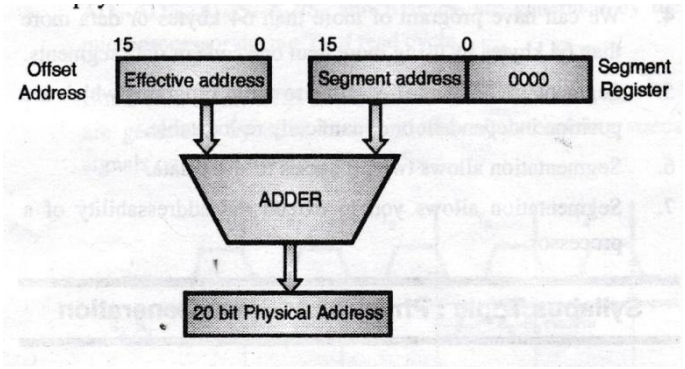
Q. No.	Sub Q.N.	Answer	Marking Scheme
1.	**a) Ans.**	**Attempt any FIVE of the following:** **Draw symbol and write truth table of EX-OR gate.** **Symbol**  **Truth Table** Truth Table for two input EX-OR gate. A logical gate whose output is one when odd number of inputs are one, for any other condition output is low. Inputs: A, B — Output: Y 0 0 0 1 0 1 0 1 1 1 1 0	**10** **2M** ***Symbol 1M*** ***Truth Table 1M***

WINTER – 2018 EXAMINATION
MODEL ANSWER

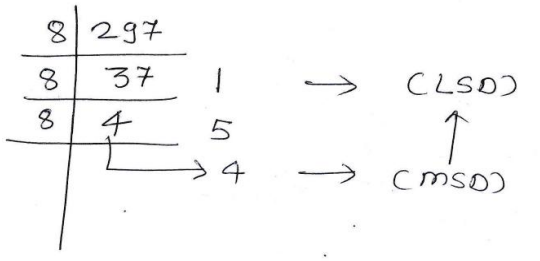
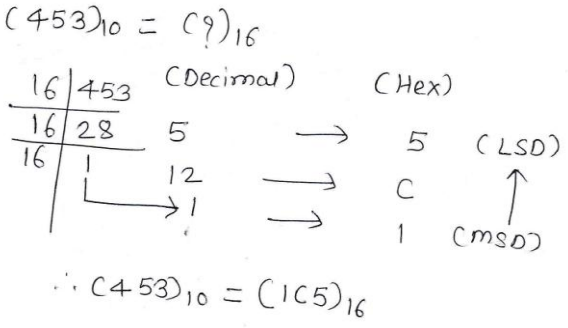
Subject: Digital Techniques & Microprocessor **Subject Code:** 22323

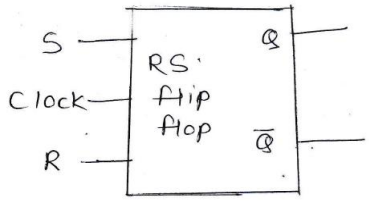
	b) **Ans.**	**Define terms "Minterm" and "Maxterm" with proper example of each.** **Minterm:** Each individual term in the canonical SOP form is called as Minterm. Example:  **Maxterm:** Each individual term in the canonical POS form is called as Maxterm. Example:  	***2M*** ***Each Definition with example 1M*** ***Each Definition with example 1M***
	c) **Ans.**	**Draw symbol of JK flip-flop and write its truth table.** **Symbol**  **Truth Table:** Inputs: J_n, K_n; Output: Q_{n+1} 0 0 Q_n 0 1 0 1 0 1 1 1 $\overline{Q_n}$	**2M** ***Symbol 1M*** ***Truth Table 1M***

	d) Ans.	**State importance of pipelining in 8086 microprocessor** • In pipelining, while the current instruction is executing, next instruction is fetched using a queue. • Pipelining enables many instructions to be executed at the same time. • It allows execution to be done in fewer cycles. • Speed up the execution speed of the processor. • More efficient use of processor.	***2M*** ***Any two points 2M***
	e) Ans.	**Give any four applications of digital circuits.** Applications of digital circuits i) Object Counter ii) Dancing Lights iii) Scrolling Notice board iv) Multiplexer v) Digital Computers vi) Washing machines, Television vii) Digital Calculators viii) Military Systems ix) Medical Equipments x) Mobile Phones xi) Radar navigation and guiding systems xii) Microprocessors	***2M*** ***Any relevant four applications 2M***
	f) **Ans.**	**Define the following terms –** **(i) Physical Address** **(ii) Effective Address** **(i) Physical Address** ***(Note: Diagram is Optional)*** Physical: The address given by BIU is 20 bit called as physical address. It is the actual address of the memory location accessed by the microprocessor. It is calculated as	***2M*** ***Each definition 1M***

		 **(ii) Effective Address** Effective Address: Effective address or the offset address is the offset for a memory operand. It is an unassigned 16 bit number that gives the operand's distance in bytes from the beginning of the segment.	
	g) **Ans**	**Choose instruction for following situations:** **(i) Addition of 16 bit Hex. No with carry** **(ii) Division of 8 bit No.** **(iii) Rotate content of BL by 4 bit.** **(iv) Perform logical AND operation of AX and BX** **(i) Addition of 16 bit Hex. No with carry** ***(Note any other relevant registers shall also be considered)*** ADC Destination 16, Source 16 **OR** ADC AX, BX **OR** ADC AX, 4500H **(ii) Division of 8 bit No.** ***(Note any other relevant registers shall also be considered)*** DIV SOURCE **OR** DIV BL	***2M*** ***Each instruction ½ M***

		(iii)Rotate content of BL by 4 bit.
		MOV CL,04H
		ROR BL, CL
		OR
		MOV CL, 04H
		ROL BL, CL
		
		(iv) Perform logical AND operation of AX and BX
		AND AX,BX	

2.	**a)** **Ans.**	**Attempt any THREE of the following:** **Convert following decimal to octal and Hexadecimal** **i) $(297)_{10}$ = ( )$_8$** **ii) $(453)_{10}$ = ( )$_{16}$** **(i) $(297)_{10}$ = ( )$_8$**  $\therefore (297)_{10} = (451)_8$ **(ii) $(453)_{10}$ = ( )$_{16}$**  	**12** **4M** ***Each conversion 2M***

	b) **Ans.**	**Convert the given minterm into standard POS form.** $Y(A,B,CD) = (\bar{A}.BC) + (B.\bar{C}\,\bar{D}) + (\bar{A}\,\bar{B})$ ***Note: Solution is given by considering*** $Y(A,B,CD)$ ***as*** $Y(A,B,C,D)$ $Y(A,B,C,D) = \bar{A}BC + B\bar{C}\bar{D} + \bar{A}\bar{B}$ Converting into standard SOP form, $Y(A,B,C,D) = \bar{A}BC(D+\bar{D}) + (A+\bar{A})B\bar{C}\bar{D} + \bar{A}\bar{B}(C+\bar{C})(D+\bar{D})$ $= \bar{A}BCD + \bar{A}BC\bar{D} + AB\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}\bar{B}CD + \bar{A}BC\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}\bar{C}\bar{D}$ $= \sum m(7,6,12,4,3,2,1,0)$ $= \sum m(0,1,2,3,4,6,7,12)$ $= \prod M(5,8,9,10,11,13,14,15)$ ... Standard Pos form $= (A+\bar{B}+C+\bar{D})(\bar{A}+B+C+D)(\bar{A}+B+C+\bar{D})$ $(\bar{A}+B+\bar{C}+D)(\bar{A}+B+\bar{C}+\bar{D})(\bar{A}+\bar{B}+C+\bar{D})$ $(\bar{A}+\bar{B}+\bar{C}+D)(\bar{A}+\bar{B}+\bar{C}+\bar{D})$.	***4M*** ***Standard SOP form 2M*** ***Conversion to Standard POS 2M***
	c) **Ans.**	**Draw symbol and write truth table for the following flip flop and give one application of each.** **i) Clocked R-S flip flop** **ii) T- flip flop** **(i) Clocked R-S flip flop** **Symbol**  	***4M*** ***Symbol ½ M***

Truth Table

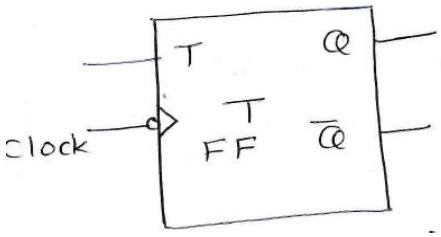
Clock	S	R	Q_{n+1}	$\overline{Q}_{n+1}$	Remark
0	X	X	Q_n	$\overline{Q}_n$	No change
1	0	0	Q_n	$\overline{Q}_n$	No Change
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	Race	Race	Avoid

Truth table 1M

Application:
i) Clocked RS flip-flop can be used in sequential circuits.
ii) It can be used to design counters.
iii) It can be used as a latch in digital circuits.

Any one Application ½ M

(ii) T- flip flop
Symbol

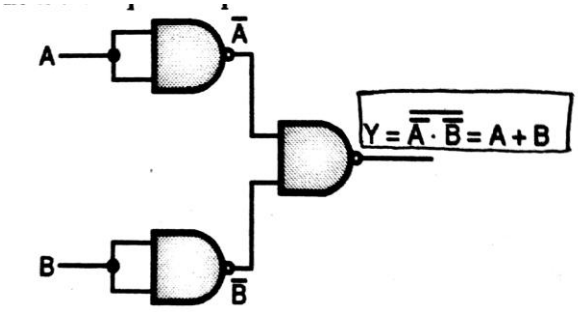


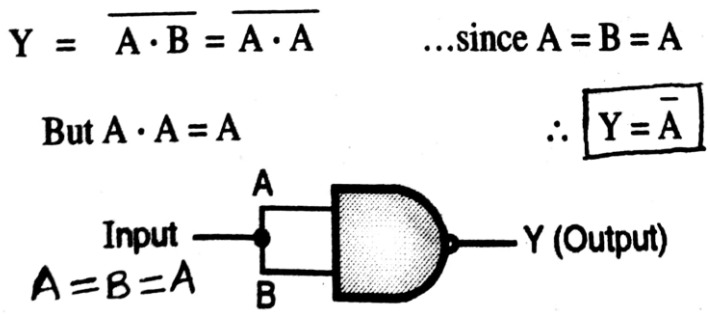
Symbol ½ M

Truth Table

T	Q_{n+1}
0	Q_n
1	$\overline{Q}_n$

Truth table 1M

		Application: i) Used to design counters in digital circuits. ii) Can be used in frequency divider circuits.	***Any one Application ½ M***
	d) Ans.	**Prove $A(\bar{A}+C)(\bar{A}B+C)(\bar{A}BC+\bar{C})=0$** ***Note: Any other relevant laws applied shall be considered while obtaining the correct answer.*** L·H·S. $= A(\bar{A}+C)(\bar{A}B+C)(\bar{A}BC+\bar{C})$ $=(A\bar{A}+AC)(\bar{A}B+C)(\bar{A}BC+\bar{C})$ $=(0+AC)(\bar{A}B+C)(\bar{A}BC+\bar{C})$ $(\because A\bar{A}=0)$ $=(A\bar{A}BC+AC)(\bar{A}BC+\bar{C})$ $(\because CC=C)$ $=(0+AC)(\bar{A}BC+\bar{C})$ $(\because A\bar{A}=0)$ $=A\bar{A}BC+AC\bar{C}$ $(\because CC=C)$ $=0+0$ $(\because A\bar{A}=0 \text{ and } C\bar{C}=0)$ $=0$ $=$ R·H·S· Hence Proved	***4M*** ***Correct solution 4M***
3	**a) Ans.**	**Attempt any <u>THREE</u> of the following:** **Implement OR gate and NOT gate using "Universal NAND gate". Write expressions for both.** 1. "OR" gate using "Universal NAND" gate:  	**12** **4M** ***Output Expression 1M*** ***Circuit Diagram 1M***

		2. "NOT" gate using "Universal NAND" gate: 	***Output Expression 1M*** ***Circuit Diagram 1M***
	b) **Ans**	**Explain following instructions for 8 bit and 16 bit data.** **(i) PUSH** **(ii) DAA** **(iii) IDJV** **(iv) XOR** ***Note: Any other relevant registers shall also be considered in the example/explanation.*** **(i) PUSH** Format: **PUSH source** This instruction decrements the SP (Stack Pointer) register (by 2) and copies the word specified by source to the location at the top of the stack. Here, Source can be a 16-bit general purpose register, segment register or memory location. Example- PUSH AX **OR** **PUSH AX** This instruction decrements the stack pointer by 2 and copies the 16 bit data from AX register to the stack segment where the stack pointer then points.	**4M** ***Explaination of each ½ M*** ***Example for each case ½ M***

(ii) DAA

DAA stands for Decimal Adjust Accumulator AL after BCD Addition

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 AH =1
+  0110 0110
-------------------------
   1001 1000 =AL =98 in BCD
```

same type example for 16 bit can be considered.

OR

DAA instruction is used to convert the sum of two packed BCD numbers in the register AL into a correct BCD number.

Example :

