USN | | | | | | | | | | |

Internal Assessment Test 1 – November 2024

Sub:	**Digital Design and Computer Organization**					Sub Code:	BCS302	Branch:	CSE
Date:	06/11/2024	Duration:	90 mins	Max Marks:	50	Sem / Sec:	3rd semester/A,B,C		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1 a)	What is positive logic and negative logic?	[2]	CO1	L3
b)	A digital system is to be designed in which the month of the year is given as I/P in four-bit form. The month of January is represented as '0000', February as "0001" and so on. The output of the system should correspond to the input of the month containing 31 days, or otherwise, it is '0'.Consider the excess number in the I/P beyond 1011' as don't care condition: (i) Write truth table, SOP and POS form (ii) Simplify for SOP using K-map (iii) Realize using basic gates	[8]		
2 a)	Define 1.Implicant 2.Prime Implicant 3.Essential Prime Implicant 4. Redundant prime implicant. Identify Prime implicant and essential prime implicant f(A,B,C,D)=∑m(1,3,4,5,10,1112,13,14,15)	[3+2]	CO1	L2
b)	What is a tristate buffer? Design 4-to-1-line mux using tristate buffer.	[2+3]	CO2	

USN | | | | | | | | | | |

Internal Assessment Test 1 – November 2024

Sub:	**Digital Design and Computer Organization**					Sub Code:	BCS302	Branch:	CSE
Date:	06/11/2024	Duration:	90 mins	Max Marks:	50	Sem / Sec:	3rd semester/A,B,C		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1 a)	What is positive logic and negative logic?	[2]	CO1	L3
b)	A digital system is to be designed in which the month of the year is given as I/P in four-bit form. The month of January is represented as '0000', February as "0001" and so on. The output of the system should correspond to the input of the month containing 31 days, or otherwise, it is '0'.Consider the excess number in the I/P beyond 1011' as don't care condition: (i) Write truth table, SOP and POS form (ii) Simplify for SOP using K-map (iii) Realize using basic gates	[8]		
2 a)	Define 1.Implicant 2.Prime Implicant 3.Essential Prime Implicant 4. Redundant prime implicant. Identify Prime implicant and essential prime implicant f(A,B,C,D)=∑m(1,3,4,5,10,1112,13,14,15)	[3+2]	CO1	L2
b)	What is a tristate buffer? Design 4-to-1-line mux using tristate buffer.	[2+3]	CO2	

3a) b)	What is the drawback of n-bit ripple carry adder? Design and explain 4 bit carry look ahead adder. Write a Verilog HDL code to simulate the working of Half adder and Full adder in data flow and structural model	**[5]** **[2.5+2.5]**	**CO2**	**L3**
4a) b)	Implement Y (A, B, C, D) = ∑m (0, 1, 6, 7, 8, 9, 10, 11, 12, 14) using 8-to-1 multiplexer. Design 9:1 MUX using 2:1	**[5+5]**	**CO2**	**L2**
5a) b)	With neat diagram, explain basic operational concepts of a computer With a neat diagram, explain the different processor registers.	**[6+4]**	**CO3**	**L2**
6.a) b)	What is the difference between latch and flip flop? Design NOR S-R latch. Derive the characteristic equation of J-k Flip flop .Convert J-K flip flop to D flip flop	**[1+2]** **[4+3]**	**CO2**	**,L3**

CI CCI HOD

3a) b)	What is the drawback of n-bit ripple carry adder? Design and explain 4 bit carry look ahead adder. Write a Verilog HDL code to simulate the working of Half adder and Full adder in data flow and structural model	**[5]** **[2.5+2.5]**	**CO2**	**L3**
4a) b)	Implement Y (A, B, C, D) = ∑m (0, 1, 6, 7, 8, 9, 10, 11, 12, 14) using 8-to-1 multiplexer. Design 9:1 MUX using 2:1	**[5+5]**	**CO2**	**L2**
5a) b)	With neat diagram, explain basic operational concepts of a computer With a neat diagram, explain the different processor registers.	**[6+4]**	**CO3**	**L2**
6.a) b)	What is the difference between latch and flip flop? Design NOR S-R latch. Derive the characteristic equation of J-k Flip flop .Convert J-K flip flop to D flip flop	**[1+2]** **[4+3]**	**CO2**	**,L3**

CI CCI HOD

Internal Assessment Test 1

Solution

Sub:	Digital Design and Computer Organization					Sub Code:	BCS302	Branch:	CSE
Date:	06/11/2024	Duration:	90 mins	Max Marks:	50	Sem / Sec:	3 A,B,C		

1 a) **What is positive logic and negative logic?**

Positive logic and negative logic are conventions that define how logical values relate to the voltages that represent them:

Positive logic
A high voltage level represents a logic 1, and a low voltage level represents a logic 0.
Negative logic
A low voltage level represents a logic 1, and a high voltage level represents a logic 0.

