CBCS SCHEME

USN | | | | | | | | 0 | 8 | 1 |

BCS302

Third Semester B.E/B.Tech. Degree Examination, Dec.2025/Jan.2026
Digital Design and Computer Organization

Time: 3 hrs. Max. Marks:100

Note: *1. Answer any FIVE full questions, choosing ONE full question from each module.*
2. M : Marks , L: Bloom's level , C: Course outcomes.

			M	L	C
		Module – 1			
1	a.	Obtain the minimum expression for the POS expression : $F(A, B, C, D) = \pi M(0, 1, 5, 7, 9, 13, 15) + d(3, 10)$.	5	L2	CO1
	b.	Implement the following logic function in SOP form using NOR gates. $Y = A\bar{B} + B\bar{C} + ABC$	5	L3	CO1
	c.	Identify the essential prime implicants of the following functions : $F(w, x, y, z) = (0, 1, 4, 5, 6, 7, 9, 11, 14, 15)$ $F(A, B, C, D) = (0, 2, 3, 5, 7, 8, 10, 11, 14, 15)$.	10	L3	CO1
		OR			
2	a.	Demonstrate the positive and negative logic using AND gate.	5	L2	CO1
	b.	Simplify the following Boolean functions using K-map : i) $F(P, Q, R, S) = \Sigma(0, 2, 5, 7, 8, 10, 13) + d(1, 4, 15)$ ii) $F(A,B,C,D) = (\bar{A}+B+C)(\bar{A}+\bar{C}+D)(\bar{B}+C+D)$.	10	L3	CO1
	c.	Explain Dataflow Modeling in verilogy with an example program.	5	L1	CO1
		Module – 2			
3	a.	Explain the difference between combinational and sequential circuits with their block diagrams and examples.	5	L2	CO2
	b.	Write the verilog program to implement full adder and full subtractor circuits.	7	L2	CO2
	c.	Describe and explain 4 bit adder with carry look ahead.	8	L3	CO2
		OR			
4	a.	Implement the Boolean function : $F(A, B, C, D) = \Sigma m(1, 3, 4, 11, 12, 13, 14, 15)$ using 8 : 1 MUX.	5	L3	CO2
	b.	What is encoder? Design 8 : 3 encoder circuits with logic diagram and truth table and also list its applications.	7	L3	CO2
	c.	What is Latch? Demonstrate the working of SR flip-flop and D Flip-flop and write the characteristics table and equations.	8	L3	CO2

Module – 3

5	a.	What do you mean by an addressing mode? Explain any 5 addressing modes.	10	L2	CO3
	b.	Describe the Big-endian and Little-endian address assignment.	5	L1	CO3
	c.	A program with 5000 machine instructions needs an average of 3 basic steps to execute one instruction. Find the performance of the computer having a clock speed of 500 KHz.	5	L3	CO3
		OR			
6	a.	Demonstrate the Branching operations using loop to add n numbers with block diagram.	8	L3	CO3
	b.	Show how below expression will be executed in one address and three address processor in accumulator organization. X = (A * B) + (C * D).	7	L3	CO3
	c.	What are Condition Code Flags? Mention the significance of the flag N, Z, V and C.	5	L1	CO3