```
MOV AL, 23H
MOV BL, 47H
ADD AL, BL
DAA
```

After the execution of the above instructions, the result in AL = 70H.

(iii) IDJV
(NOTE: CONSIDER THE GIVEN INSTRUCTION AS IDIV):
Syntax : IDIV source
It divides a signed word in AX by an signed byte in source during 16/8 division. Also it is used to divide a signed double word in DX,AX by an signed word in source during 16/8 division.
operation:

a. if the source is byte then

AL ⟵ AL/signed 8 bit source
AH ⟵ AL MOD signed 8 bit source

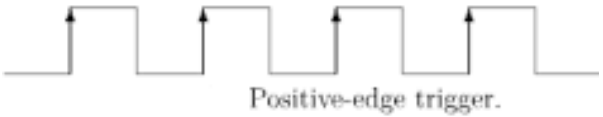
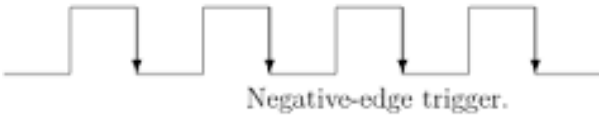
b. if the source is word then

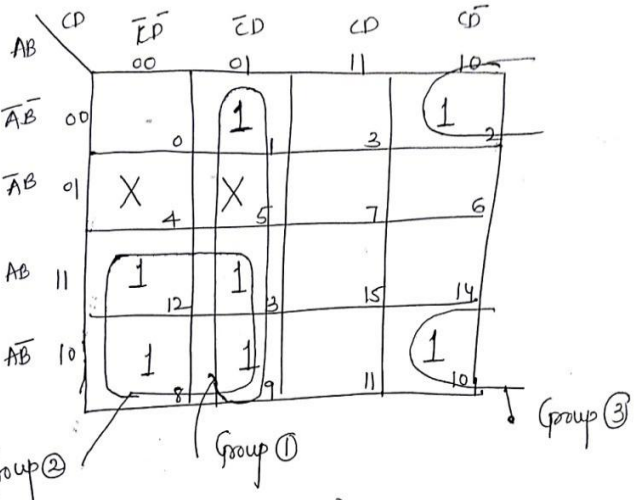
AX ⟵ DX,AX/signed 16 bit source
DX ⟵ DX,AX MOD signed 16 bit source

OR

IDIV BL
This instruction is used to divide signed word in AX register by signed byte in BL register. The quotient after division will be stored in AL register, whereas the remainder is stored in AH register.
IDIV BX
This instruction is used to divide signed double word in DX,AX register by signed word in BX register. The signed 16 bit quotient will be stored in AX register, whereas the signed 16 bit remainder is stored in AH register.

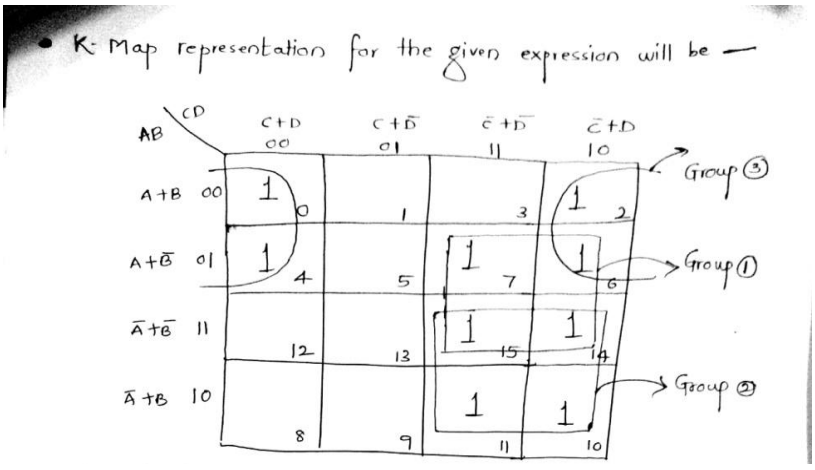
(iv) **XOR** − Used to perform Exclusive-OR operation over each bit in a byte/word with the corresponding bit in another byte/word.
Syntax: XOR Destination, Source
Example:
For 8bit data:
XOR AL, BL
This instruction performs Exclusive-OR bit by bit at AL with BL and the result is stored in AL..
For 16bit data:
XOR AX, BX
This instruction performs Exclusive-OR bit by bit word at AX with word in BX and the result is stored in AX.

	c) **Ans.**	**Draw waves for positive and negative triggering with proper labeling. Identify two situations where these triggering can be used?** ***Note: Any additional relevant point related to triggering situation shall be considered***  Positive-edge trigger.  Negative-edge trigger. 1. Edge triggering can be used in flipflops as clock input. 2. It is used in counters circuits. 3. They can be used in shift registers 4. They can be used to synchronous data.	**4M** ***Diagram 2M*** ***Any relevant situation where triggering is used 2M***
	d) **Ans**	**Simplify Y=F(A, B,CD)** **= ∑m (1, 2, 8, 9, 10, 12, 13) + d(4,5)** **Using K-map and write expression** ***Note: Solution is given considering Y=F(A, B,CD) as Y= F(A, B,C,D)***	**4M**

		• K-map representation for the given expression will be –  • To find equation (expression) – Group ① (QUAD) ⇒ $\bar{C}D$ Group ② (QUAD) ⇒ $A\bar{C}$ Group ③ (PAIR) ⇒ $\bar{B}C\bar{D}$ Therefore, The Required expression is, $f(A,B,C,D) = \bar{C}D + A\bar{C} + \bar{B}C\bar{D}$	***Correct K-map 2M*** ***Correct equation 2M***
4	**a)**	**Attempt any <u>THREE</u> of the following** **Suggest "Two instruction" for each of the following addressing modes.** **(i) Register Addressing Mode.** **(ii) Direct Addressing Mode** **(iii) Based Indexed Addressing Mode** **(iv) Immediate Addressing Mode.**	**12** **4M**

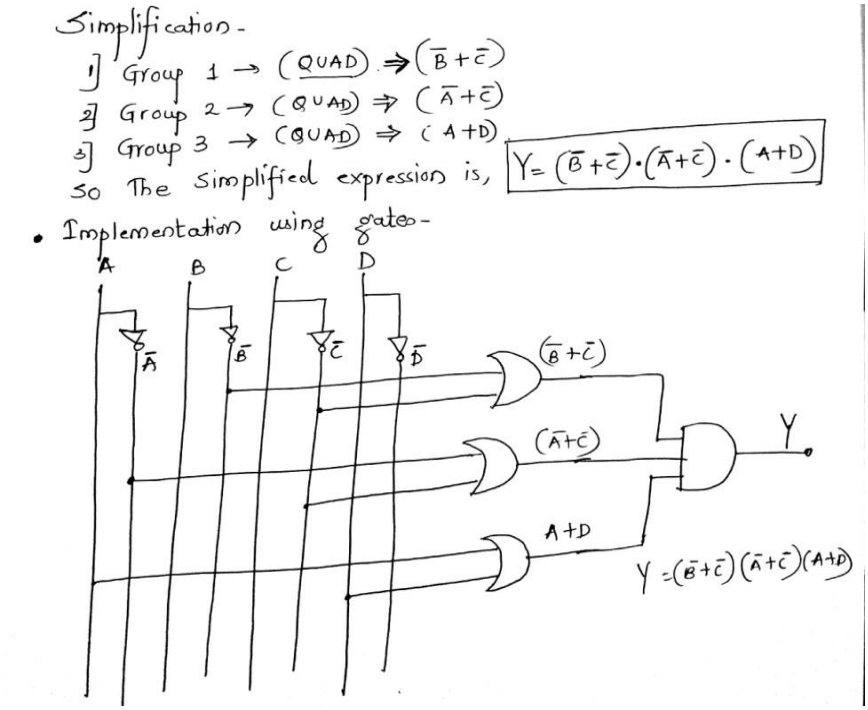
WINTER – 2018 EXAMINATION
MODEL ANSWER

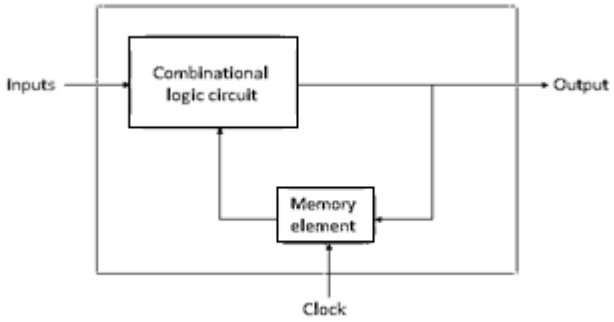
Subject: Digital Techniques & Microprocessor **Subject Code:** 22323

	Ans	i) **Register Addressing Mode:** a. MOV AX, CX b. AND AL, BL c. ROR AL, CL ii) **Direct addressing mode:** a.MOV AL, [3000H] b. AND AX,[8000H] c.INC [4712H] iii) **Based indexed Addressing mode:** 1.MOV AX, [BX][SI] 2.ADD AL, [BX][DI] 3.MOV AX, [BX+SI] iv) **Immediate addressing mode:** 1.MOV AL, 46H 2. MOV BX, 1234H 3. MOV DX, 0040H	***Consider any two instruction, each instruction ½ M***
	b) **Ans.**	**Minimize the expression and draw logic circuit using basic gates.** **F (A,B,CD) = πm {0, 2, 4, 6, 7, 10, 11, 14, 15}** ***Note: Solution is given considering Y=F(A, B,CD) as Y= F(A, B,C,D)*** 	***4M*** ***Correct K-Map 2M***

WINTER – 2018 EXAMINATION
MODEL ANSWER

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

			***Simplification 1M*** ***Logic diagram 1M***
	c) **Ans.**	**Compare combinational and sequential circuits. Draw block diagram of sequential circuit and describe the function of each block** **PARAMETERS** | **COMBINATIONAL CIRCUIT** | **SEQUENTIAL CIRCUIT** **Definition** | The output at any instant of time depends upon the input present at that instant of time. | The output at any instance of time depends upon the present input as well as past input and output. **Need of Memory** | No memory element required in the ckt | Memory element required to stored bit **Need of clock** | Clock input not necessary | Clock input necessary **Examples** | E.g. Adders, Subtractors ,Code converters, comparators etc. | E.g. Flip flop, Shift registers, counters etc, **Applications** | Used to simplify Boolean expressions, k-map , Truth table | Used in counters & registers	**4M** ***Any 2 differences 2M***



1. Sequential logic circuits are those, whose output depends not only on the present value of the input but also on previous values of the input signal.
2. Sequential circuit can be considered as combinational circuit with feedback circuit.
3. Sequential circuit uses a memory element like flip – flops as feedback circuit in order to store past values.

Block diagram 1M

Explanation 1M

d) **i) Differentiate between RISC and CISC processor (Three point)**
ii) Compare 8086 and 80586 (Pentium)(3 points)

4M

Ans **i) Differentiate between RISC and CISC processor (Three point)**

Sr. No	PARAMETER	RISC PROCESSOPR	CISC PROCESSOR
1.	Instruction set	Few instructions	More instructions
2.	Data types	Few data types	More data types
3.	Addressing mode	Few Addressing modes	More Addressing modes
4.	Registers	Large number of general purpose registers	Small number of general purpose registers & special purpose registers.
5.	Architecture type	Load/store architecture	No load/store architecture
6.	Operation	Single- cycle	Multi-cycle
7.	Design	Hardwired control	Micro-coded
8.	Instruction Set format	Fixed length	Variable length

Any three points 2M

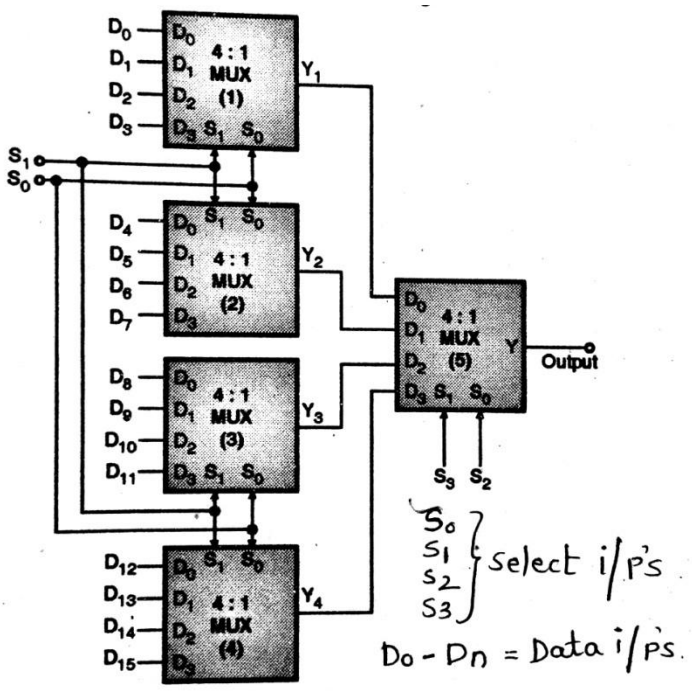
ii) Compare 8086 and 80586 (Pentium)(3 points)

SR. NO	PARAMETER	8086	80586 (Pentium)
1.	Data Bus	16 bit	64 bit
2	Address Bus	20 bit	32 bit
3	Physical memory	1 MB	4 GB
4	Register size	16 bit	32 bit
5	Voltage required	5 V	3.3 V
6	Clock type	1x	3x
7	Pipelining	Yes	Yes

Any three points 2M

e) **Ans.**

Draw 16:1 multiplexer using 4:1 multiplexers "ONLY" with proper labels.



4M

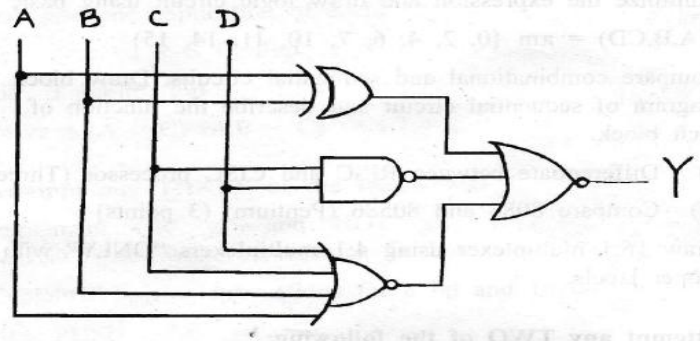
Correct Diagram 3M

Proper Labeling 1M

5		**Attempt any <u>TWO</u> of the following:**	**12**
	a)	**Write algorithm and 8086 assembly language program to find average salary of five employees of "SILICON Systems". Assume 4 digit salary of each employee. Also write output.**	**6M**
	Ans.	***Note: Any other correct logic shall be considered.***	
		ALGORITHM	***Algorithm 2M***
		1. START 2. DEFINE ARRAY SALARY OF 5 NUMBERS EACH 4 DIGIT IN DATA SEGMENT 3. DEFINE VARIABLE AVG TO STORE RESULT IN DATA SEGMENT 4. MOVE DATA IN AX 5. MOVE DATA FROM AX TO DS 6. MOVE NUM1 TO CX TO SET COUNTER 7. LOAD ADDRESS OF ARRAY SALARY TO BX 8. MOVE 0000H TO AX 9. ADD CONTENTS OF MEMORY POINTED BY BX TO AX 10. IF NO CARRY, GOTO STEP 12 11. INCREMENT DX REGISTER 12. INCREMENT BX TWICE TO POINT TO NEXT NUMBER 13. DECREMENT COUNTER CX; IF NOT ZERO GOTO STEP 9 14. DIVIDE THE SUM BY NUM1 15. STORE THE RESULT AX INTO AVG 16. END	
		PROGRAM	***Program 3M***

		DATA SEGMENT
		SALARY DW 4000H,5000H,6000H,7000H,8000H
		NUM1 DW 05H
		AVG DW ?
		DATA ENDS
		CODE SEGMENT
		ASSUME DS:DATA, CS:CODE
		START:
		MOV AX,DATA
		MOV DS,AX
		MOV CX,NUM1
		MOV BX, OFFSET SALARY
		MOV AX,0000H	

		L1: ADD AX, [BX]
		JNC NEXT
		INC DX
		NEXT: INC BX
		INC BX
		LOOP L1
		DIV NUM1
		MOV AVG,AX
		MOV AH,4CH
		INT 21H
		CODE ENDS
		END START
		
		Output
		AVG=6000H
			Output 1M

	b)	**Refer Fig No. 1 and write truth table and output "Y", write expression at output of gates. Redraw the Fig. No. 1."** Truth Table Inputs / Output A B C D Y 0 0 0 0 ⋮ ⋮ ⋮ ⋮ 1 1 1 1  **Fig No.1**	**6M**

WINTER – 2018 EXAMINATION
MODEL ANSWER