b) **A digital system is to be designed in which the month of the year is given as I/P in four-bit form. The month of January is represented as '0000', February as "0001" and so on. The output of the system should correspond to the input of the month containing 31 days, or otherwise, it is '0'.Consider the excess number in the I/P beyond 1011' as don't care condition:**
(i) Write truth table, SOP and POS form (ii) Simplify for SOP using K-map (iii) Realize using basic gates

Truth Table

Name of Month	Month in binary	Output (F)
January	0000	1
February	0001	0
March	0010	1
April	0011	0
May	0100	1
June	0101	0
July	0110	1

August	0111	1
September	1000	0
October	1001	1
November	1010	0
December	1011	1
	1100	x
	1101	x
	1110	x
	1111	x

SOP Form:

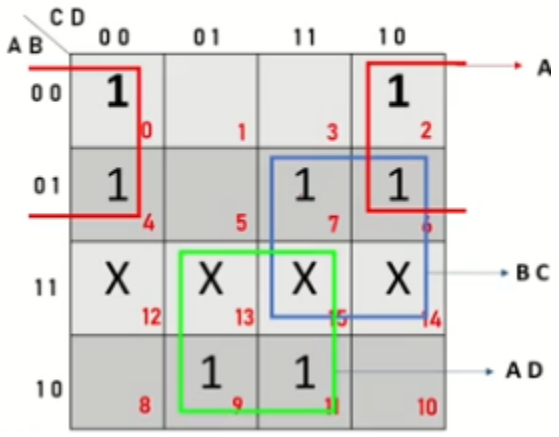
Σm= m0 + m2 + m4 + m6 + m7 + m9 + m11 + d12 + d13 + d14 + d15

Y = Σm(0,2,4,6,7,9,11) + d(12,13,14,15)

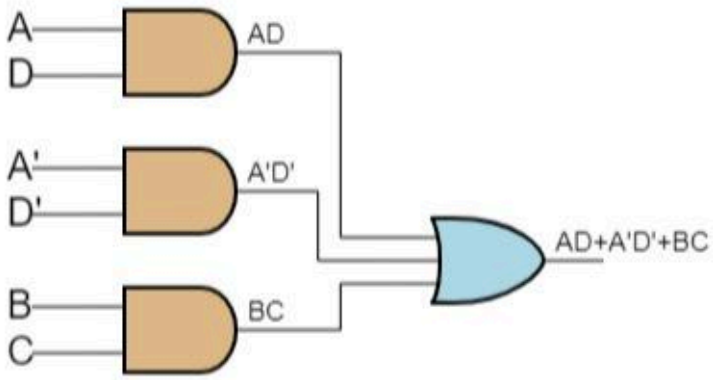
POS Form :

πM = M1+M3+M5+M8+M10+D12 + D13 + D14 + D15

Y= πM(1,3,5,8,10) + D(12,13,14,15)