Module – 4

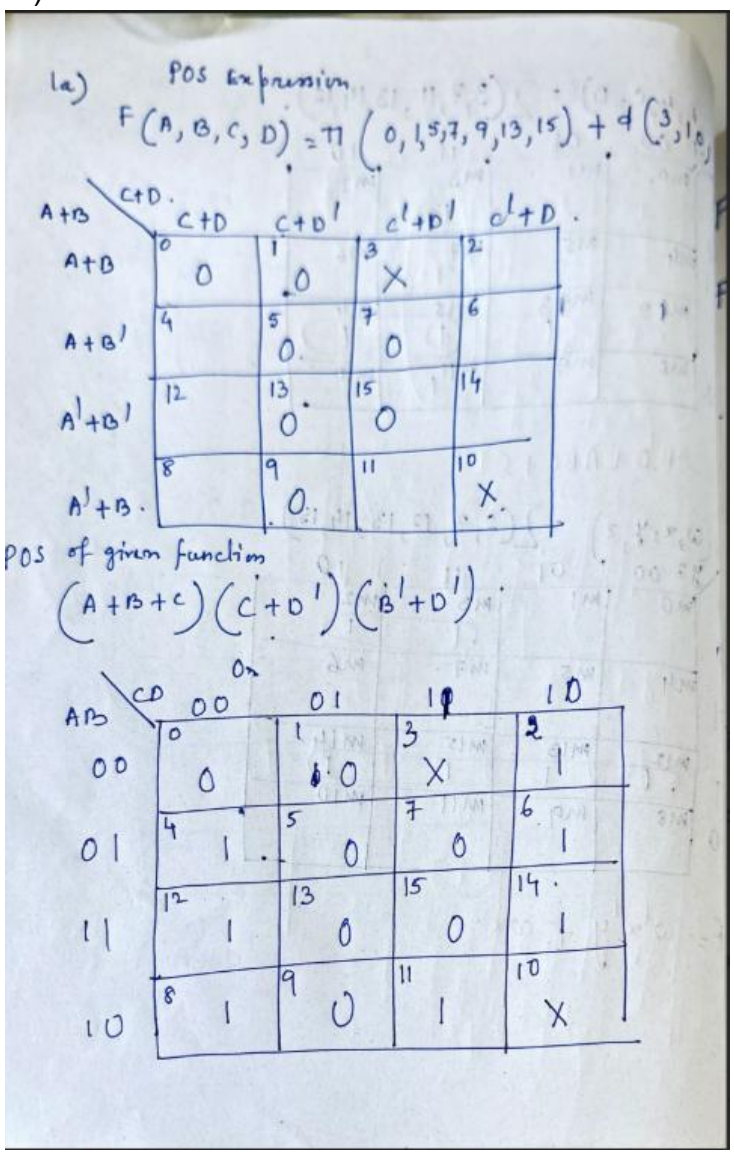
7	a.	Explain memory mapped I/O and I/O interface for an input device with a diagram.	10	L2	CO4
	b.	Explain DMA with a neat diagram.	10	L4	CO4
		OR			
8	a.	Explain how to handle interrupt from multiple devices using daisy chain and priority scheme.	10	L3	CO4
	b.	Explain centralized and distributed Bus Arbitration approaches.	10	L2	CO4

Module – 5

9	a.	With a diagram, explain the single bus organization of the data path inside a processor.	10	L2	CO5
	b.	Describe the basic idea of instruction pipeline.	10	L2	CO5
		OR			
10	a.	Explain the process of fetching word from memory in processor.	10	L4	CO5
	b.	Explain the pipeline performance of a processor and pipeline stalls.	10	L2	CO5

		Module – 1	M	L	CO
1	a.	Obtain the minimum expression for the POS expression : $F(A, B, C, D) = \pi M(0, 1, 5, 7, 9, 13, 15) + d(3, 10)$.	5	L2	CO1
	b.	Implement the following logic function in SOP form using NOR gates. $Y = A\bar{B} + B\bar{C} + ABC$	5	L3	CO1
	c.	Identify the essential prime implicants of the following functions : $F(w, x, y, z) = (0, 1, 4, 5, 6, 7, 9, 11, 14, 15)$ $F(A, B, C, D) = (0, 2, 3, 5, 7, 8, 10, 11, 14, 15)$.	10	L3	CO1