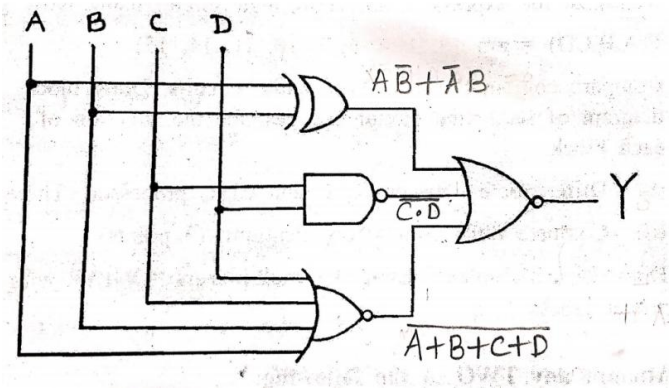
Subject: Digital Techniques & Microprocessor **Subject Code:** 22323

Ans

Truth Table

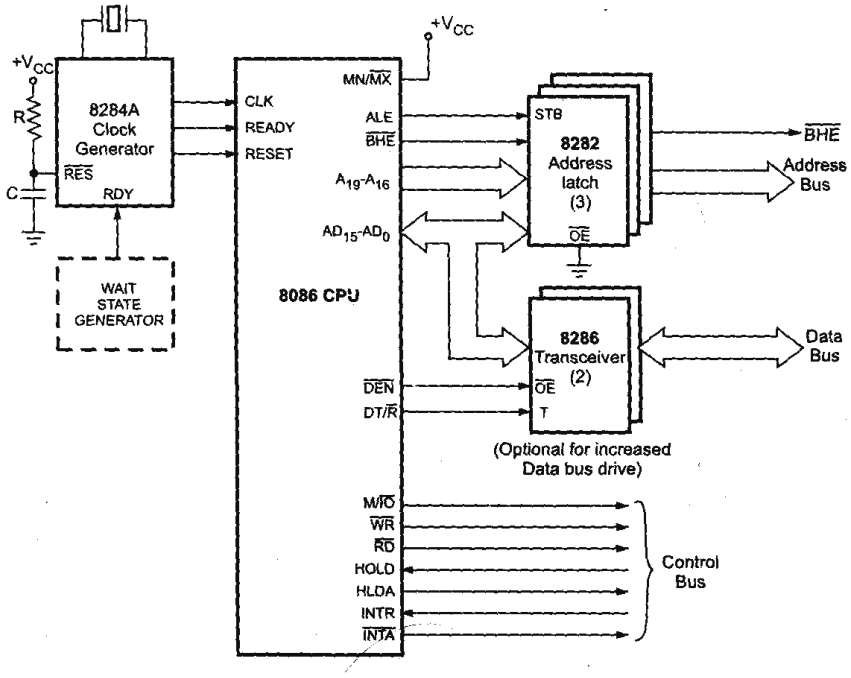
Truth Table				
Input				Output
A	B	C	D	Y
0	0	0	0	**0**
0	0	0	1	**0**
0	0	1	0	**0**
0	0	1	1	**1**
0	1	0	0	**0**
0	1	0	1	**0**
0	1	1	0	**0**
0	1	1	1	**0**
1	0	0	0	**0**
1	0	0	1	**0**
1	0	1	0	**0**
1	0	1	1	**0**
1	1	0	0	**0**
1	1	0	1	**0**
1	1	1	0	**0**
1	1	1	1	**1**

Truth Table 3M



$$Y = \overline{(A\bar{B}+\bar{A}B) + (\overline{C \cdot D}) + (\overline{A+B+C+D})}$$

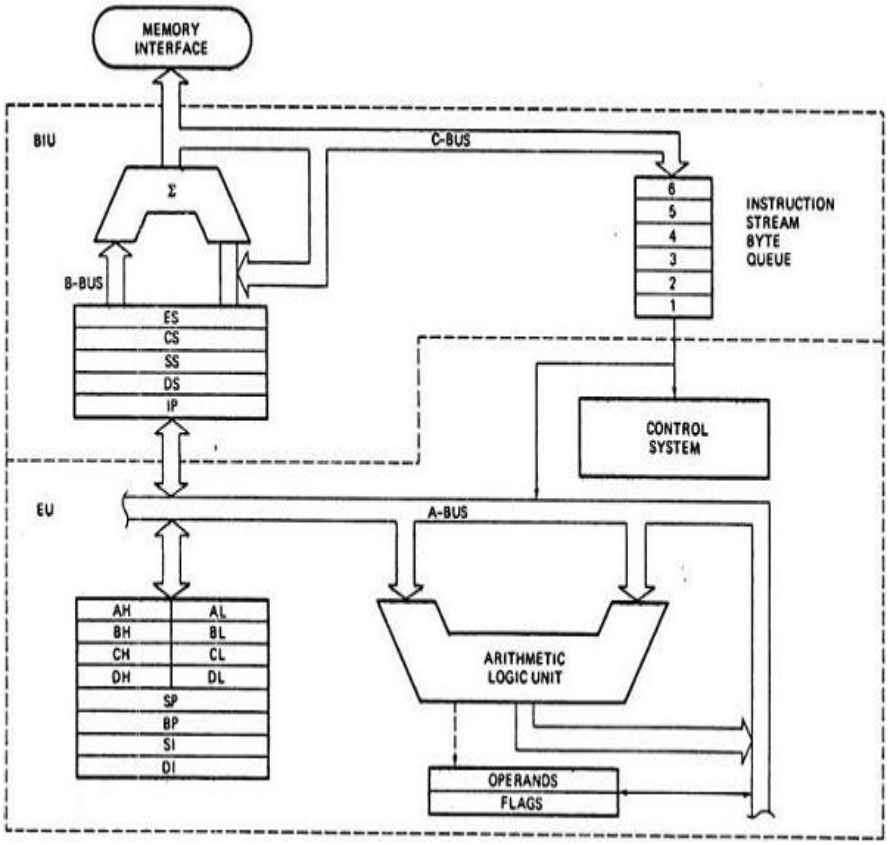
Expression at output of gates 3M

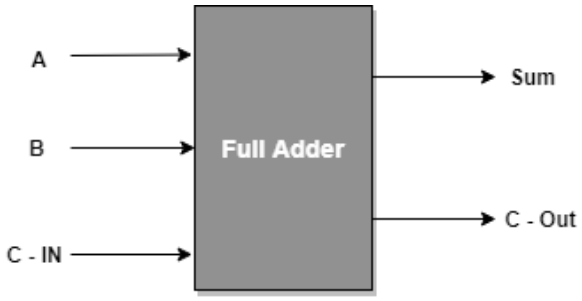
	c) **Ans.**	**Draw minimum mode configuration of 8086 and explain the function of each block.**  When MN/$\overline{MX}$ = 1 or connected to +V_{CC} as shown in the figure, the 8086 microprocessor operates in minimum mode system. • In this mode, the microprocessor chip itself gives out all the control signals. This is a single processor mode. • **The 8284 clock generator** in the system is used to generate the CLK and to synchronize some external signals with the system clock. It is also used to generate RESET and READY signal through wait state generator. • **Three 8282 address latches** are used for separating the valid address from the multiplexed address/data signals and the controlled by the ALE signal generated by 8086. • **Two 8286 Transceivers** are the bi-directional buffers. They are required to separate the valid data from the time multiplexed address/data signal. This is controlled by two signals, DEN & DT/$\overline{R}$.	**6M** ***Diagram 3M*** ***Explanation 3M***

WINTER – 2018 EXAMINATION
MODEL ANSWER

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

6		**Attempt any <u>TWO</u> of the following:**	**12**
	a)	**Draw architectural block diagram of 8086 microprocessor and describe the function of each block.**	**6M**
	Ans	***Note: Any other relevant diagram shall be considered.***	
		Internal architecture of Intel 8086:	***Explanation of blocks 3M***
		Intel 8086 is a 16 bit integer processor. It has 16-bit data bus and 20-bit address bus. The internal architecture of Intel 8086 is divided into two units, 1. Bus Interface Unit (BIU) 2. Execution Unit (EU). Bus Interface Unit (BIU) **Memory Interface:** The Bus Interface Unit (BIU) generates the 20-bit physical memory address and provides the interface with external memory (ROM/RAM). 8086 has a single memory interface. **Instruction Byte queue:** To speed up the execution, 6-bytes of instruction are fetched in advance and kept in a 6- byte Instruction Queue while other instructions are being executed in the Execution Unit (EU). **Segment registers:** There are four 16-bit segment registers, viz., the code segment (CS), the stack segment (SS), the extra segment (ES), and the data segment (DS). The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register. **Adder**: 8086's BIU produces the 20-bit physical memory address by combining a 16-bit segment address with a 16-bit offset address using the adder circuit. **2. Execution Unit:** **Control unit:** The instructions fetched by BIU in the instruction byte queue are decoded under the control of timing and control signals. **Arithmetic and Logic Unit (ALU) :** Execution unit has a 16 bit ALU, which performs arithmetic & logic operations. **General purpose register unit:** All general registers of the 8086 microprocessor can be used for arithmetic and logic operations. The general registers are: Accumulator register AL (8 bit), AX (AL & AH for 16 bit), Base register, Count register, Data register , Stack Pointer (SP), Base Pointer (BP), Source Index (SI), Destination Index (DI). **Flags**: is a 16-bit register containing 9 1-bit flags: Overflow Flag	

		(OF), Direction Flag (DF), Interrupt-enable Flag (IF), Single-step Flag (TF), Sign Flag (SF), Zero Flag (ZF), Auxiliary carry Flag (AF), Parity Flag (PF), Carry Flag (CF).  	***Block Diagram 3M***
	b) **Ans.**	**Design full adder using K-MAP and draw logic circuit using basic gates and write truth table.** ***Note : In logic diagram, instead of basic gates, Exclusive –OR (EXOR) gates shall be considered.*** Full Adder is the adder which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C_{in}. The output carry is designated as C_{out} and the normal output is designated as S which is SUM. A full adder logic is designed in such a manner that can take eight inputs together to create a byte-wide adder and cascade the carry bit from one adder to the another.	**6M**



Truth Table

Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Truth Table 2M

Based on the truth table, the Boolean functions for Sum (S) and Carry – out (C_{out}) can be derived using K – Map.

For Sum S :

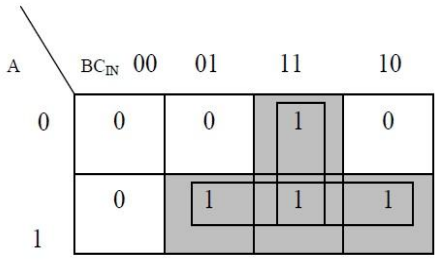
A \ BC_{IN}	00	01	11	10
0	0	1	0	1
1	1	0	1	0

The simplified equation for sum is

S = $\bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C} + ABC_{in}$

= $A \oplus B \oplus Cin$

For Carry – out (C_{out}) :

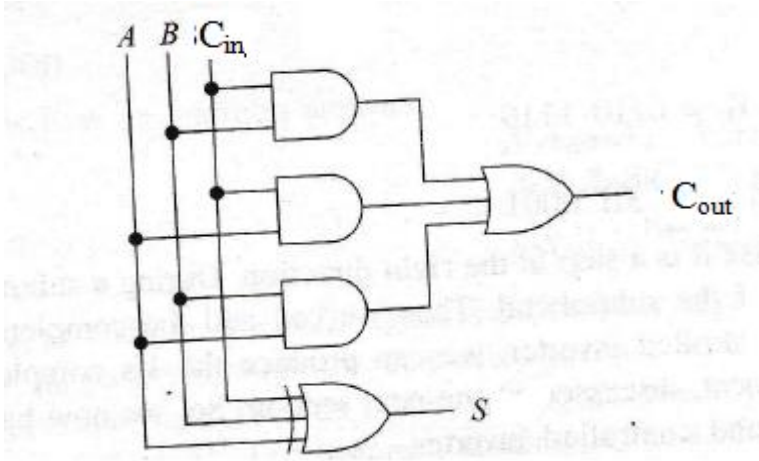


K map design 2M

The simplified equation for $C_{out}$is

$C_{out} = AB + AC_{in} + BC_{in}$

Logic Circuit Diagram



Logic diagram 2M

	c)
	
	Ans
		Write an assembly language program to find the largest number from an array of a 10 numbers. Assume suitable data.
		Note: Either 8bit or 16bit data shall be considered.
		DATA SEGMENT
		ARR DB 1,4,2,3,9,8,6,7,5,10
		LN DW 10
		L DB ?
		DATA ENDS
		CODE SEGMENT
		ASSUME DS:DATA, CS:CODE
		START:
		MOV AX,DATA
		MOV DS,AX
		LEA SI,ARR
		MOV AL,ARR[SI]
		MOV L,AL
		MOV CX,LN
		REPEAT: MOV AL,ARR[SI]
		CMP L,AL
		JG NOCHANGE (or JNC NOCHANGE)
		MOV L,AL
		NOCHANGE: INC SI
		LOOP REPEAT
		MOV AH,4CH
		INT 21H
		CODE ENDS
		END START
			6M
			
			Correct logic 3M
			
			Correct Instructions 3M

V2V EDTECH LLP

Online Coaching at an Affordable Price.

OUR SERVICES:

- Diploma in All Branches, All Subjects
- Degree in All Branches, All Subjects
- BSCIT / CS
- Professional Courses

+91 93260 50669

v2vedtech.com

V2V EdTech LLP

v2vedtech

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.		**Attempt any FIVE of the following:**	**10**
	a) Ans.	**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.	**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**

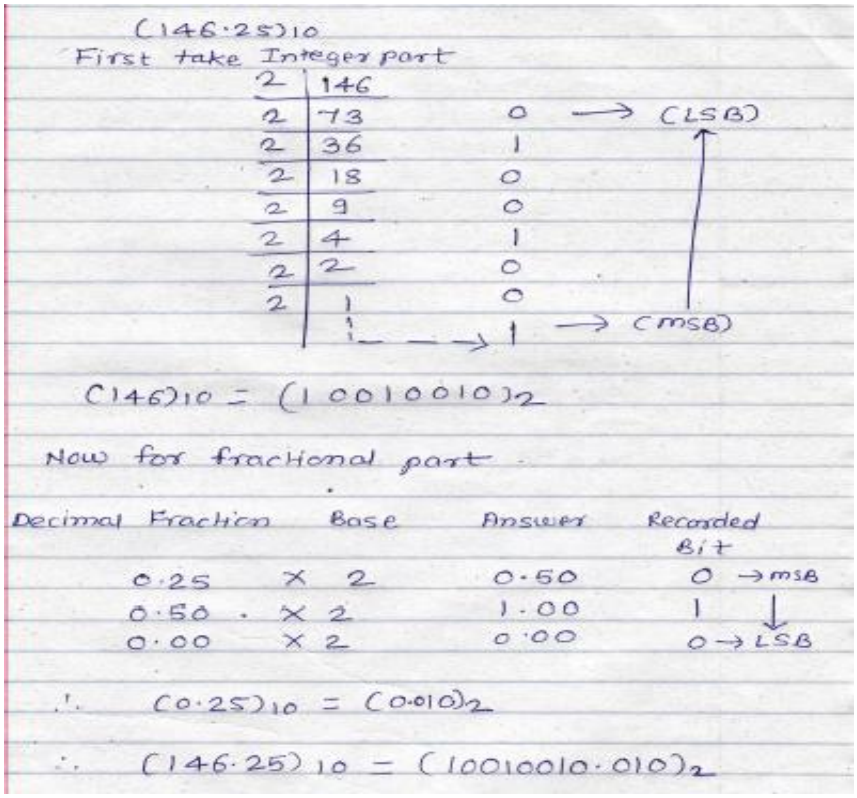
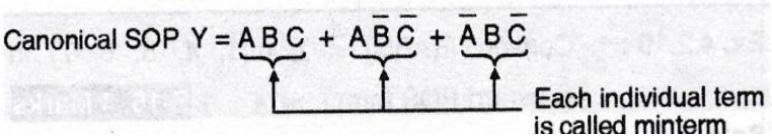
		2. Register Addressing Mode:
		Example: MOV AL, BL
		ADD CL, DL
		MOV DS, AX
		3. Direct Addressing Mode:
		Example: MOV AL, [2000H]
		MOV [1020], 5050H
		4. Register Indirect Addressing Mode
		Example: MOV [DI], 1234H
		MOV AX, [BX]
		5. Based Addressing with displacement
		Example: MOV AX, [BX+300H]
		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 1/2M each

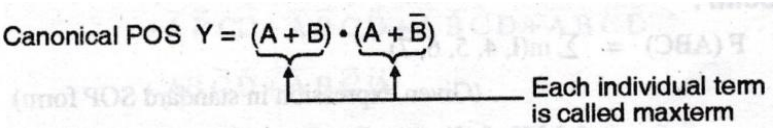
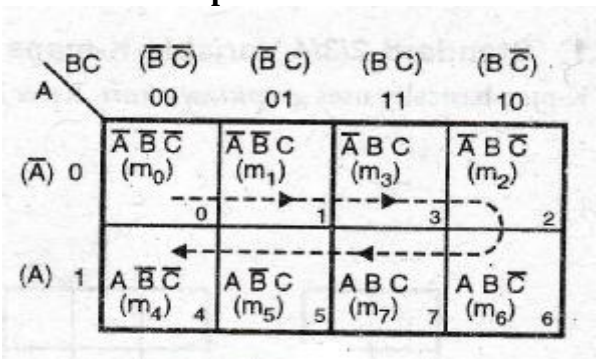
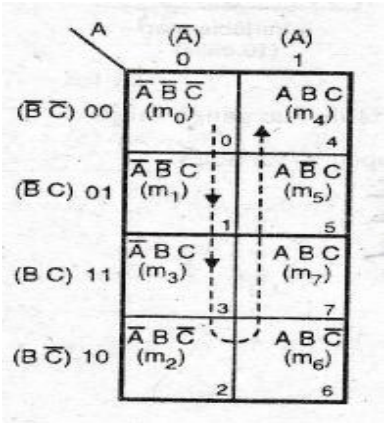
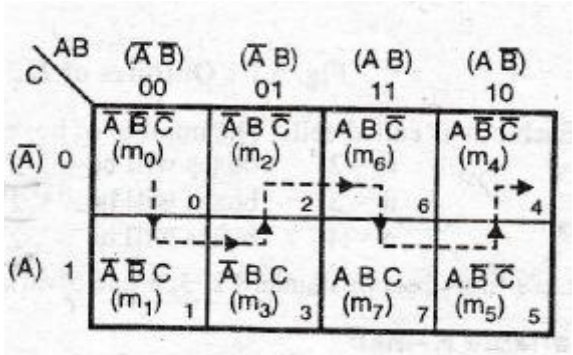
	c) Ans.	**State any two Boolean laws with expression.** 1. A . 0 = 0 2. A . 1 = A (And law) 3. A . A = A 4. A . $\overline{A}$ = 0 5. Commutative Law A. B. = B. A. 6. Associative Law	**2M** ***Any 2 Boolean laws 1M each***

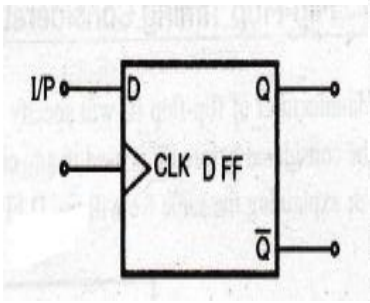
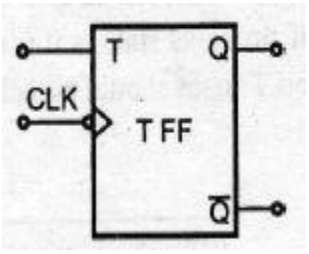
		A. (B.C) = (A.B)C 7. Distributive Law A.(B+C) = A.B + A.C. 8. A.(A+B) = A 9. A. ($\overline{A}$ + B) = AB 10. $\overline{\overline{A}}$ = A 11. De-Morgan's theorem $\overline{A.B} = \overline{A} + \overline{B}$ 12. A + 0 = A 13. A + 1 = 1 $\quad \overline{A}$ + 1 =1) OR law 14. A + A = A 15. A + $\overline{A}$ = 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 + $\overline{A}$B = A + B 21. $\overline{A}$ + AB = $\overline{A}$+ B 22. $\overline{A}$ + A$\overline{B}$= $\overline{A}$ + $\overline{B}$ 23. $\overline{A + B}$ = $\overline{A}$. $\overline{B}$	
	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**

SUMMER – 2019 EXAMINATION
MODEL ANSWER

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

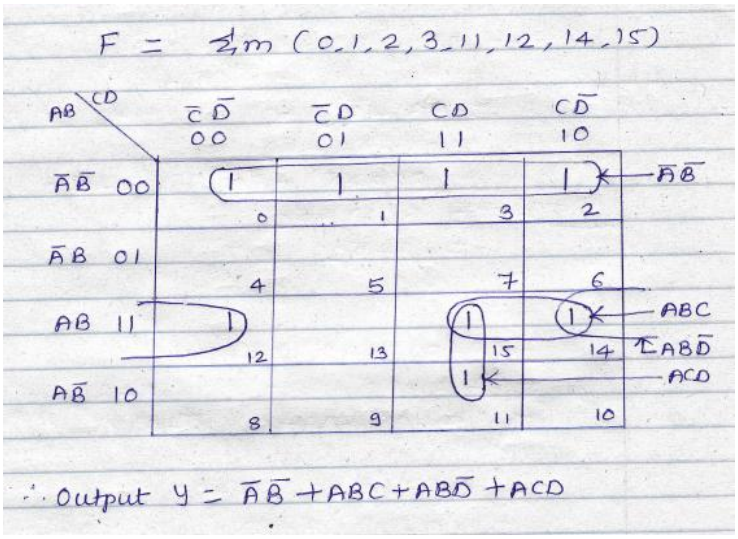
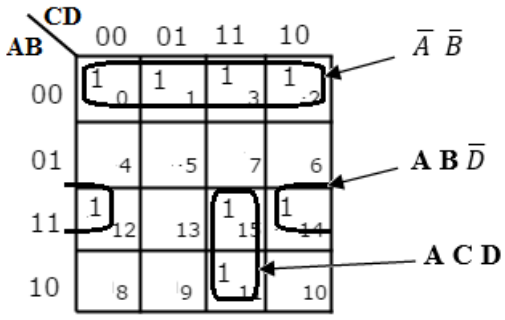
			*1M* *1M*