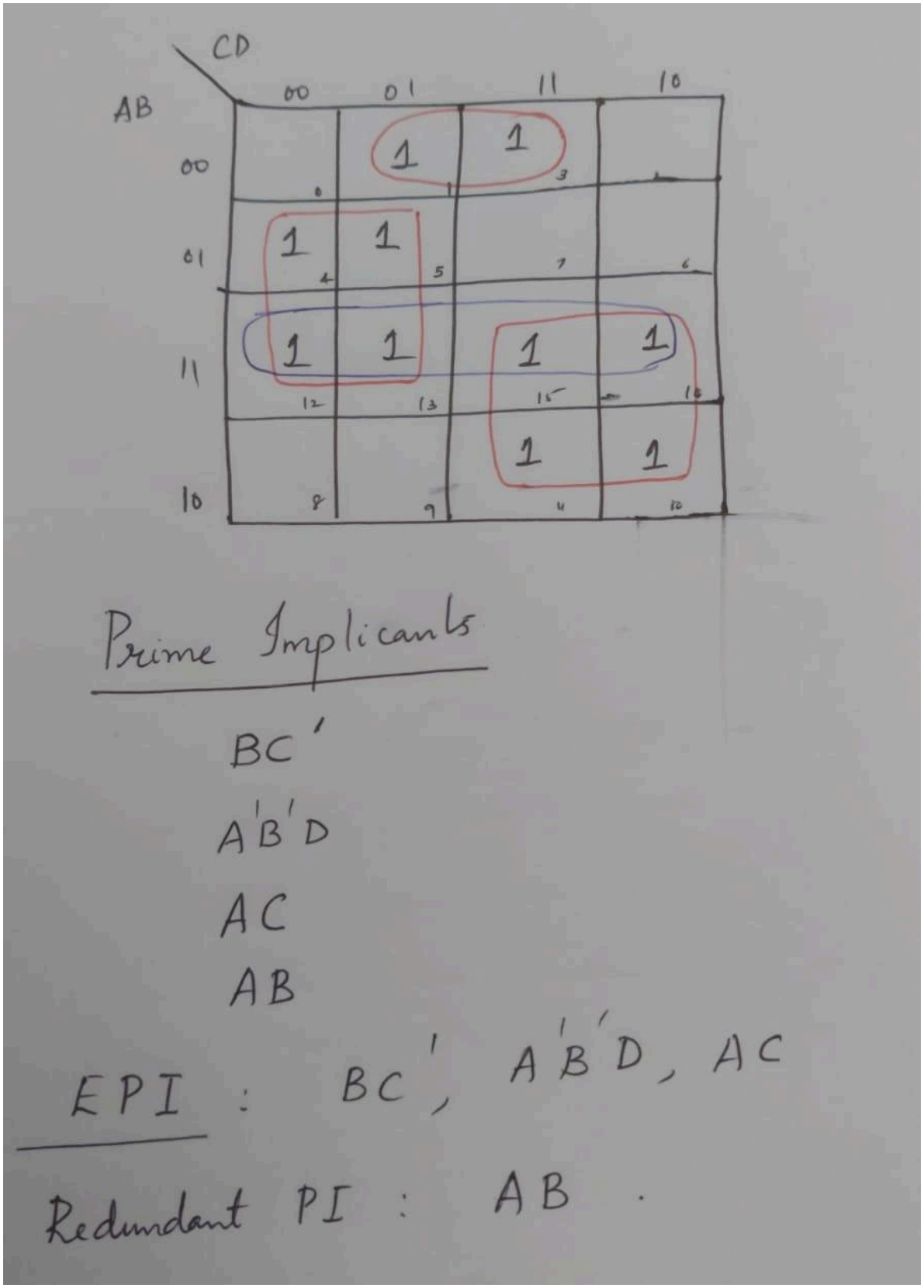


Y= AD+A'D'+BC


A
D
AD
A'
D'
A'D'
AD+A'D'+BC
B
C
BC

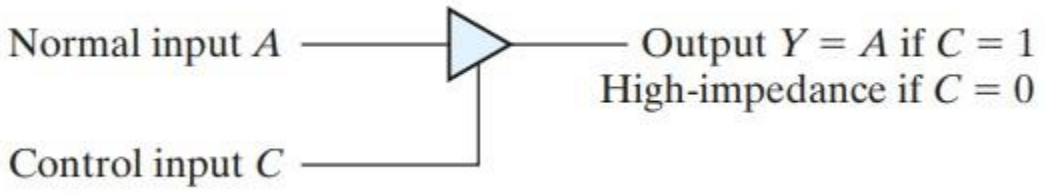
2 a)	**Define 1.Implicant 2.Prime Implicant 3.Essential Prime Implicant 4. Redundant prime implicant** **Identify Prime implicant and essential prime implicant** **$f(A,B,C,D)=\sum m(1,3,4,5,10,11,12,13,14,15)$** Implicant is a product/minterm term in Sum of Products (SOP) or sum/maxterm term in Product of Sums (POS) of a Boolean function. A group of squares or rectangles made up of a bunch of adjacent minterms which is allowed by the definition of K-Map are called **prime implicants(PI)** **These are those subcubes(groups) that cover at least one minterm that can't be covered by any other prime implicant. Essential prime implicants(EPI) are those prime implicants that always appear in the final solution.** **The prime implicants for which each of its minterm is covered by some essential prime implicant are redundant prime implicants(RPI). This prime implicant never appears in the final solution.**

b)


CD
AB
00
01
11
10
Prime Implicants
BC'
A'B'D
AC
AB
EPI : BC', A'B'D, AC
Redundant PI : AB .