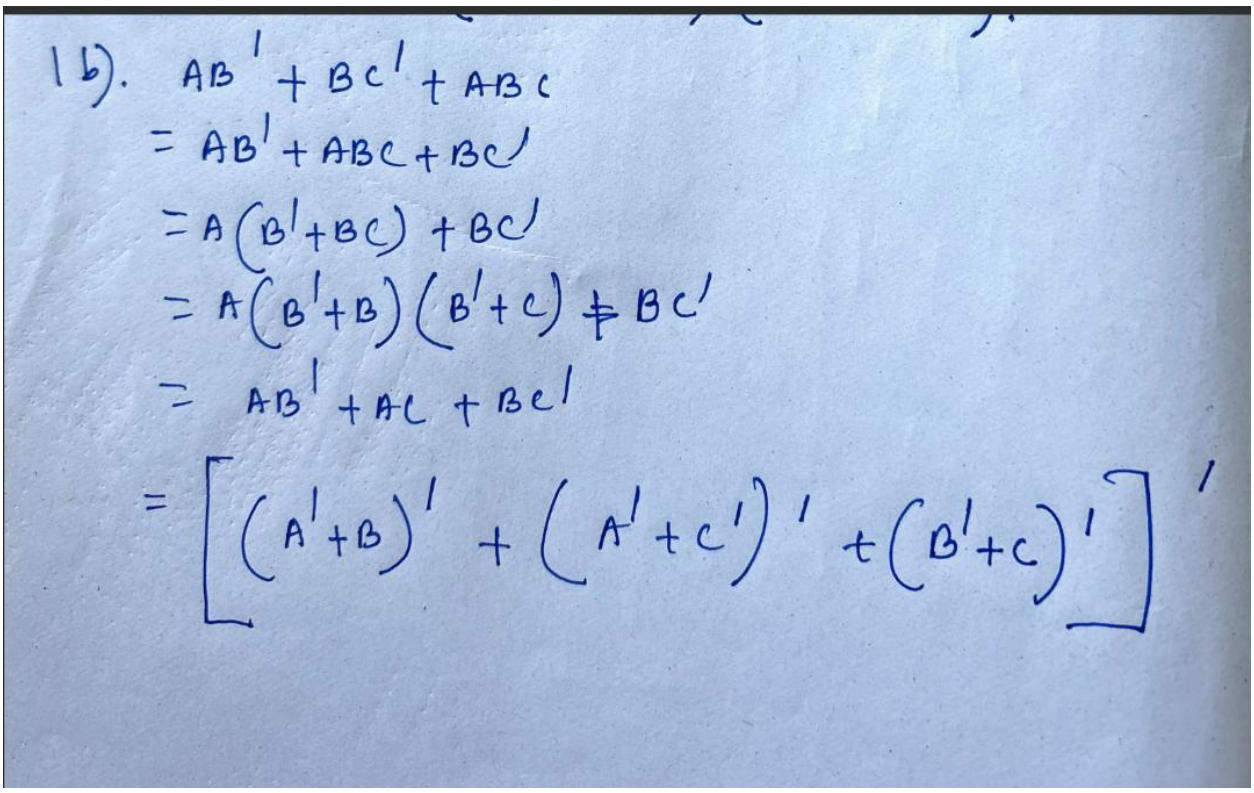
1a)