	f) Ans.	**Define Minterm and Maxterm.** **Minterm:** Each individual term in the canonical SOP (Sum of Products) form is called as Minterm. *Example:*  **Maxterm:** Each individual term in the canonical POS (Product of Sums) form is called as Maxterm. *Example:*	**2M** ***Each definition 1M***

			
	g) Ans.	**Draw three variable K-map format.**  **OR**  **OR** 	**2M** ***Correct diagram 2M***

2	**a)**	**Attempt any <u>THREE</u> of the following:** **Draw symbol and truth table of D and T flip flop. State theie applications.**	**12** **4M**
	Ans.	**D flip flop:**  **Symbol** 	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 | **Truth Table** **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:**  **Symbol** | CLK	T	Q_{n+1}	$\bar{Q}_{n+1}$
↓	0	Q_n	$\bar{Q}_n$
↓	1	$\bar{Q}_n$	Q_n	 **Truth Table** **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)	***D flip flop Symbol - ½M; Truth table- 1M; One applicati on -½M*** ***T flip flop Symbol - ½M; Truth table- 1M; One applicati on -½M***
	b)	**Minimize the following function using K-map.** **F = Σ m (0,1,2,3,11,12,14,15).** ***(Note: Any other equations shall be considered).***	**4M**
	Ans.		

SUMMER – 2019 EXAMINATION
MODEL ANSWER

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

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

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

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

2 | 63		
2 | 31	1	(LSB)
2 | 15	1	
2 | 7	1	
2 | 3	1	
2 | 1	1	
	1	(MSB)

2 | 20		
2 | 10	0	LSB
2 | 5	0	
2 | 2	1	
2 | 1	0	
	1	MSB

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

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

1's complement of $(20)_{10}$: 101011

\+ 1

2's complement of $(20)_{10}$: 101100

$(63)_{10} \Rightarrow$ 1 1 1 1 1 1

\+ $(-20)_{10} \Rightarrow$ 1 0 1 1 0 0

1 0 1 0 1 1 $\Rightarrow (43)_{10}$

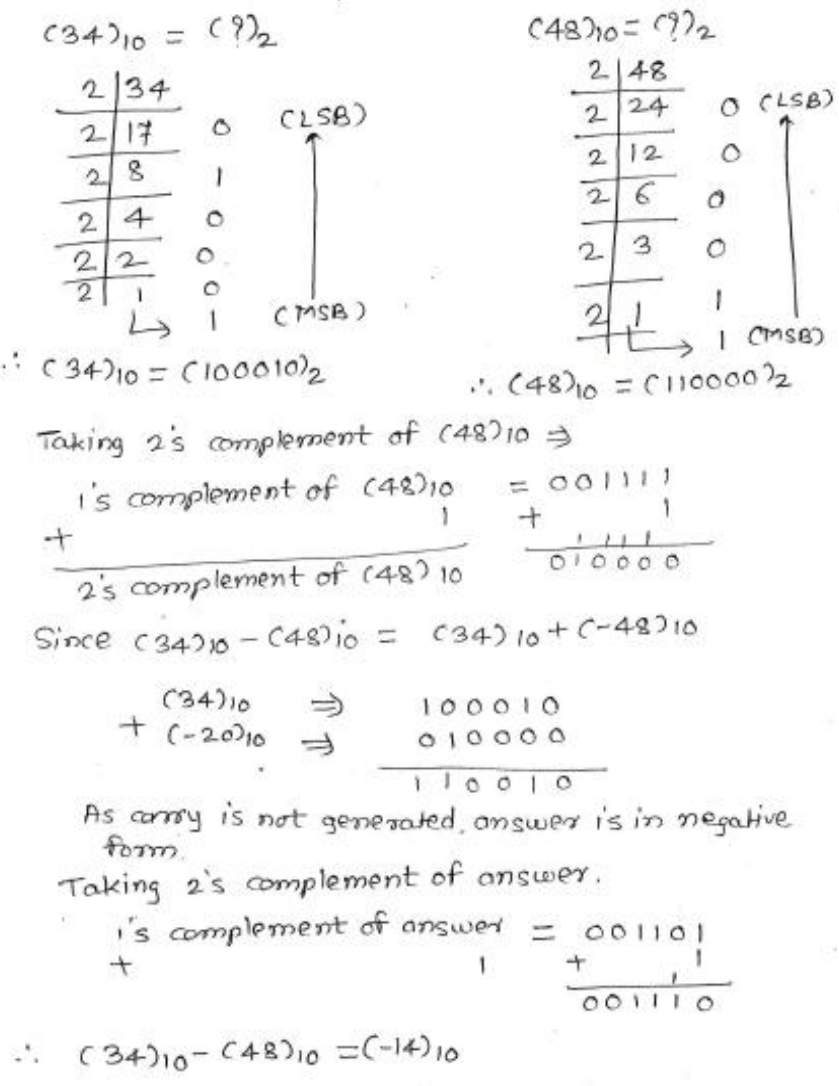
Discard Carry

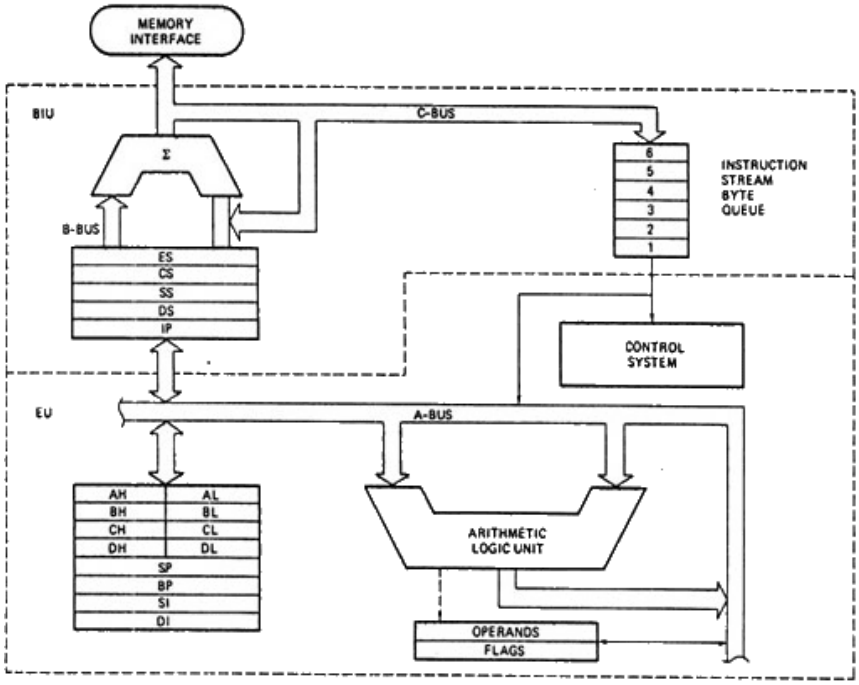
As carry is generated, result is positive and in its true form.

2M

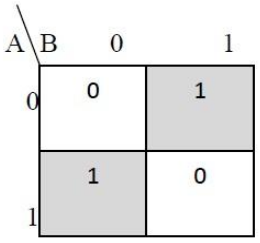
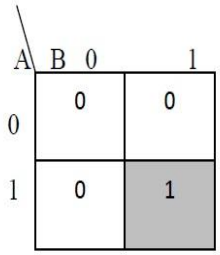
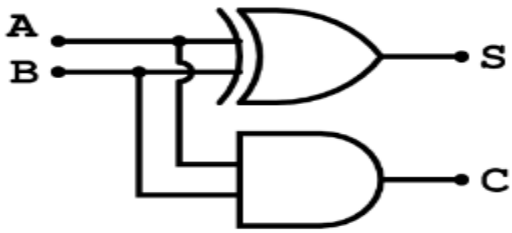
SUMMER – 2019 EXAMINATION
MODEL ANSWER

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

		ii) $(34)_{10}$ - $(48)_{10}$ = ?  	***2M***
	d) **Ans.**	**Simplify the following Boolean expression** **i) $Y = AB + ABC + \bar{A}B + \bar{A}\bar{B}C$** **ii) $Y = (A + B)(A + \bar{B})(\bar{A} + B)$** **Note: Any other method of simplifying using the Boolean laws shall also be considered.** **i) $Y = AB + ABC + \bar{A}B + \bar{A}\bar{B}C$** $= AB(1 + C) + \bar{A}(B + \bar{B}C)$	**4M**

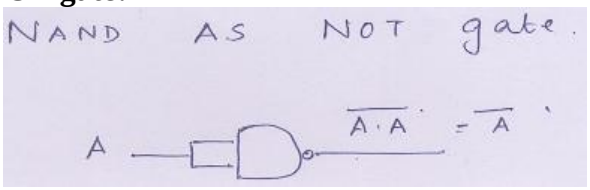
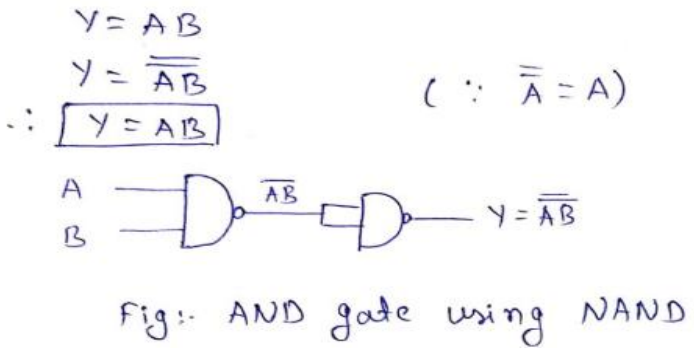
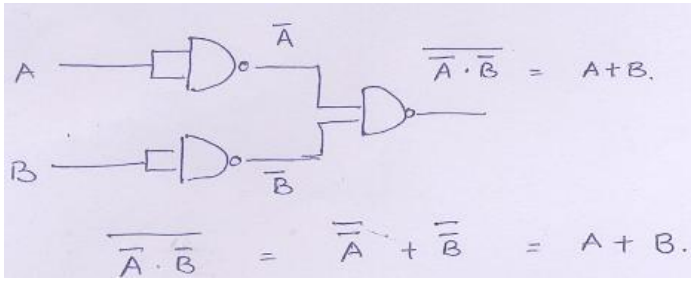
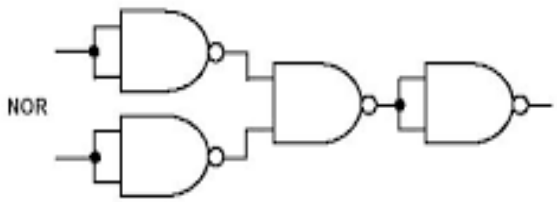
		= AB + $\overline{A}$(B + C) ∵ 1 + C = C, B + $\overline{B}$C = B + C = AB + $\overline{A}$B + $\overline{A}$C = B (A + $\overline{A}$) + $\overline{A}$C ∵ A + $\overline{A}$ = 1 = B (1) + $\overline{A}$C = **B + $\overline{A}$C** **ii) Y = (A + B) (A + $\overline{B}$) ($\overline{A}$ + B)** =(A.A + A $\overline{B}$ + AB + B $\overline{B}$) ($\overline{A}$ + B) = (A + A $\overline{B}$ + AB + 0) ($\overline{A}$+ B) (∵ A.A = A, B $\overline{B}$ =0) = A (1 + $\overline{B}$ + B) ($\overline{A}$ + B) = A (1) ($\overline{A}$ + B) (∵ B+$\overline{B}$=1, 1+A=1) = A ($\overline{A}$ + B) = A $\overline{A}$ + AB = 0 + AB (∵ A $\overline{A}$ = 0) **= AB**	***2M*** ***2M***
3.	**a)** **Ans.**	**Attempt any THREE 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**	**12** **4M** ***Diagram 2M***

		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: A, B — Output: Sum, Carry 0 0 0 0 0 1 1 0 1 0 1 0 1 1 0 1 K map for sum  Sum= $A\bar{B}+A\bar{B}$ K map for Carry  Carry=A.B **Logic Diagram for Half Adder:**  	*1M each for K map of sum and carry* *1M for Logic Diagram*

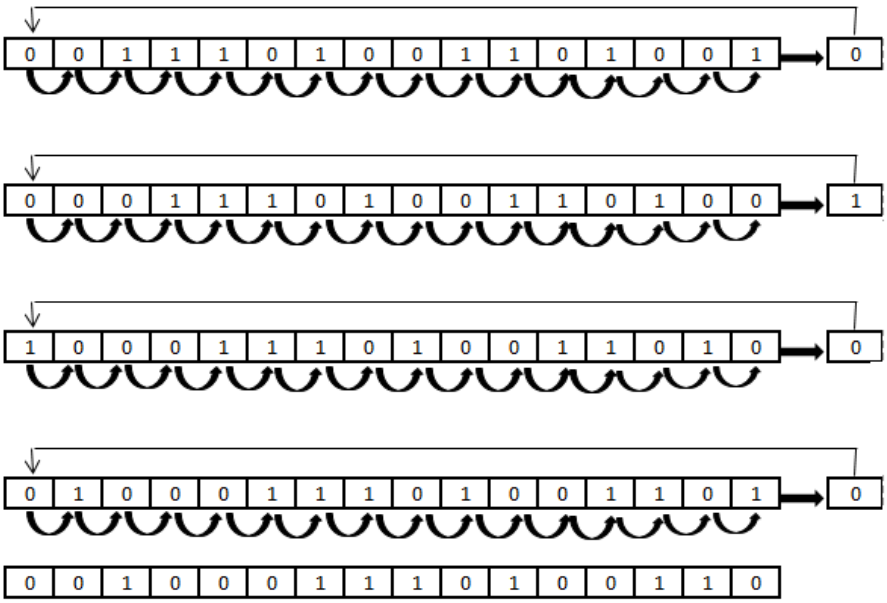
SUMMER – 2019 EXAMINATION
MODEL ANSWER

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

	c) Ans.	**Construct NOT, AND, OR, NOR gates using NAND gate.** **NAND as NOT gate:**  **AND using NAND:**  Fig:- AND gate using NAND **OR using NAND:**  **NOR using NAND:** 	**4M** ***1M for each conversion***

SUMMER – 2019 EXAMINATION
MODEL ANSWER

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

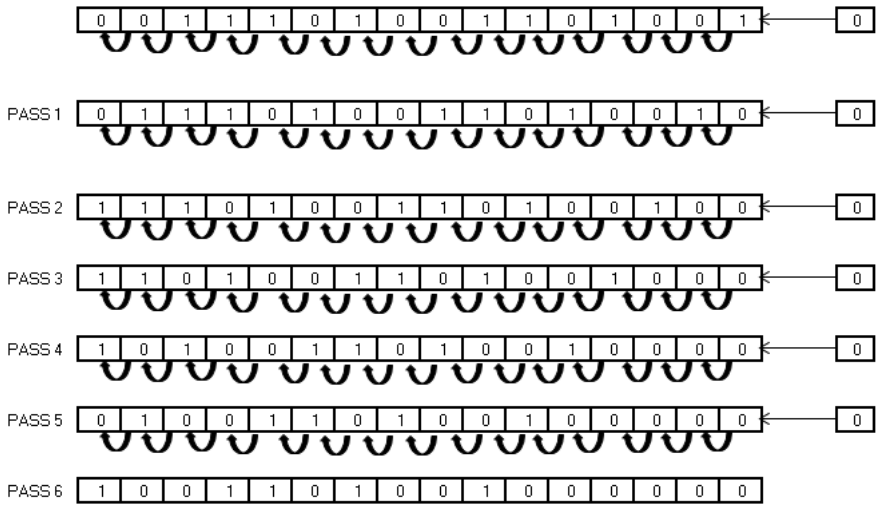
	d)	**Interpret the given program and specify the output for the following situation.** **MOV AX, 34F9H** **MOV BX, 3A69H.** **(i) Masking of lower nibble of AX.** **(ii) Rotate right through carry contents of BX by 4 positions.** **(iii) Shift left contents of BX by 6 positions..** **(iv) XOR AX, BX** ***(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).***	**4M**
	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***

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



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

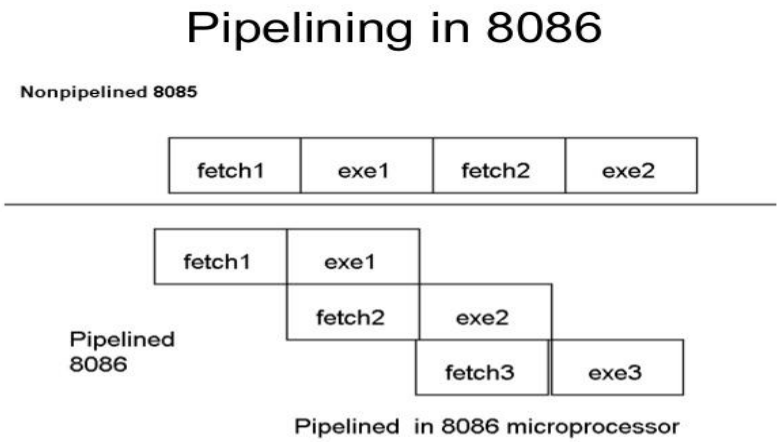
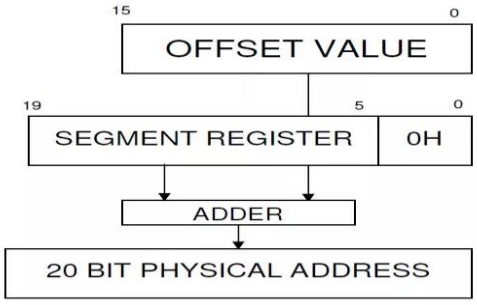
After the Execution of the instruction Register AX will contain data **0E90H**

AX= 0E90H

4. a) Ans. **Attempt any THREE of the following:** **12**

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

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.  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.  **Fig: Physical address formation** Calculate the physical address for the given CS=3420H, IP=689AH. CS=3420H	**4M** ***2M for explaination*** ***1M diagram***

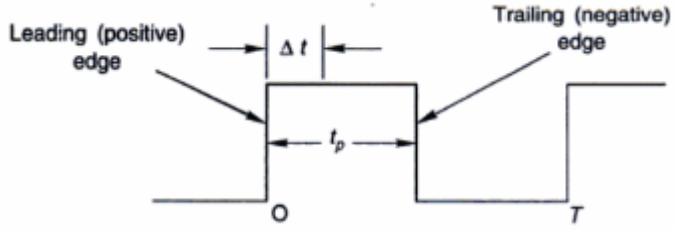
		IP=689AH Zero is inserted 3 4 2 0 0 + 6 8 9 A **= 3 A A 9 A**	***1M for example***
	c) Ans.	**State and prove De-Morgan's Theorems.** **Theorem no 1:** It states that the, complement of a sum is equal to product of their complements **Verification of the second theorem :** $\overline{A+B}$ LHS $\overline{A+B}=\bar{A}\cdot\bar{B}$ 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. LHS $\overline{AB}=\bar{A}+\bar{B}$ RHS **: Verification of the theorem $\overline{AB}=\bar{A}+\bar{B}$**	**4M** ***For each theorem 2M***

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

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

SUMMER – 2019 EXAMINATION
MODEL ANSWER

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

	d) **Ans.**	**Describe race-around condition in JK flip flop and suggest ways to overcome it.** **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. 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.  	**4M** ***2M for description*** ***2M for suggestion***
	e) **Ans.**	**Compare combinational and sequential circuits (four points).** **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	**4M** ***Any four points 1M each***
5.	**a)** **Ans.**	**Attempt any <u>TWO</u> of the following:** **Write an assembly language program with algorithm for finding smallest number from the array of 10 numbers (Assume suitable data).** ***(Note: Any other logic shall be considered).***	**12** **6M**

		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.
		
		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
			Algorithm 2M
			
			Correct Program 4M

	b) **Ans.**	**Draw minimum mode configuration of 8086 and explain the function of any four control signals.**	**6M**

SUMMER – 2019 EXAMINATION
MODEL ANSWER

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

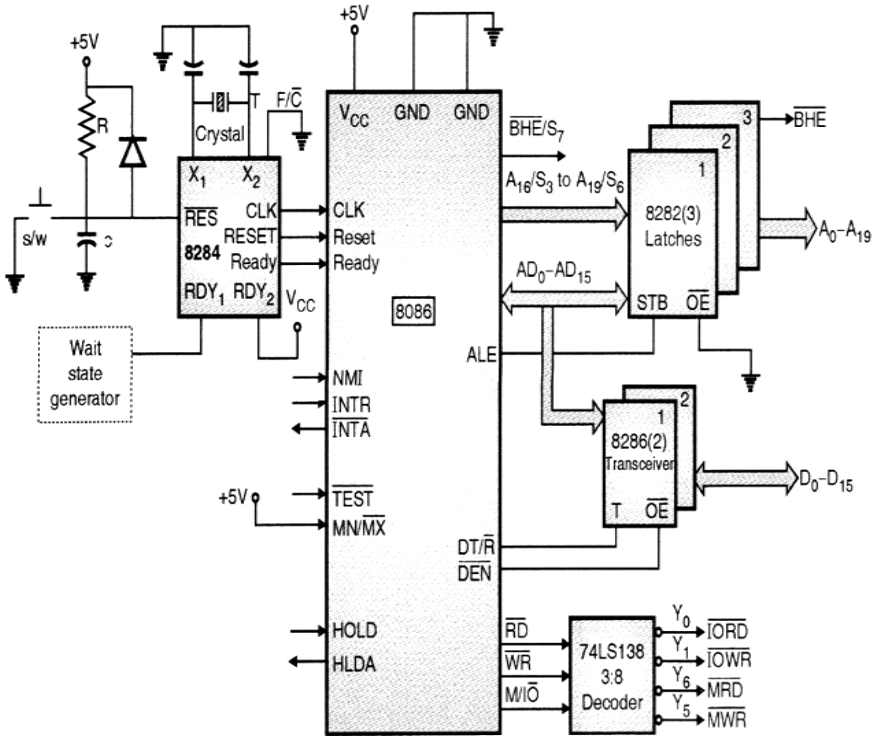


Diagram 4M

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]

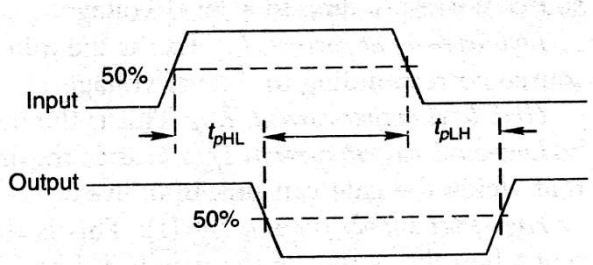
   ii. Based Addressing with displacement: In this mode EA is sum of an 8/16 bit displacement and the contents of base register (BX or BP).
      Ex: MOV AX, [BX+300H]; MOV AX, [BX-2H]

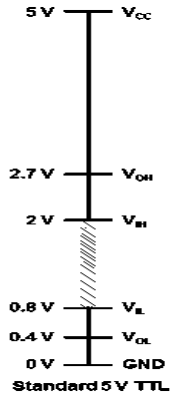
   iii. Indexed Addressing Mode: In this EA is the sum of the 8/16 bit displacement plus the contents of the index registers SI or DI.
      Ex: MOV [DI + 2345H], 1234H; MOV AX, [SI + 45H]

   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

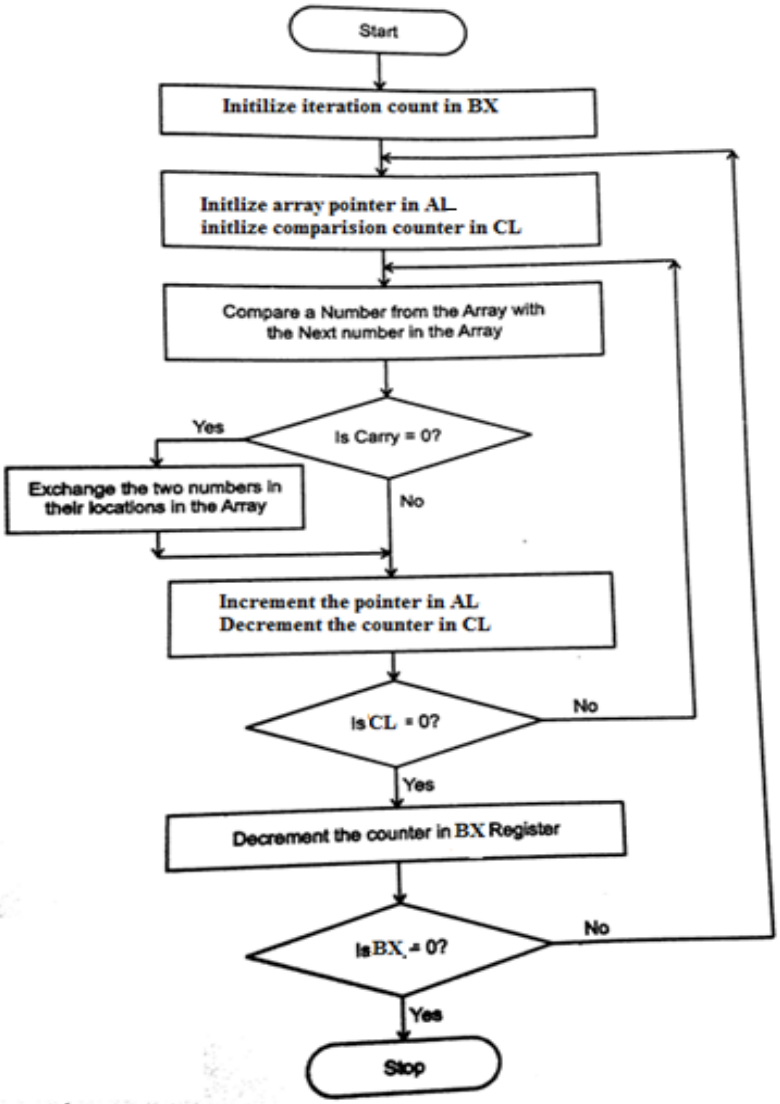