What is a tristate buffer? Design 4-to-1-line mux using tristate buffer

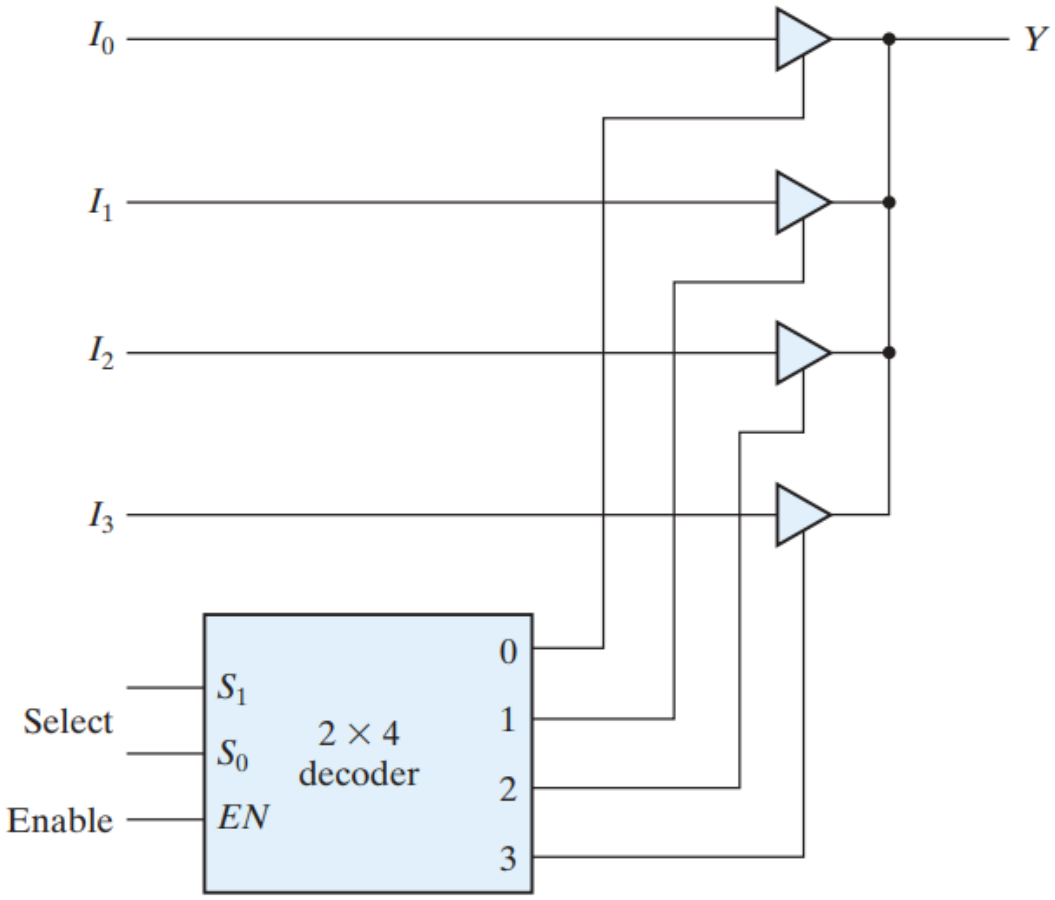
Two of the states are signals equivalent to logic 1 and logic 0 as in a conventional Gate. The third state is a high-impedance state in which
(1) the logic behaves like an open circuit, which means that the output appears to be disconnected,
(2) the circuit has no logic significance, and
(3) the circuit connected to the output of the three-state gate is not affected by the inputs to the gate. Three-state gates may perform any conventional logic, such as AND or NAND. However, the one most commonly used is the buffer gate.
The graphic symbol for a three-state buffer gate is



Mux Implementation :

Y= S1'S0' I0 + S1'S0I1 + S1 S0'I2 + S1S0 I3

S1	S0	Output Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3



(b) 4-to-1-line mux

3	a)**What is the drawback of n-bit ripple carry adder? Design and explain 4 bit carry look ahead adder** The main drawback of an n-bit ripple carry adder is that it can be slow when the value of (N) is large. This is because each full adder must wait for the carry-in from the previous full adder.This means that the nth full adder must wait until all (n-1) full adders have finished their operations ## Carry Look Ahead Adder • Overcomes the drawback of Parallel adder • Sum and carry outputs of each stage are made independent of the results of previous stages- ripple effect is eliminated • Uses concept of look-ahead carry • Requires additional circuitry but speed becomes independent of number of bits (or stages) • $C_i = A_i B_i + C_{i-1} (A_i \oplus B_i)$ • Let P_i be carry propogate, $P_i = A_i \oplus B_i$ • Let G_i be carry generate , $G_i = A_i B_i$

- $C_i = G_i + P_i C_{i-1}$
- $C_{i+1} = G_{i+1} + P_{i+1} C_i$
- $C_{i+1} = G_{i+1} + P_{i+1} G_i + P_{i+1} P_i C_{i-1}$

For first full adder

- Carry propagate: $P_0 = A_0 \oplus B_0$
- Carry generate: $G_0 = A_0 B_0$
- Carry out of first full adder: $C_0 = G_0 + P_0 C_{-1}$

For second full adder

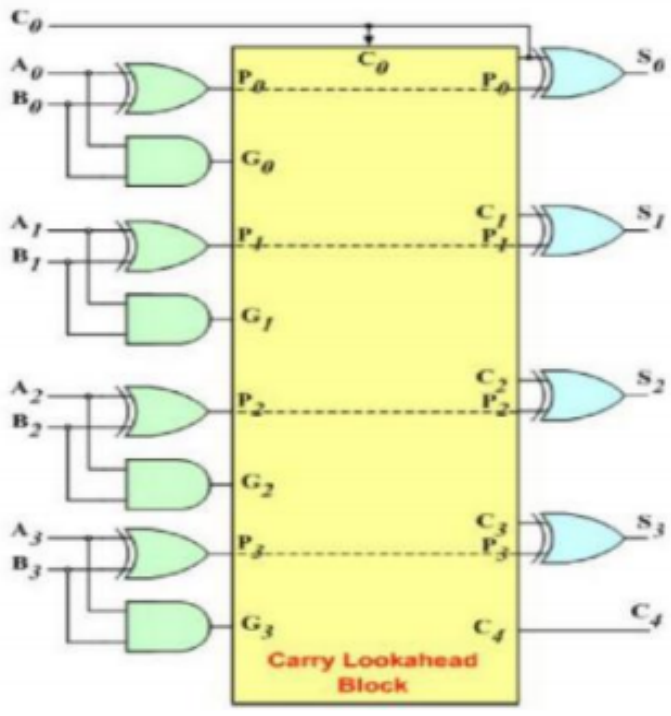
$C_1 = G_1 + P_1 C_0$

$C_1 = G_1 + P_1 G_0 + P_1 P_0 C_{-1}$

Continuing to other stages

$C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{-1}$

$C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{-1}$



3 b)	Write a Verilog HDL code to simulate the working of Half adder and Full adder in data flow and structural model