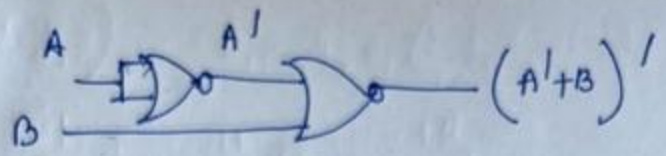
$F' = C'D + BD + A'B'C'$

$(F')' = (C + D')(B' + D')(A + B + C)$ applying duality theorem

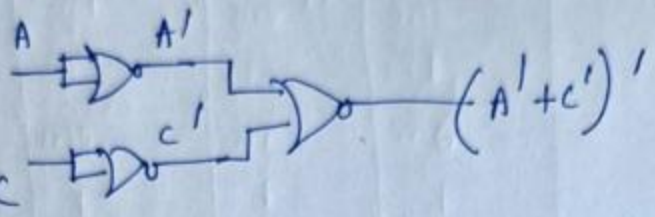
1b



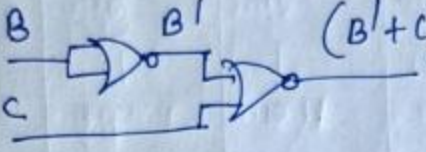
$(A'+B)'$ =



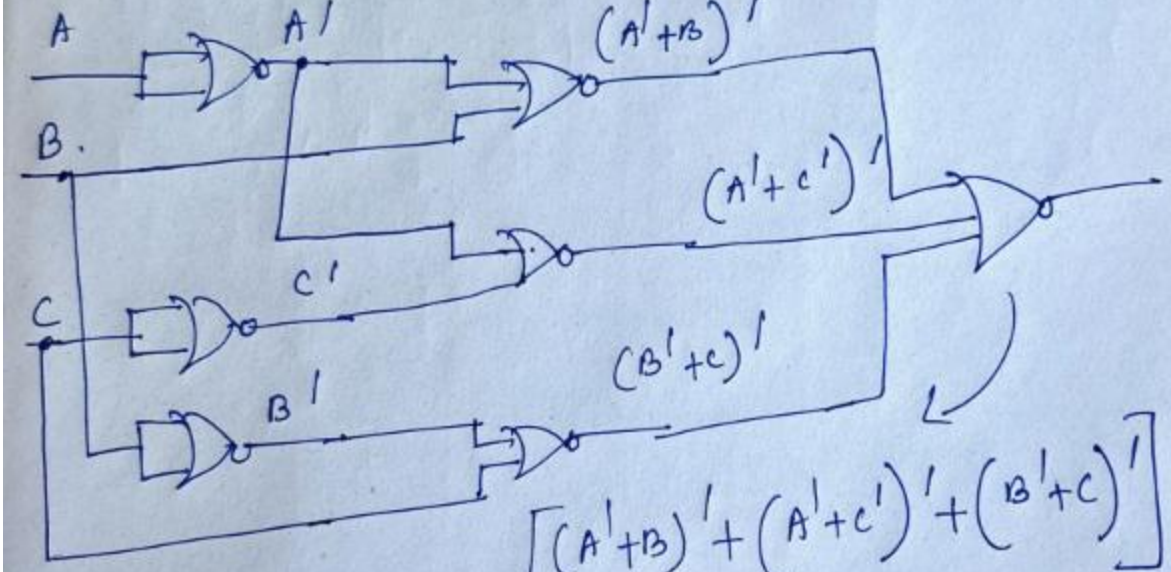
$(A'+C')'$ =



$(B'+C)'$ =



Combining.

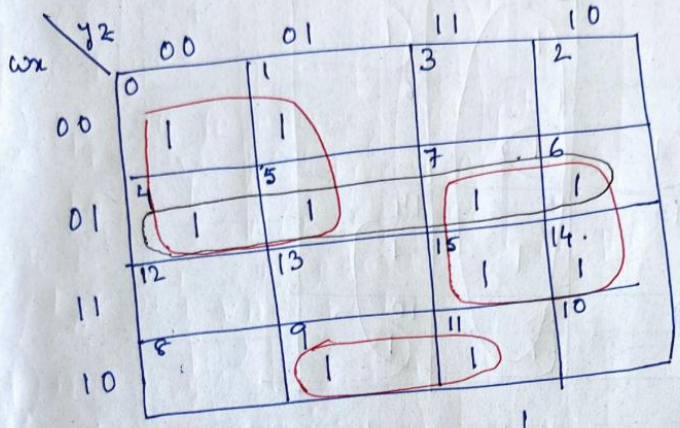


$[(A'+B)'+(A'+C')'+(B'+C)']'$

(NOR implementation).

1c) Identify the essential prime implicants.

$F(w,x,y,z) = (0,1,4,5,6,7,9,11,14,15)$.



Essential $w'y'$, xy, $wx'z$

non essential. $w'x$

$F = w'y' + xy + wx'z$.