		address will be provided in the instruction. This allows fixed access to ports numbered 0 to 255 (00-FFH). Ex: OUT 06H, AL; IN AX, 85H ii. Variable or Indirect Port Addressing: In this case port address will not be explicitly in the instruction. The address of port number is taken from DX allowing 64K 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)**	**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.**	**12** **6M**
	Ans.	**(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	***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

SUMMER – 2019 EXAMINATION
MODEL ANSWER

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

			***Flowchart 2M***
	c)	**Refer given Fig. No.1 and write the outputs for each of the following input:**	**6M**

Ans.

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	

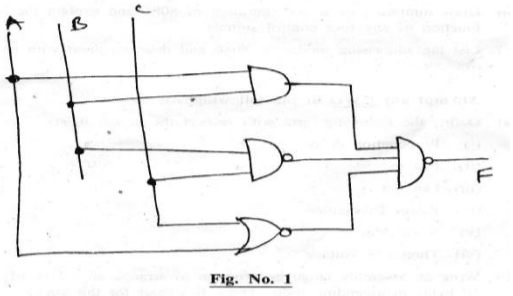


Fig. No. 1

(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

V2V EDTECH LLP

Online Coaching at an Affordable Price.

OUR SERVICES:

- Diploma in All Branches, All Subjects
- Degree in All Branches, All Subjects
- BSCIT / CS
- Professional Courses

+91 93260 50669

v2vedtech.com

V2V EdTech LLP

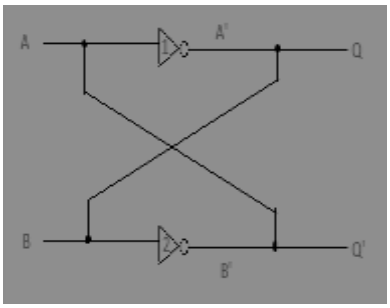
v2vedtech

WINTER – 19EXAMINATION

Subject Name: Digital Techniques and Microprocessor Model Answer Subject Code: 22323

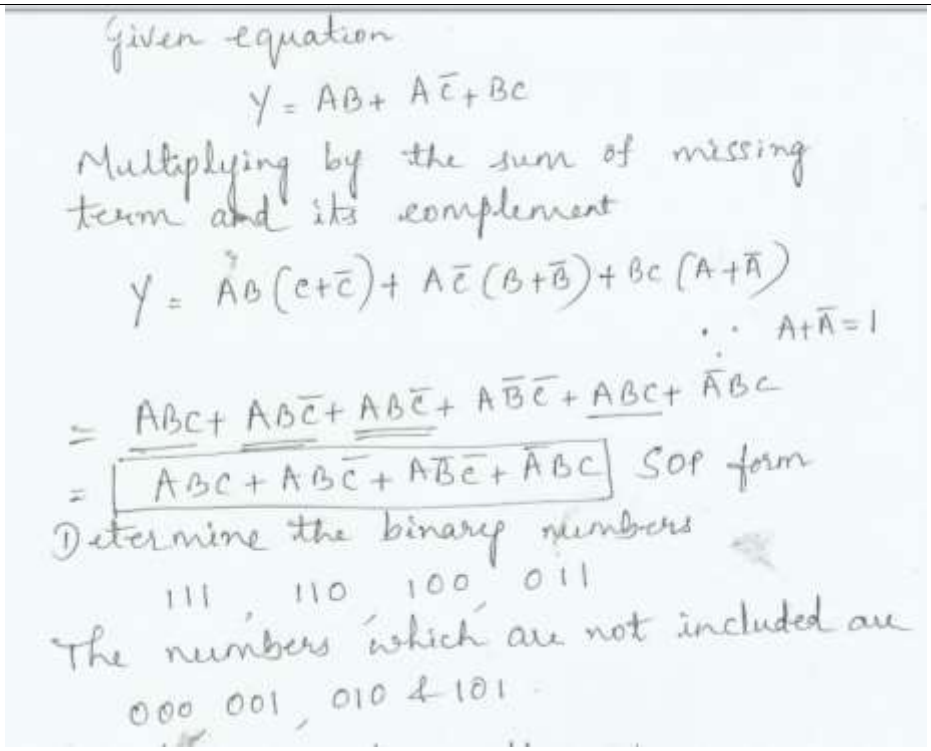
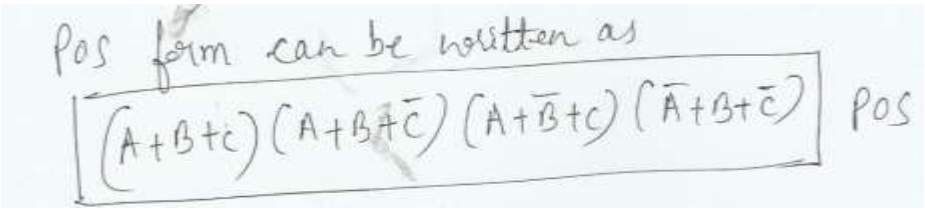
Important Instructions to examiners:

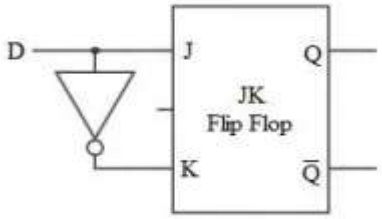
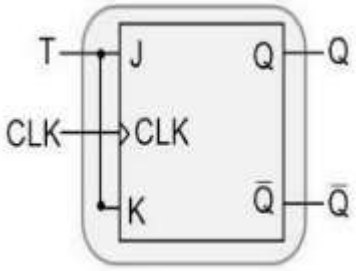
1) The answers should be examined by key words and not as word-to-word as given in themodel answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may tryto assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given moreImportance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in thefigure. 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 constantvalues 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.

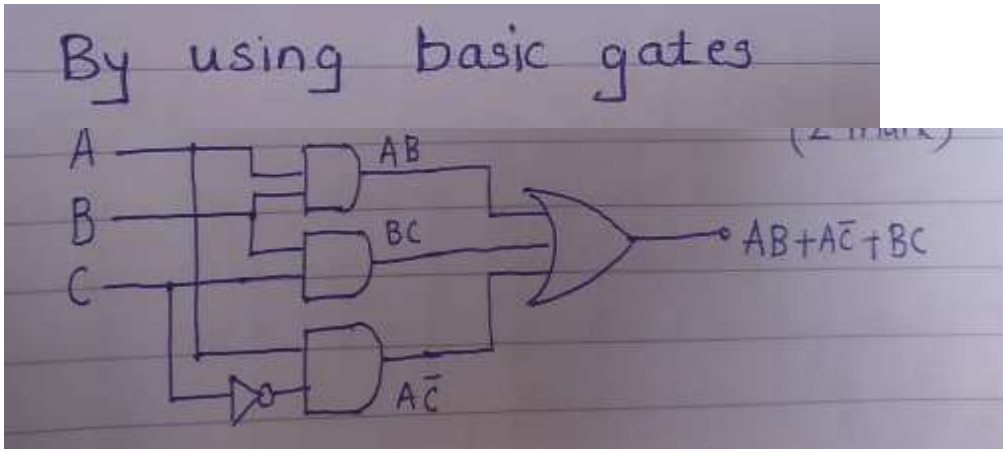
Q. No.	Sub Q. N.	Answer	Marking Scheme
Q.1		**Attempt any FIVE of the following:**	**10-Total Marks**
	a)	**List one application of each of following** **(i) Gray code** **(ii) ASCII code**	**2M**
	Ans:	(i) Gray codes are used for error correction in digital communication system. (ii)ASCII codes are used for identifying characters and numerals in a keyboard.	**1M Each**
	b)	**State the principle of multiplexer and mention its two types.**	**2M**
	Ans:	**Principle of multiplexer** Multiplexer is a circuit with many inputs and only one output. By applying control signals on select lines we can direct any input to the output. Types 4:1,16:1 etc	**1M** **1M**
	c)	**Draw the circuit of one bit memory cell.**	**2M**
	Ans:	Circuit : 	**2M**
	d)	**List features of 8086 microprpcessor.(Any four)**	**2M**

	Ans:	**(Note: Consider any 4)** Features of 8086 microprpcessor 1)It requires +5v power supply. 2)It has 20 bit address bus,can acceess 2^{20} =1MB memory location. 3)16 bit data bus. 4)It is a 16 bit processor having 16 bit ALU,16 bit registers. 5)It has instruction queue which is capable of storing 6 instruction bytes from the memory for faster processing. 6)It has pipelining,fetch and execute stagefor improving performance. 7)It has 256 vectored interrupts. 8)Clock range is 5-10 MHz.	**2M**
	e)	**Convert the following numbers into Hexadecimal number.** **(i) $(10110111)_2 = (?)_{16}$** **(ii) $(567)_8 = (?)_{16}$**	**2M**
	Ans:	(i) $(10110111)_2 = (B7)_{16}$ (ii) $(567)8 = (177)_{16}$	**2M**
	f)	**State four characteristics of RISC processor.**	**2M**
	Ans:	1)Reduced instruction set. 2)Simple addressing mode. 3)RISC processor consumes less power and has high performance. 4)Instruction is of uniform fixed length. 5)Large number of registers.	**2M**
	g)	**Give example of any two types of addressing mode of 8086**	**2M**
	Ans:	1)Direct addressing mode Eg:MOV CL,[1234] 2)Immediate addressing mode Eg:MOV AX,0005H 3)Register addressing mode Eg: MOV AX,BX 4)Base indexed addressing mode Eg:MOV CL,[BX+SI]	**Any two 1M each**

Q.2		**Attempt any THREE of the following:**	**12-Total Marks**
	a)	**Perform the following subtaction using 1's compliment and 2's compliment $(1010\ 0101)_2 - (1110\ 1110)_2$.**	**4M**
	Ans:	Subtaction using 1's compliment $(1010\ 0101)_2 - (1110\ 1110)_2$. Find 1's complement of the subtrahend 00010001 Add minuend 00010001 + 10100101	

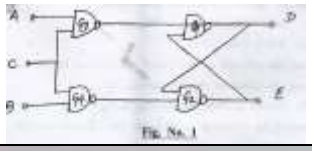
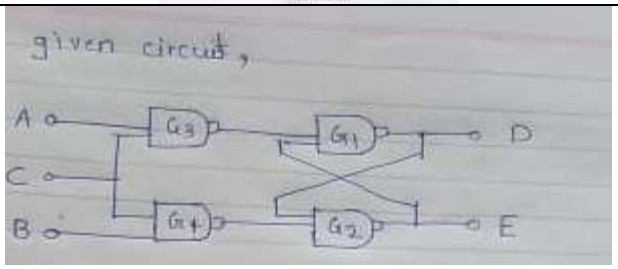
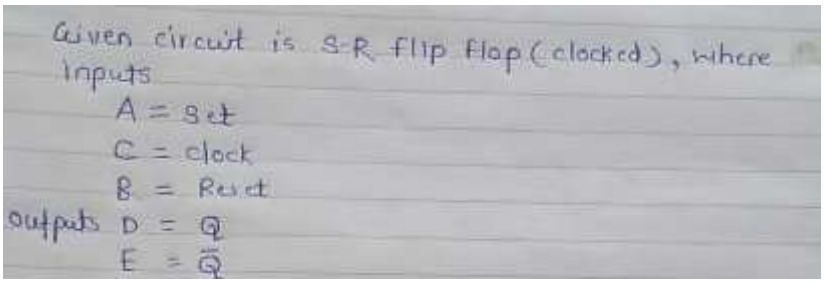
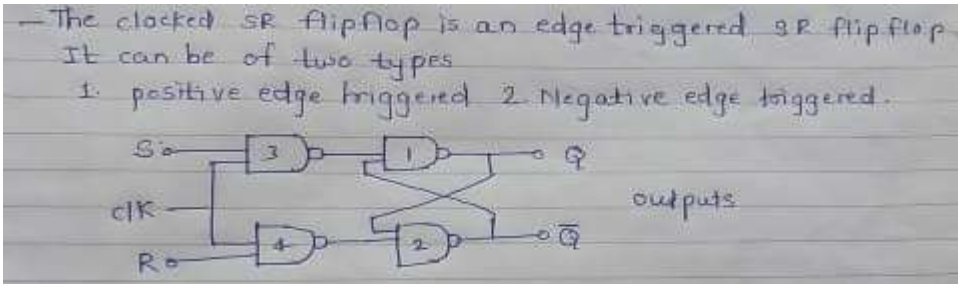
		10110110 Since carry is 0,the resullt is –ve and in 1's complement form. The answer is -01001001 Subtaction using 2's compliment Find 1's complement of the subtrahend 00010001+1=00010010+ Add minuend 10100101 10110111 Since there is no carry,answer is –ve and is in its 2's complement form. The answer is -01001001	
	b)	**Simplify the given equation into standard SOP form Y = AB +A$\overline{C}$+ BC and represent the same equation in standard POS form.**	**4M**
	Ans:	 	**2M** **2M**
	c)	**Differentiate between D FF and T FF.**	**4M**

	Ans:			**4M**
		Input is transferred after a delay	When T=1, output toggles	
		Used in shift registers	Used in counters, frequency dividers	
		D Qn+1 0 0 **1 1**	T Qn+1 0 Qn **1 $\overline{\mathbf{Qn}}$**	
	d)	**Describe the characteristics of digital IC's (Any four).**		**4M**
	Ans:	Characteristics of digital IC's are 1)Fan out:It is the number of loads that the output of the gate can drive. 2)Power dissipation:Power consumed by the gate when fully driven by all its inputs. 3)Propagation delay:Time for the signal to propagate from input to output.\ 4)Noise margin:The maximum noise voltage added to an input signal that does not cause undesirable change in output. Fan in:It is the number of inputs connected to the gate without any degradation in the voltage level. Operating Temperature:It is the range of temperature in which the performance of IC is effective. Figure of merit:It is the product of speed and power.		**1M each**
Q.3		**Attempt any THREE of the following:**		**12-Total Marks**
	a)	**Reduce the following Boolean expression using laws of Boolean algebra and realize using logic gates.** **$Y = (A + BC)(B + \overline{C}A)$**		**4M**

	Ans:	Given $Y = (A+BC)(B+\bar{C}A)$ $= AB + A\bar{C}A + BCB + BC\bar{C}A$ [AND laws $= AB + A\bar{C} + BC + 0$ → $A \cdot A = A$ $= AB + A\bar{C} + BC$ $B \cdot B = B$ $C \cdot \bar{C} = 0$] 	**2M** **2M**
	b)	**Write an assembly language program to transfer block of 10 numbers from one memory location to another.** **(Assume suitable data.)**	**4M**

	Ans:	.model small
		.data
		src_arr dw 1,2,3,4,5,6,7,8,9,10
		dst_arr dw 10(dup) ;empty array
		.code
		mov ax,@data ;initialize data
		mov ds, ax
		mov cx,10 ; initialize counter
		mov si,offset src_arr ;initialize memory pointer for source
		mov di,offset dst_arr ;initialize memory pointer for destination
		up:
		mov ax,[si] ;read number from source array
			4M

		mov [di] , ax ;write number to destination array
		add si,2 ;increment source memory pointer
		add di,2 ;increment destination memory pointer
		loop up ;check word counter for zero ,if not zero then read up number from array
		ends
		end	

	c)	**For the given circuit, identify the inputs and outputs. Name the circuit and draw its truth table. Refer Fig .No. 1.** 	**4M**
	Ans:	   	**4M**

Truth table

	Inputs		outputs		Remark
CIK	S	R	Q_{n+1}	$\bar{Q}_{n+1}$	
↑	0	0	Q_n	$\bar{Q}_n$	No change
↑	0	1	0	1	Reset
↑	1	0	1	0	Set
↑	1	1	Race	Race	Avoid

	d)	**Simplify the given K-map using standard form and realize the circuit using gates. Refer Fig. No. 2.**	**4M**

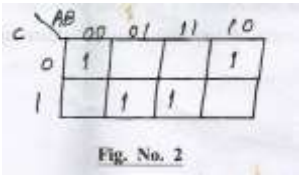
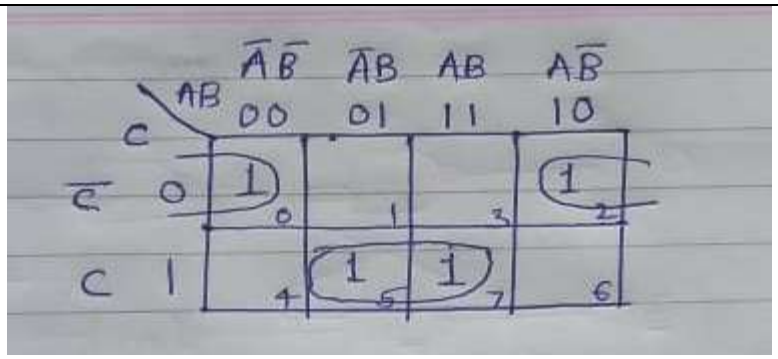
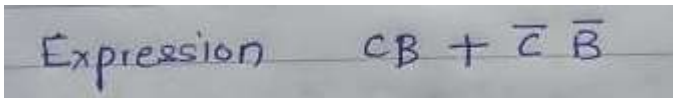
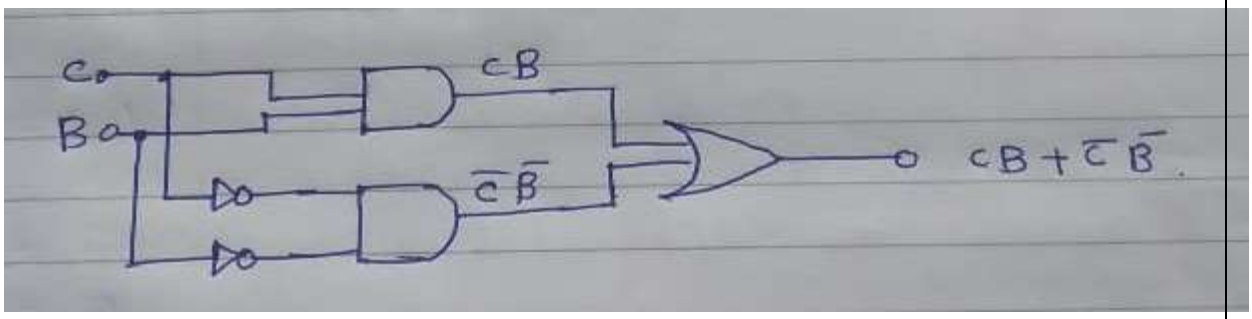


Fig. No. 2

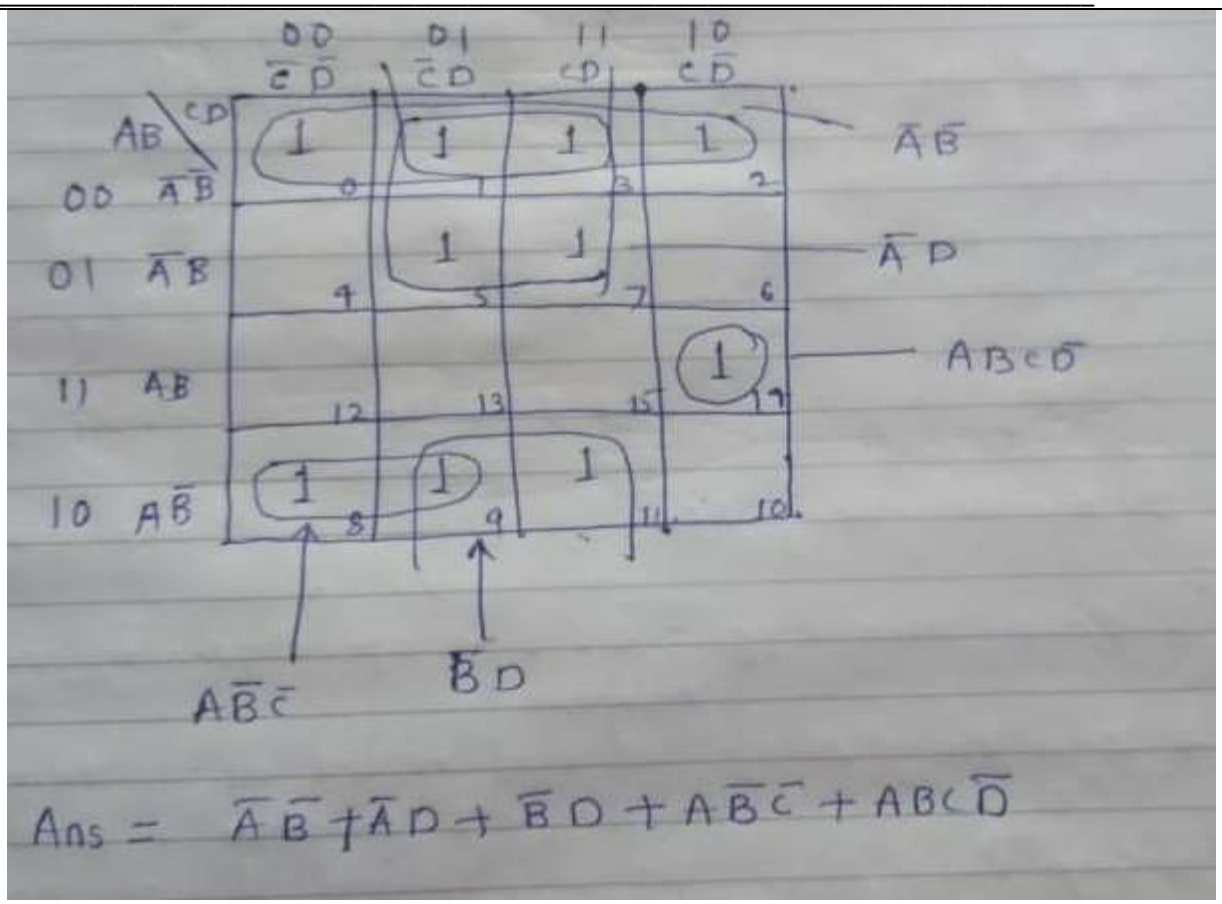
	Ans:		**2M**
			**2M**
			

Q.4		**Attempt any THREE of the following :**	**12-Total Marks**
	a)	**Write an assembly language program to find the sum of series of ten numbers stored in memory.(Assume suitable data.)**	**4M**
	Ans:	Sum Of Series:	**4M**