HALF ADDER

Dataflow Modelling

```
module half_adder (
    input a,b,
    output sum,carry
);

assign sum = a ^ b;
assign carry = a & b;

endmodule
```

Structural

```
module half_adder (
    input a,b,
    output sum,carry
);
xor (s, a, b);
and(c, a, b);

endmodule
```

FULL ADDER

Dataflow Modelling

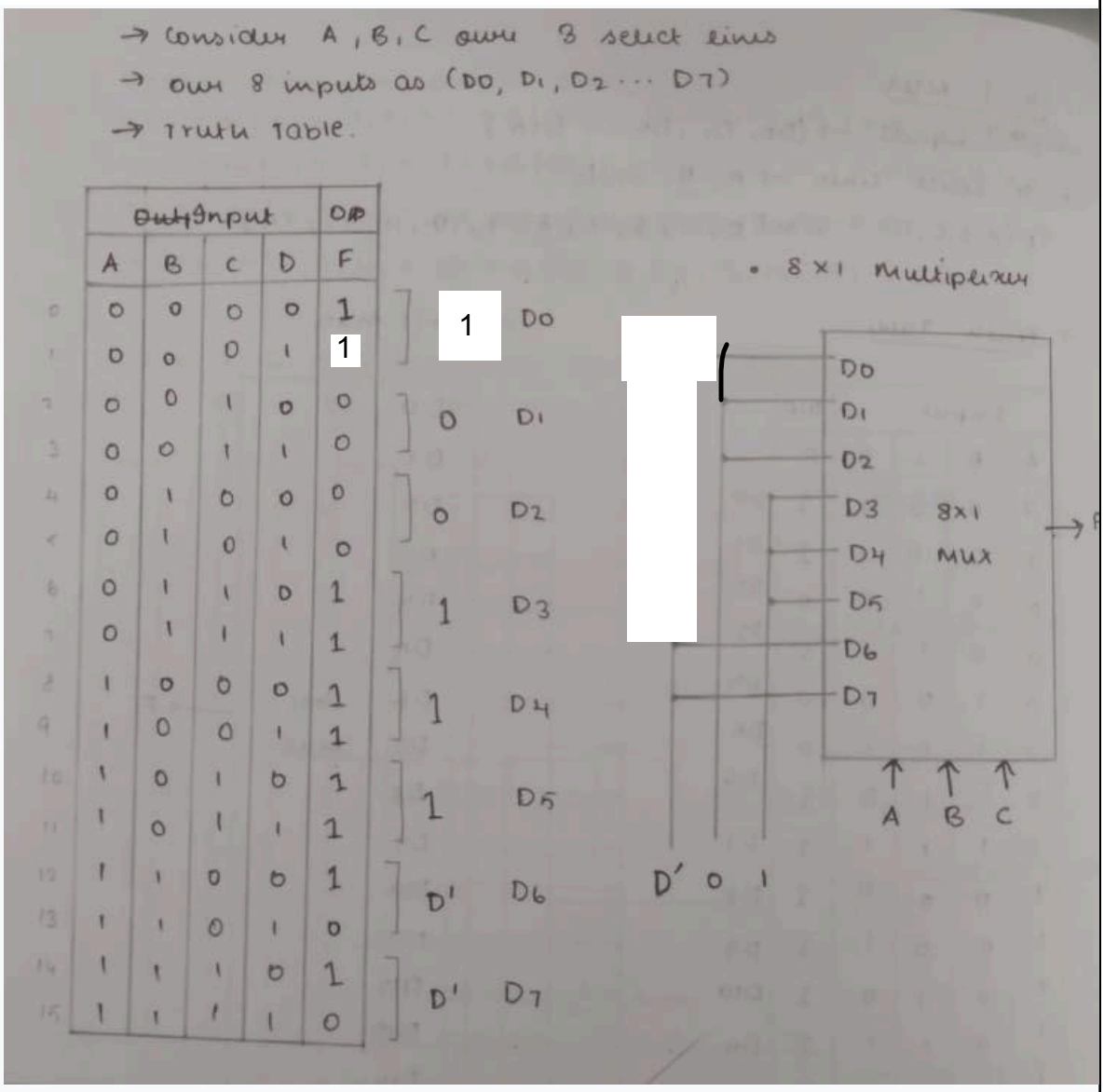
```
module full_adder_d (
    input a,b,cin,
    output sum,carry
);
assign sum = a ^ b ^ cin;
assign carry = (a & b) | (b & cin)  | (cin & a)
;

endmodule
```

Structural Modeling

```
module full_adder_s (
    input a,b,cin,
    output sum,carry
);

wire w1,w2,w3,w4;          //Internal connections

xor(w1,a,b);
xor(sum,w1,cin);           //Sum output

and(w2,a,b);
and(w3,b,cin);
and(w4,cin,a);

or(carry,w2,w3,w4);        //carry output


endmodule
```

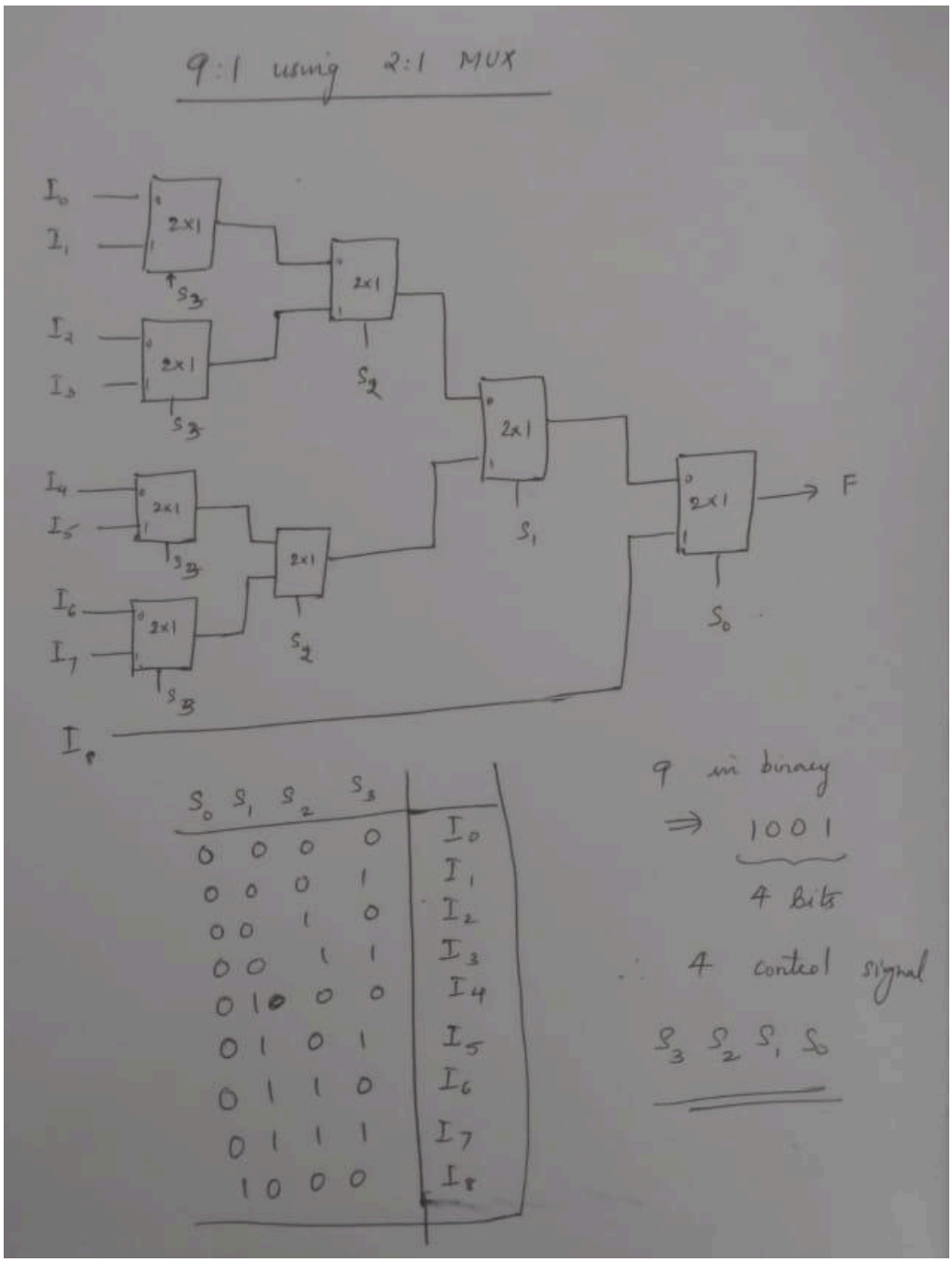
4 a) Implement Y (A, B, C, D) = ∑m (0, 1, 6, 7, 8, 9, 10, 11, 12, 14) using 8-to-1 multiplexer.

4 b)



Design 9:1 MUX using 2:1

5 a)



S_0	S_1	S_2	S_3	
0	0	0	0	I_0
0	0	0	1	I_1
0	0	1	0	I_2
0	0	1	1	I_3
0	1	0	0	I_4
0	1	0	1	I_5
0	1	1	0	I_6
0	1	1	1	I_7
1	0	0	0	I_8

With neat diagram, explain basic operational concepts of a computer

The program to be executed is stored in memory. Instructions are accessed from memory to the processor one by one and executed.