```
.model small
.data
    array   db   0ffh, 11h, 22h, 33h, 44h, 55h, 66h, 77h, 88h,
                 99h.
    Sum_lsb db   0
    Sum_msb db   0
.code
    mov ax,@data          ; Initialize data segment
    mov ds, ax
    mov cx, 10            ; initialize byte counter
    mov si, offset array  ; initialize memory pointer,
                          from
up:
    mov al, [si]          ; Read byte^ memory
    add sum_lsb, al       ; add with sum
    jnc next              ; if sum > 8 bit
    inc Sum_msb           ; increment msb counter
next:
    inc si                ; Increment memory pointer
    loop up               ; decrement byte counter
                          ; if byte counter=0 then exit
                           else read next number

    ends
end
```

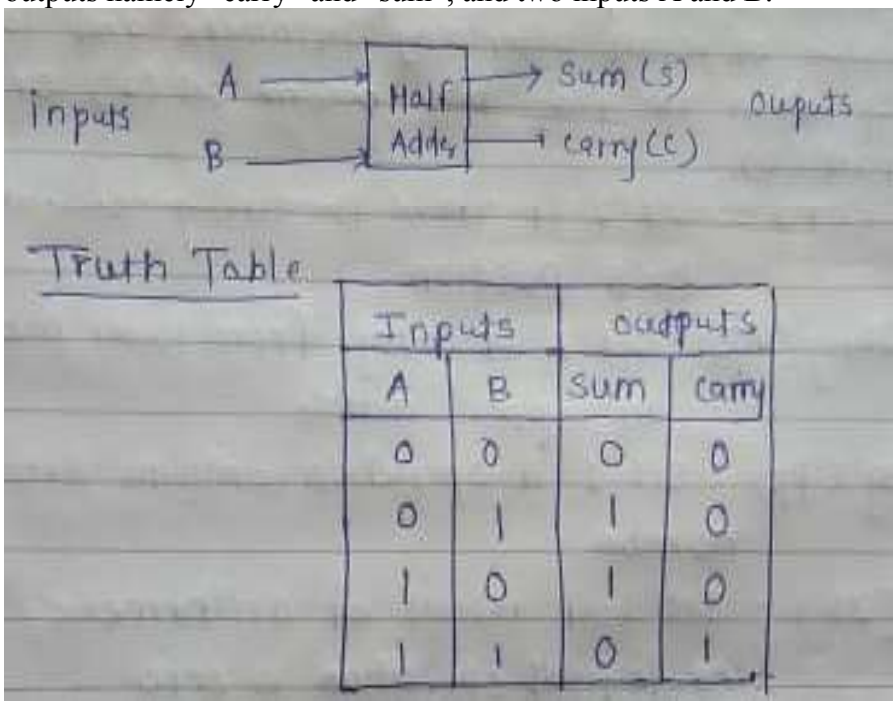
	b)	**Minimize the four variable logic function using K- map.** **F(A,B,C,D) Σm (0, 1, 2, 3, 5, 7, 8, 9, 11, 14)**	**4M**
	Ans:	f(A,B,C,D) Σm(0,1,2,3,5,7,8,9,11,14)	**4M**

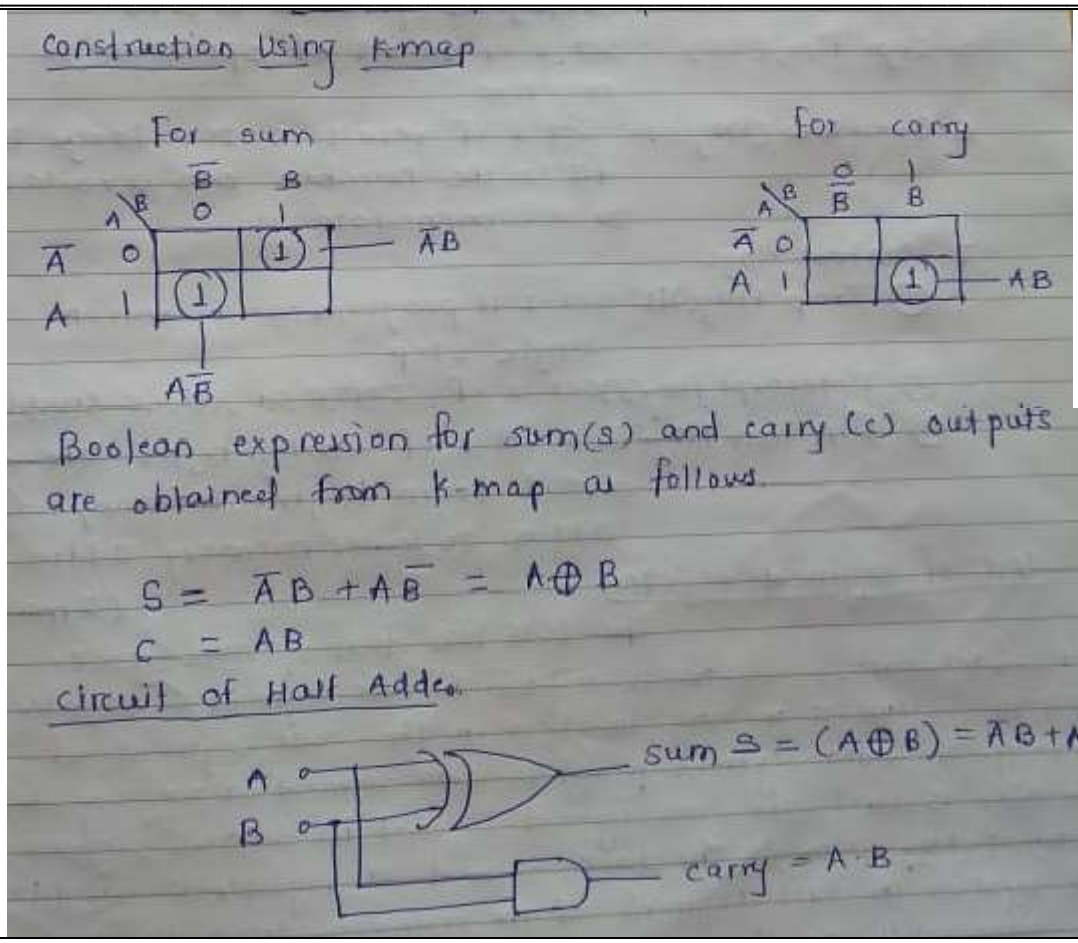


	c)	**Differentiate between sequential and combinational logic circuits. (Any four points)**	**4M**
	Ans:		**4M**

Sr.no	parameter	Sequential circuit	Combinational circuit
1	Output depends on	present inputs and past inputs/outputs	Input present at that instant of time
2.	Memory	Necessary	Not necessary
3.	clock input	Necessary	Not necessary
4	Example	Flip flops, shift registers, counters	Adder, substractors, code converters

	d)	**Describe the use of flag register and segment registers in 8086.**	**4M**

	Ans:	• Use of Flag Register: Microprocessor 8086 has 16 bit flag register among which 9 bits are active. The purpose of flag register is to indicate the status of the processor Depending upon the value of result after any arithmetic and logical operation the flag bits become set (1) or reset (0). 1. Carry Flag (CF): Set 1 if there is carry out of MSB position. 2. Auxiliary Flag (AF): Set 1 if carry from lower nibble to upper nibble. 3. Parity Flag (PF): Set 1 if operation contains even number. 4. Zero Flag (ZF): Set 1 if result of arithmetic or logical operation is zero. 5. Sign Flag (SF): Set 1 if result of operation is negative. 6. Overflow Flag (OF): Set 1 if result is too large to fit in the numbers bits available to accommodate it. 7. Control Flags: (i) Trap Flag (TF): Set 1 if program can be run in single step. (ii) Interrupt Flag (IF): Set 1 if INTR of 8086 is enabled. (iii) Direction Flag (DF): Set 1 if string bytes are write or read from higher memory address to lower memory address. • Use of Segment Register: The 8086 has four segment register of 16 bit each. i.e. CS,DS,SS and ES. The code segment CS register used to address a memory location in the code segment of memory. The data segment point to data segment of memory where the data is stored the extra segment ES used to address the segment is additional data segment. The Stack segment SS register is used to point location in stack segment of the memory, used to store data temporarily on the stack.	**2M** **2M**
	e)	**Describe the construction of half adder using K – map.**	**4M**
	Ans:	Half Adder using k-map: Half adder is a combinational logic circuit with two inputs and two output , circuit has two outputs namely “carry” and “sum”, and two inputs A and B.  Truth Table Inputs: A, B — outputs: sum, carry 0 0 0 0 0 1 1 0 1 0 1 0 1 1 0 1	**2M**

		Construction Using K-map  Boolean expression for sum(S) and carry (C) outputs are obtained from K-map as follows. $S = \bar{A}B + A\bar{B} = A \oplus B$ $C = AB$ Circuit of Half Adder. 	**2M**
Q.5		**Attempt any TWO of the following**	**12-Total Marks**
	(a)	**Write an assembly language program to find the factorial of a number using looping process.**	**6M**

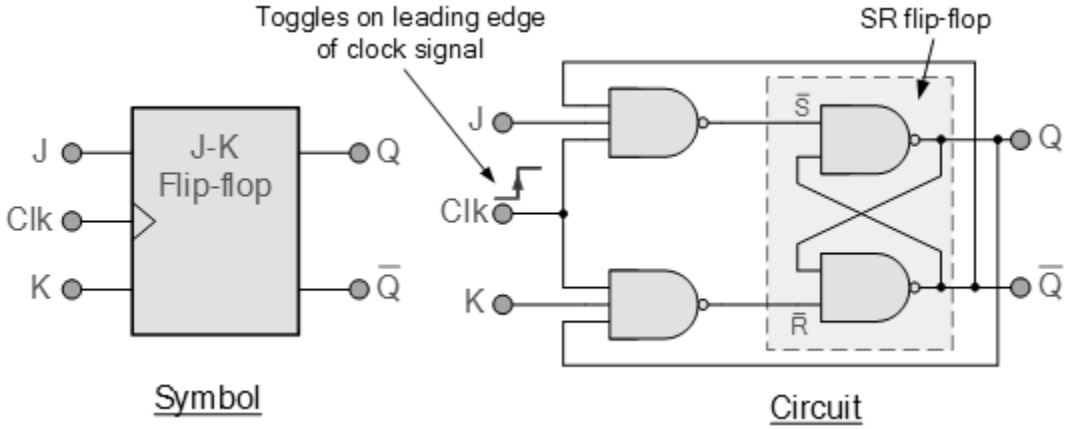
	Ans:	DATA SEGMENT
		A DW0005H
		FACT_LSBDW?
		FACT_MSBDW?
		DATA ENDS
		CODE SEGMENT
		ASSUME DS:DATA,CS:CODE
		START:MOVAX,DATA
		MOV DS,AX
		CALL FACTORIAL
		MOVAH,4CH
		INT 21H
		FACTORIAL PROC
		MOV AX,A
		MOV BX,AX
		DEC BX
			6M

		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
		OR
		DATA SEGMENT
		NUM DB 05H
		RES DW ?
		DATA ENDS
		CODE SEGMENT
		ASSUME CS:CODE,DS:DATA
		START:
		MOVAX,DATA
		MOV DS,AX
		CALL FAC
		MOV RES,AX
		MOVAH,4CH
		INT 21H
		FAC PROC
		MOVCL,NUM
		DEC CL
		MOV AL,NUM
		MOVAH,00H
		MOV BL,CL
		MOV BH,00H
		L1: MUL BX
		DEC BX
		DEC CL
		JNZ L1
		RET
		FAC ENDP
		CODE ENDS
		END START
		Correct Program with any other logic can be given marks.	

	(b)	**Describe the principle of working of JK FF and draw its circuit diagram and truth table.**	**6M**
	Ans:	The **JK flip flop** is basically a gated SR flip-flop with the addition of a clock input circuitry that prevents the illegal or invalid output condition that can occur when both inputs S and R are equal to logic level "1". Due to this additional clocked input, a JK flip-flop has four	

possible input combinations, "logic 1", "logic 0", "no change" and "toggle". The symbol for a JK flip flop is similar to that of an *SR Bistable Latch* as seen in the previous tutorial except for the addition of a clock input.

The Basic JK Flip-flop



2M

Both the S and the R inputs of the previous SR bistable have now been replaced by two inputs called the J and K inputs. Then this equates to: J = S and K = R.

The two 2-input AND gates of the gated SR bistable have now been replaced by two 3-input NAND gates with the third input of each gate connected to the outputs at Q and Q. This cross coupling of the SR flip-flop allows the previously invalid condition of S = "1" and R = "1" state to be used to produce a "toggle action" as the two inputs are now interlocked.

If the circuit is now "SET" the J input is inhibited by the "0" status of Q through the lower NAND gate. If the circuit is "RESET" the K input is inhibited by the "0" status of Q through the upper NAND gate. As Q and Q are always different we can use them to control the input. When both inputs J and K are equal to logic "1", the JK flip flop toggles as shown in the following truth table.

The Truth Table for the JK Function

CLK	J	K	Q_{n+1}	$\overline{Q}_{n+1}$	Description
1 , 0 , ↑	X	X	Q_n	$\overline{Q}_n$	No Change
↓	0	0	Q_n	$\overline{Q}_n$	No Change
↓	0	1	0	1	Reset Condition
↓	1	0	1	0	set Condition
↓	1	1	$\overline{Q}_n$	Q_n	Toggle condition

2M

Then the JK flip-flop is basically an SR flip flop with feedback which enables only one of its two input terminals, either SET or RESET to be active at any one time thereby eliminating the invalid condition seen previously in the SR flip flop circuit.

Also when both the J and the K inputs are at logic level "1" at the same time, and the clock input is pulsed "HIGH", the circuit will "toggle" from its SET state to a RESET state, or visa-versa. These results in the JK flip flop acting more like a T-type toggle flip-flop when both terminals are "HIGH".

2M

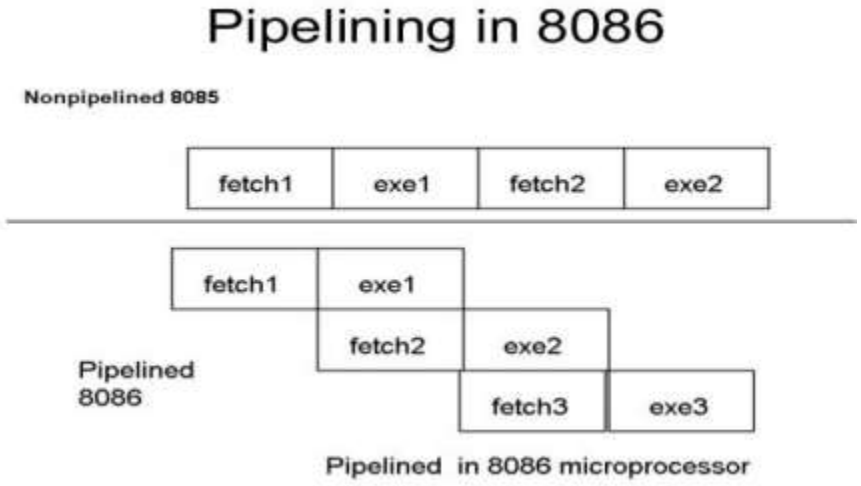
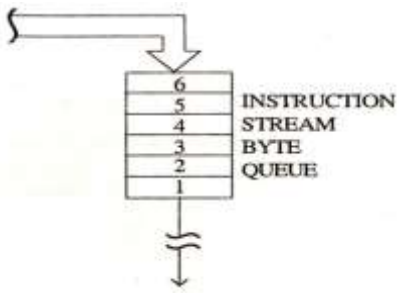
Although this circuit is an improvement on the clocked SR flip-flop it still suffers from timing problems called "race" if the output Q changes state before the timing pulse of the clock input has time to go "OFF". To avoid this the timing pulse period (T) must be kept as short as possible (high frequency). As this is sometimes not possible with modern TTL IC's the much improved **Master-Slave JK Flip-flop** was developed.

	(c)	**Differentiate between CISC and RISC and justify use of each of them in practice.**	**6M**
	Ans:	The architecture of the Central Processing Unit (CPU) operates the capacity to function from	**2M**

"Instruction Set Architecture" to where it was designed. The architectural design of the CPU is Reduced instruction set computing (RISC) and Complex instruction set computing (CISC). CISC has the capacity to perform multi-step operations or addressing modes within one instruction set. It is the CPU design where one instruction works several low-level acts. For instance, memory storage, loading from memory, and an arithmetic operation. Reduced instruction set computing is a Central Processing Unit design strategy based on the vision that basic instruction set gives a great performance when combined with a microprocessor architecture which has the capacity to perform the instructions by using some microprocessor cycles per instruction. The hardware part of the Intel is named as Complex Instruction Set Computer (CISC), and Apple hardware is Reduced Instruction Set Computer (RISC).

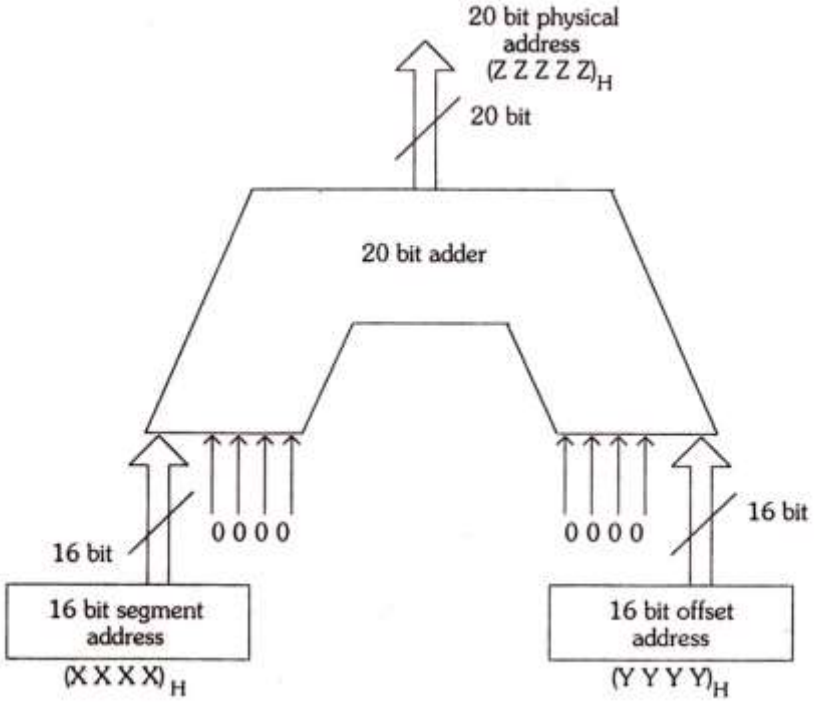
Sr. No.	CISC	RISC
1	A large number of instructions are present in the architecture.	Very fewer instructions are present. The number of instructions are generally less than 100.
2	Some instructions with long execution times. These include instructions that copy an entire block from one part of memory to another and others that copy multiple registers to and from memory.	No instruction with a long execution time due to very simple instruction set. Some early RISC machines did not even have an integer multiply instruction, requiring compilers to implement multiplication as a sequence of additions.
3	Variable-length encodings of the instructions.	Fixed-length encodings of the instructions are used.
4	Example: IA32 instruction size can range from 1 to 15 bytes.	Example: In IA32, generally all instructions are encoded as 4 bytes.
5	Multiple formats are supported for specifying operands. A memory operand specifier can have many different combinations of displacement, base and index registers.	Simple addressing formats are supported. Only base and displacement addressing is allowed.
6	CISC supports array.	RISC does not supports array.
7	Arithmetic and logical operations can be applied to both memory and register operands.	Arithmetic and logical operations only use register operands. Memory referencing is only allowed by load and store instructions, i.e. reading from memory into a register and writing from a register to memory respectively.
8	Implementation programs are hidden from machine level programs. The ISA provides a clean abstraction between programs and how they get executed.	Implementation programs exposed to machine level programs. Few RISC machines do not allow specific instruction sequences.
9	Condition codes are used.	No condition codes are used.
10	The stack is being used for procedure arguments and return addresses.	Registers are being used for procedure arguments and return addresses. Memory references can be avoided by some procedures.

4M

Q.6		Attempt any TWO of the following:	12Total Marks
	(a)	**Describe the concept of pipelining and process of physical address generation in 8086 microprocessor.**	6M
	Ans:	CONCEPT OF PIPELINING Fetching the next instruction while the current instruction executes is known as pipelining it means 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.  Pipelined in 8086 microprocessor To speed up program execution, the Bus Interface Unit(BIU) fetches as many as 6 instruction bytes ahead of time from the memory and these are held for execution unit in the (FIFO) group of registers called QUEUE. The BIU can fetch instruction bytes while EU is decoding or executing an instruction which does not require the use of buses. When the EU is ready for the next instruction, it simply reads the instruction from the QUEUE in the BIU. This is much faster than sending out addresses to system memory and waiting for the memory to send back the next instruction byte. The Queue is refilled when at least two bytes are empty as 8086 has a 16-bit data bus. In case of Branch instructions however, the instructions pre-fetched in the queue are of no use. Hence the QUEUE has to be dumped and new instructions are fetched from the destination addresses specified by the branch instructions.  	3M

Physical Address Generation:

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.



3M

The physical address is calculated as follows:

X X X X 0

\+ 0 Y Y Y Y

Z Z Z Z Z

Where $(X\ X\ X\ X)_H \rightarrow$ 16 bit segment address

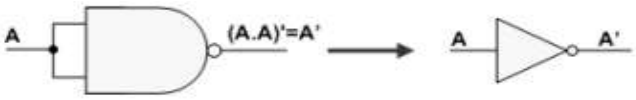
$(Y\ Y\ Y\ Y)_H \rightarrow$ 16 bit offset address

and $(Z\ Z\ Z\ Z\ Z)_H \rightarrow$ 20 bit offset address

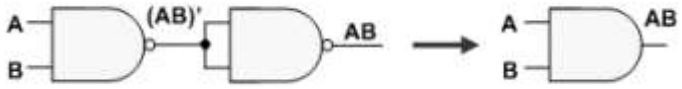
- A segment is a 64 K block, hence, there could be 16 non overlapping segments in 1 MB of memory.
- The size of any segment cannot be greater than 64KB.
- The size of any segment cannot be lesser than 16 Byte.

	(b)	**State the names of universal logic gates and design basic gates using universal gates.**	**6M**
	Ans:	**Universal logic gates :** NAND gate NOR gate NAND GATE AS AN UNIVERSAL GATES:	**1M** **NAND:2.5 M** **NOR:2.5M**

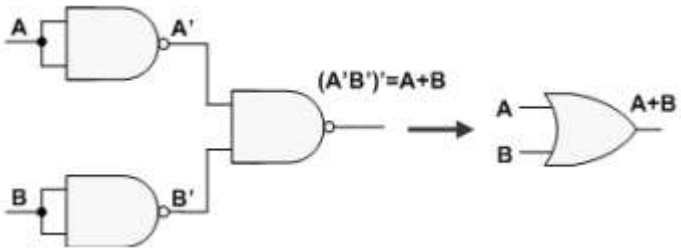
1. Implementing an Inverter Using only NAND Gate The figure shows two ways in which a NAND gate can be used as an inverter (NOT gate).All NAND input pins connect to the input signal A gives an output A'.



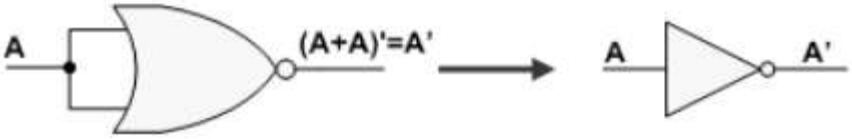
2. Implementing AND Using only NAND Gates An AND gate can be replaced by NAND gates as shown in the figure (The AND is replaced by a NAND gate with its output complemented by a NAND gate inverter).



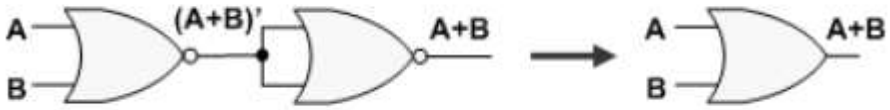
3. Implementing OR Using only NAND Gates An OR gate can be replaced by NAND gates as shown in the figure (The OR gate is replaced by a NAND gate with all its inputs complemented by NAND gate.
inverters)



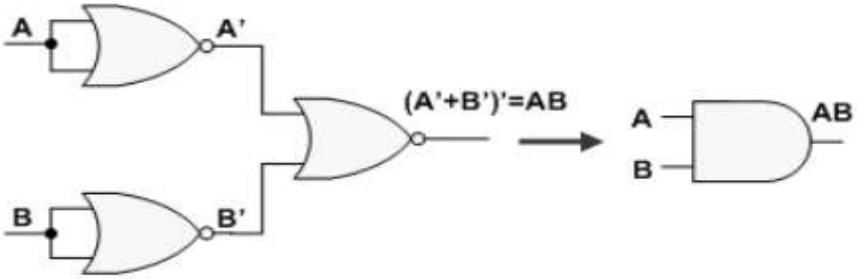
NOR GATE AS AN UNIVERSAL GATES:
1. All NOR input pins connect to the input signal A gives an output A'.



2. Implementing OR Using only NOR Gates An OR gate can be replaced by NOR gates as shown in the figure (The OR is replaced by a NOR gate with its output complemented by a NOR gate inverter).



3. An AND gate can be replaced by NOR gates as shown in the figure (The AND gate is replaced by a NOR gate with all its inputs complemented by NOR gate inverters)



	(c)	**Describe the use of shift and rotate instructions as well as string instructions with the**	**6M**

		help of one relevant examples of each.	
	Ans:	SHIFT AND ROTATE INSTRUCTIONS In the 8086 microprocessor, we have 16-bit registers to handle our data. Sometimes, the need to perform some necessary shift and rotate operations on our data may occur according to the given condition and requirement. So, for that purpose, we have various Shift and Rotate instructions present in the 8086 microprocessor. 1) SHR : Shift Right The SHR instruction is an abbreviation for 'Shift Right'. This instruction simply shifts the mentioned bits in the register to the right side one by one by inserting the same number (bits that are being shifted) of zeroes from the left end. The rightmost bit that is being shifted is stored in the Carry Flag (CF). Syntax: SHR Register, Bits to be shifted Example: SHRAX, 2 Working:  2) SAR : Shift Arithmetic Right The SAR instruction stands for 'Shift Arithmetic Right'. This instruction shifts the mentioned bits in the register to the right side one by one, but instead of inserting the zeroes from the left end, the MSB is restored. The rightmost bit that is being shifted is stored in the Carry Flag (CF). Syntax: SAR Register, Bits to be shifted Example: SAR BX, 5 Working:  2) SHL : Shift Left The SHL instruction is an abbreviation for 'Shift Left'. This instruction simply shifts the mentioned bits in the register to the left side one by one by inserting the same number (bits that are being shifted) of zeroes from the right end. The leftmost bit that is being shifted is stored in the Carry Flag (CF). Syntax: SHL Register, Bits to be shifted Example: SHLAX, 2 Working:  3) SAL : Shift Arithmetic Left The SAL instruction is an abbreviation for 'Shift Arithmetic Left'. This instruction is the same as SHL.	**INSTRUCTION:1M ,EXAMPLE:1M EACH**

Syntax: SAL Register, Bits to be shifted
Example: SAL CL, 2

Working:

5) ROL : Rotate Left

The ROL instruction is an abbreviation for 'Rotate Left'. This instruction rotates the mentioned bits in the register to the left side one by one such that leftmost bit that is being rotated is again stored as the rightmost bit in the register, and it is also stored in the Carry Flag (CF).

Syntax: ROL Register, Bits to be shifted
Example: ROL AH, 4

Working:

6) ROR : Rotate Right

The ROR instruction stands for 'Rotate Right'. This instruction rotates the mentioned bits in the register to the right side one by one such that rightmost bit that is being rotated is again stored as the MSB in the register, and it is also stored in the Carry Flag (CF).

Syntax: ROR Register, Bits to be shifted
Example: ROR AH, 4

Working:

7) RCL : Rotate Carry Left

This instruction rotates the mentioned bits in the register to the left side one by one such that leftmost bit that is being rotated it is stored in the Carry Flag (CF), and the bit in the CF moved as the LSB in the register.

Syntax: RCL Register, Bits to be shifted
Example: RCL CH, 1

Working:

8) RCR : Rotate Carry Right

This instruction rotates the mentioned bits in the register to the right side such that rightmost bit that is being rotated it is stored in the Carry Flag (CF), and the bit in the CF moved as the MSB in the register.

Syntax: RCR Register, Bits to be shifted
Example: RCRBH, 6

Working:

STRING INSTRUCTIONS

String is a group of bytes/words and their memory is always allocated in a sequential order.

1. MOVS/MOVSB/MOVSW Instruction :

This instruction copies a byte or word from a location in the data segment to a location in the extra segment. The offset of the source byte or word in the data segment must be in the SI register. The offset of the destination in the extra segment must be contained in the DI register. For multiple byte or multiple word moves the number of elements to be moved is put in the CX register so that it can function as a counter. After the byte or word is moved SI and DI are automatically adjusted to point to the next source and the next destination. If the direction flag is 0, then SI and DI will be incremented by 1 after a byte move and they will incremented by 2 after a word move. If the DF is a 1, then SI and DI will be decremented by 1 after a byte move and they will be decremented by 2 after a word move. MOVS affects no flags.

Examples :