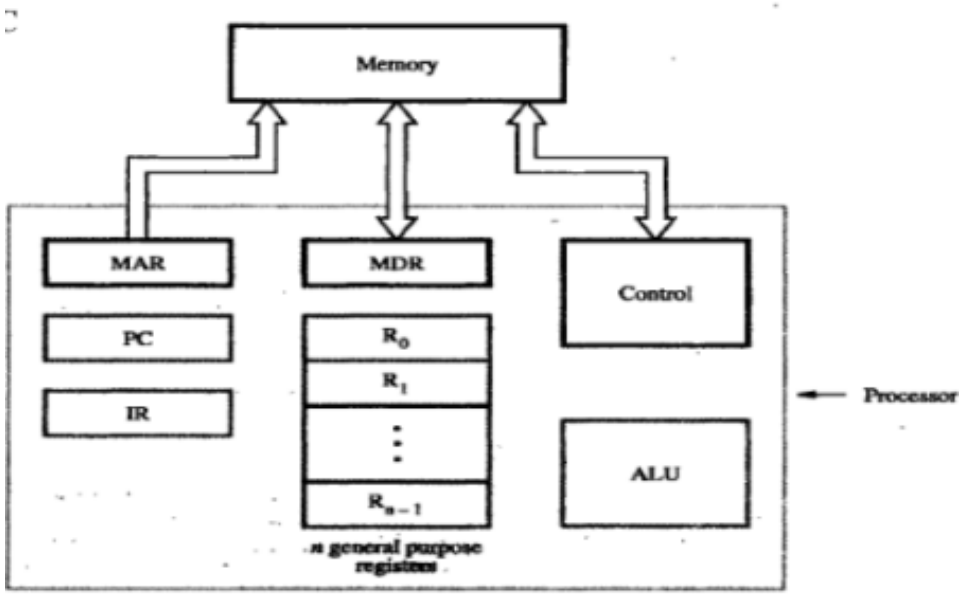
5 b)	STEPS FOR INSTRUCTION EXECUTION Consider the following instruction Ex: 1 Add LOCA, R0 This instruction is in the form of the following instruction format Opcode Source, Destination Where Add is the operation code, LOCA is the Memory operand and R0 is Register operand This instruction adds the contents of memory location LOCA with the contents of Register R0 and the result is stored in R0 Register. The symbolic representation of this instruction is R0← [LOCA] + [R0] The contents of memory location LOCA and Register R0 before and after the execution of this instruction is as follows Before instruction execution After instruction execution LOCA = 23H LOCA = 23H R0 = 22H R0 = 45H The steps for instruction execution are as follows 1. Fetch the instruction from memory into the IR (instruction register in CPU). 2. Decode the instruction 1111000000 10011010 3. Access the Memory Operand 4. Access the Register Operand 5. Perform the operation according to the Operation Code. 6. Store the result into the Destination Memory location or Destination Register. Ex:2 Add R1, R2, R3 This instruction is in the form of the following instruction format Opcode, Source-1, Source-2, Destination Where R1 is Source Operand-1, R2 is the Source Operand-2 and R3 is the Destination. This instruction adds the contents of Register R1 with the contents of R2 and the result is placed in R3 Register. The symbolic representation of this instruction is R3 ← [R1] + [R2] The contents of Registers R1,R2,R3 before and after the execution of this instruction is as follows. Before instruction execution After instruction execution

	R1 = 24H R2 = 34H R3 = 38H	R1 = 24H R2 = 34H R3 = 58H

The steps for instruction execution is as follows
1. Fetch the instruction from memory into the IR.
2. Decode the instruction
3. Access the First Register Operand R1
4. Access the Second Register Operand R2
5. Perform the operation according to the Operation Code.
6. Store the result into the Destination Register R3.

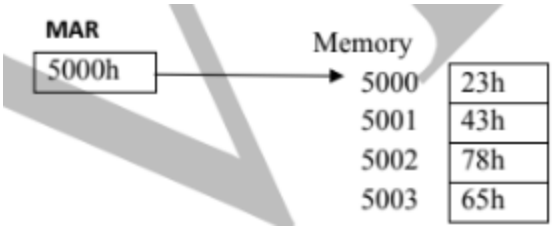
With a neat diagram, explain the different processor registers

The Processor consists of different types of registers.
1. MAR (Memory Address Register)
2. MDR (Memory Data Register)
3. PC (Program Counter)
4. General Purpose Registers
5. IR (Instruction Register)

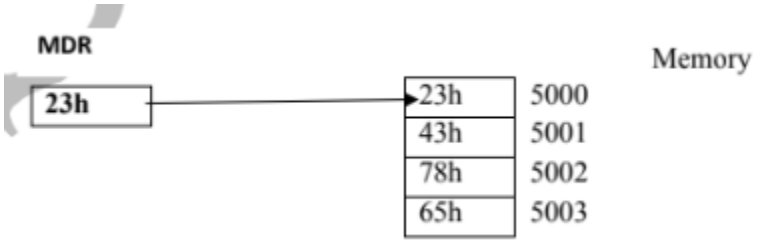


The functions of these registers are as follows
1. MAR
It establishes communication between Memory and Processor
It stores the address of the Memory Location as shown in the figure.

6 a)

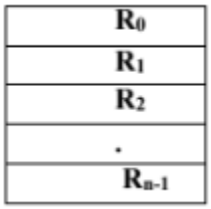


2. MDR
It also establishes communication between Memory and the Processor.
It stores the contents of the memory location (data or operand), written into or read from memory as shown in the figure.



3. PC (PROGRAM COUNTER)
It is a special purpose register used to hold the address of the next instruction to be executed.
The contents of PC are incremented by 1 or 2 or 4, during the execution of current instruction.
The contents of PC are incremented by 1 for 8 bit CPU, 2 for 16 bit CPU and for 4 for 32 bit CPU.

4. GENERAL PURPOSE REGISTER / REGISTER ARRAY
The structure of register file is as shown in the figure
It consists of set of registers.
A register is defined as group of flip flops. Each flip flop is designed to store 1 bit of Data.



It is a storage element.
It is used to store the data temporarily during the execution of the program(eg: result).
It can be used as a pointer to Memory.
The Register size depends on the processing speed of the CPU
EX: Register size = 8 bits for 8 bit CPU

5. IR (INSTRUCTION REGISTER
It holds the instruction to be executed. It notifies the control unit, which generates timing signals that control various operations in the execution of that instruction.
.

6 b)

What is the difference between latch and flip flop? Design NOR S-R latch

Flip-Flop	Latch
Flip-flop is a bistable device i.e., it has two stable states that are represented as 0 and 1.	Latch is also a bistable device whose states are also represented as 0 and 1.
It checks the inputs but changes the output only at times defined by the clock signal or any other control signal.	It checks the inputs continuously and responds to the changes in inputs immediately.
It is a edge triggered device.	It is a level triggered device.
Gates like NOR, NOT, AND, NAND are building blocks of flip flops.	These are also made up of gates.
They are classified into asynchronous or synchronous flipflops.	There is no such classification in latches
Flip-flop always have a clock signal	Latches doesn't have a clock signal
ex:D Flip-flop, JK Flip-flop	ex:SR Latch, D Latch

SR LATCH USING NOR

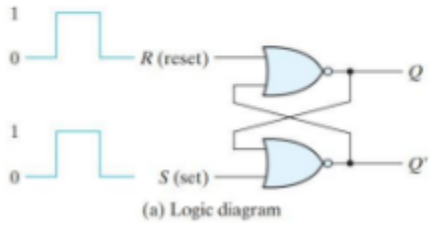
The SR latch has two inputs, S (Set) and R (Reset).It has two outputs, Q and ~Q (complement of Q).

• When S is asserted, Q is set to 1, and when R is asserted, Q is reset to 0.The SR latch is sensitive to the input conditions, and having both

S and R asserted simultaneously can lead to unpredictable behavior.

S	R	Q	Q'
0	0	NO CHANGE (Previous output)	
0	1	0	1
1	0	1	0
1	1	FORBIDDEN	

SR Latch with nor gates

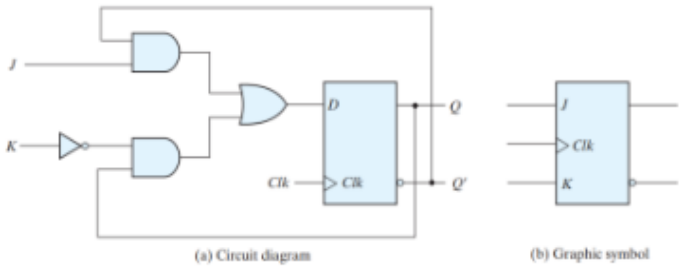


(a) Logic diagram

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)

(b) Function table

Derive the characteristic equation of J-k Flip flop .Convert J-K flip flop to D flip flop
When J = 1 and K = 0, D = Q' + Q = 1, so the next clock edge sets the output to 1.
When J = 0 and K = 1, D = 0, so the next clock edge resets the output to 0.
When both J = K = 1 and D = Q, the next clock edge complements the output.
When both J = K = 0 and D = Q, the clock edge leaves the output unchanged.



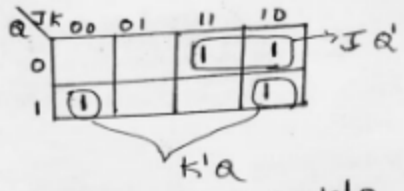
(a) Circuit diagram (b) Graphic symbol

FIGURE 5.12
JK flip-flop

Table 5.1
Flip-Flop Characteristic Tables

JK Flip-Flop			
J	K	Q(t + 1)	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

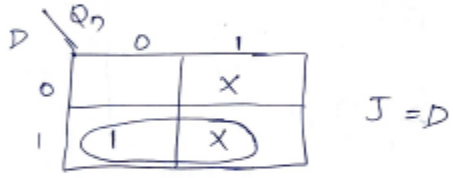
Q	J	K	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0



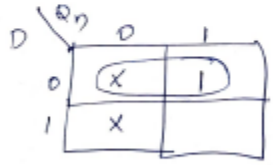
Conversion of JK to D Flip Flop

Jk to D

D	Q_n	Q_{n+1}	J	k
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	1	X	0



J = D



$k = \bar{D}$