```
CLD                     ;  Clear Direction Flag to autoincrement SI
                        ;  and DI
MOV  AX,  0000H
MOV  DS,  AX            ;  Initialize data segment register to 0
MOV  ES,  AX            ;  Initialize extra segment register to 0
MOV  SI,  2000H         ;  Load offset of start of source string
                        ;  into SI
MOV  DI,  2400H         ;  Load offset of start of destination into DI
MOV  CX,  04H           ;  Load length of string in CX as counter
REP  MOVSB              ;  Decrement CX and MOVSB until CX will be 0.
```

2. CMPS/CMPSB/CMPSW Instruction :

A 8086 String Instructions is a series of the same type of data items in sequential memory locations. The CMPS instruction can be used to compare a byte in one string with a byte in another string or to compare a word in one string with a word in another string. SI is used to hold the offset of a byte or word in the source string and DI is used to hold the offset of a byte or a word in the other string. The comparison is done by subtracting the byte or word pointed to by DI from the byte or word pointed to by SI. The AF, CF, OF, PF, SF, and ZF flags are affected by the comparison, but neither operand is affected.

Examples :

```
MOV  SI,  OFFSET  F_STRING    ;  Point SI at source string, Point DI
MOV  DI,  OFFSET  S_STRING    ;  at destination string
CLD
                              ;  DF cleared so SI and DI will
CMPS  F_STRING,  S_STRING     ;  autoincrement after compare
                              ;  The assembler uses names to determin
                              ;  whether strings were declared as typ
MOV  CX,  100                 ;  byte or as type word.
                              ;  Put number of string elements in CX,
                              ;  Point SI at source of string and DI
MOV  SI,  OFFSET  F_STRING    ;  at destination of string
MOV  DI,  OFFSET  S_STRING
STD
                              ;  DF set so SI and DI will
                              ;  autodecrement after compare
REPE  CMPSB                   ;  Repeat the comparison of string byte
                              ;  until end of string or until compare
                              ;  bytes are not equal.
```

After the comparison SI and DI will be automatically incremented or decremented according to direction flag to point to the next element in the two strings (if DF = 0, SI and DI ↑) CX functions as a counter which is decremented after each comparison. This will go on until CX = 0

3. SCAS/SCASB/SCASW Instruction :

SCAS compares a string byte with a byte in AL or a string word with word in AX. The instruction affects the flags, but it does not change either the operand in AL (AX) or the operand in the 8086 String Instructions. The string to be 'scanned must be in the extra segment and DI must contain the offset of the byte or the word to be compared.

After the comparison DI will be automatically incremented or decremented according to direction flag, to point to the next element in the two strings (if DF = 0, SI and DI ↑) CX functions as a counter which is decremented after each comparison. This will go on until CX = 0. SCAS affects the AF, CF, OF, PF, SF and ZF flags.

Examples :

```
                                ; Scan a text string of 80
                                ; characters for a carriage
                                ; return
MOV AL, 0DH                     ; Byte to be scanned for into AL
MOV DI, OFFSET TEXT_STRING      ; Offset of string to DI
MOV CX, 80                      ; CX used as element counter
CLD                             ; Clear DF, so DI
                                ; autoincrements
REPNE SCAS TEXT_STRING          ; Compare byte in string with
                                ; byte in AL.
```

SCASB says compare 8086 String Instructions as bytes and SCASW says compare strings as words.

4. LODS/LODSB/LODSW Instruction :

This instruction copies a byte from a string location pointed to by SI to AL, or a word from a string location pointed to by SI to AX. LODS does not affect any flags. LODSB copies byte and LODSW copies a word.

Examples :

```
CLD                             ; Clear direction flag so SI
                                ; is autoincremented
MOV SI, OFFSET S_STRING         ; Point SI at string
LODS S_STRING.
```

5. STOS/STOSB/STOSW Instruction :

The STOS instruction copies a byte from AL or a word from AX to a memory location in the extra segment. DI is used to hold the offset of the memory location in the extra segment. After the copy, DI is automatically incremented or decremented to point to the next string element in memory. If the direction flag, DF, is cleared, then DI will automatically be incremented by one for a byte string or incremented by two for a word 8086 String Instructions. If the direction flag is set, DI will be automatically decremented by ono for a byte string or decremented by two for a word string. STOS does not affect any flags. STOSB copies byte and STOSW copies a word.

Examples :

```
MOV DI, OFFSET D_STRING     ; Point DI at destination string
STOS D_STRING               ; Assembler uses string name to
                            ; determine whether string is of
                            ; type byte or type word. If byte
                            ; string, then string byte replaced
                            ; with contents of AL. If word
                            ; string, then string word replaced
                            ; with contents of AX.
MOV DI, OFFSET D_STRING     ; Point DI at destination string
STOSB                       ; "B" added to STOS mnemonic
                            ; directly tells assembler to
                            ; replace byte in string with
                            ; from AL. STOSW would tell assembler
                            ; directly to replace a word in
                            ; the string with a word from AX.
```