		OR			
2	a.	Demonstrate the positive and negative logic using AND gate.	5	L2	CO1
	b.	Simplify the following Boolean functions using K-map : i) $F(P, Q, R, S) = \Sigma(0, 2, 5, 7, 8, 10, 13) + d(1, 4, 15)$ ii) $F(A,B,C,D) = (\bar{A}+B+C)(\bar{A}+\bar{C}+D)(\bar{B}+C+D)$.	10	L3	CO1
	c.	Explain Dataflow Modeling in verilog with an example program.	5	L1	CO1

2a Choosing the high-level H to represent logic 1 defines a positive logic system. Choosing the low-level L to represent logic 1 defines a negative logic system. Hardware digital gates are defined in terms of signal values such as H and L. It is up

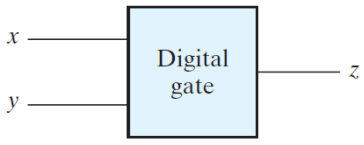
to the user to decide on a positive or negative logic polarity. Consider, for example, the electronic gate shown in Fig. (b). The truth table for this gate is listed in Fig. (a). It specifies the physical behavior of the gate when H is 3 V and L is 0 V. The truth table of Fig. (c) assumes a positive logic assignment, with H = 1 and L = 0. This truth table is the same as the one for the AND operation. The graphic symbol for a positive logic AND gate is shown in Fig. (d).

Now consider the negative logic assignment for the same physical gate with L = 1 and H = 0. The result is the truth table of Fig. (e). This table represents the OR operation, even though the entries are reversed. The graphic symbol for the negative logic

OR gate is shown in Fig. (f). The small triangles in the inputs and output

x	y	z
L	L	L
L	H	L
H	L	L
H	H	H

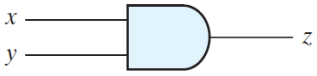
(a) Truth table with H and L



(b) Gate block diagram

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

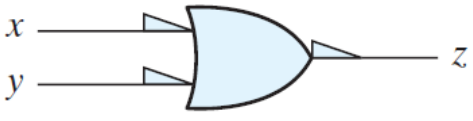
(c) Truth table for positive logic



(d) Positive logic AND gate

x	y	z
1	1	1
1	0	1
0	1	1
0	0	0

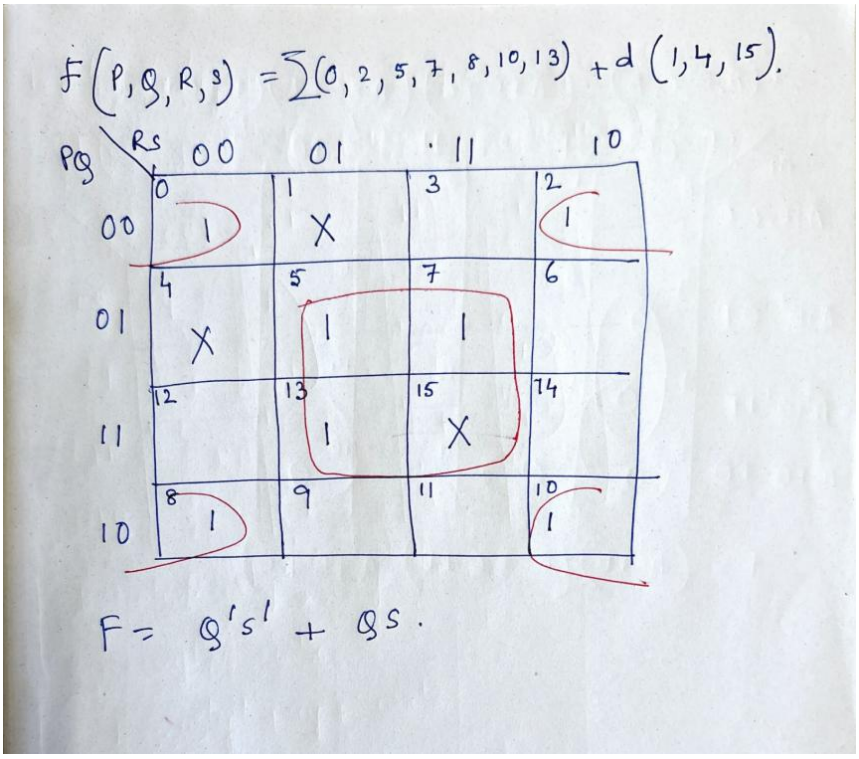
(e) Truth table for negative logic



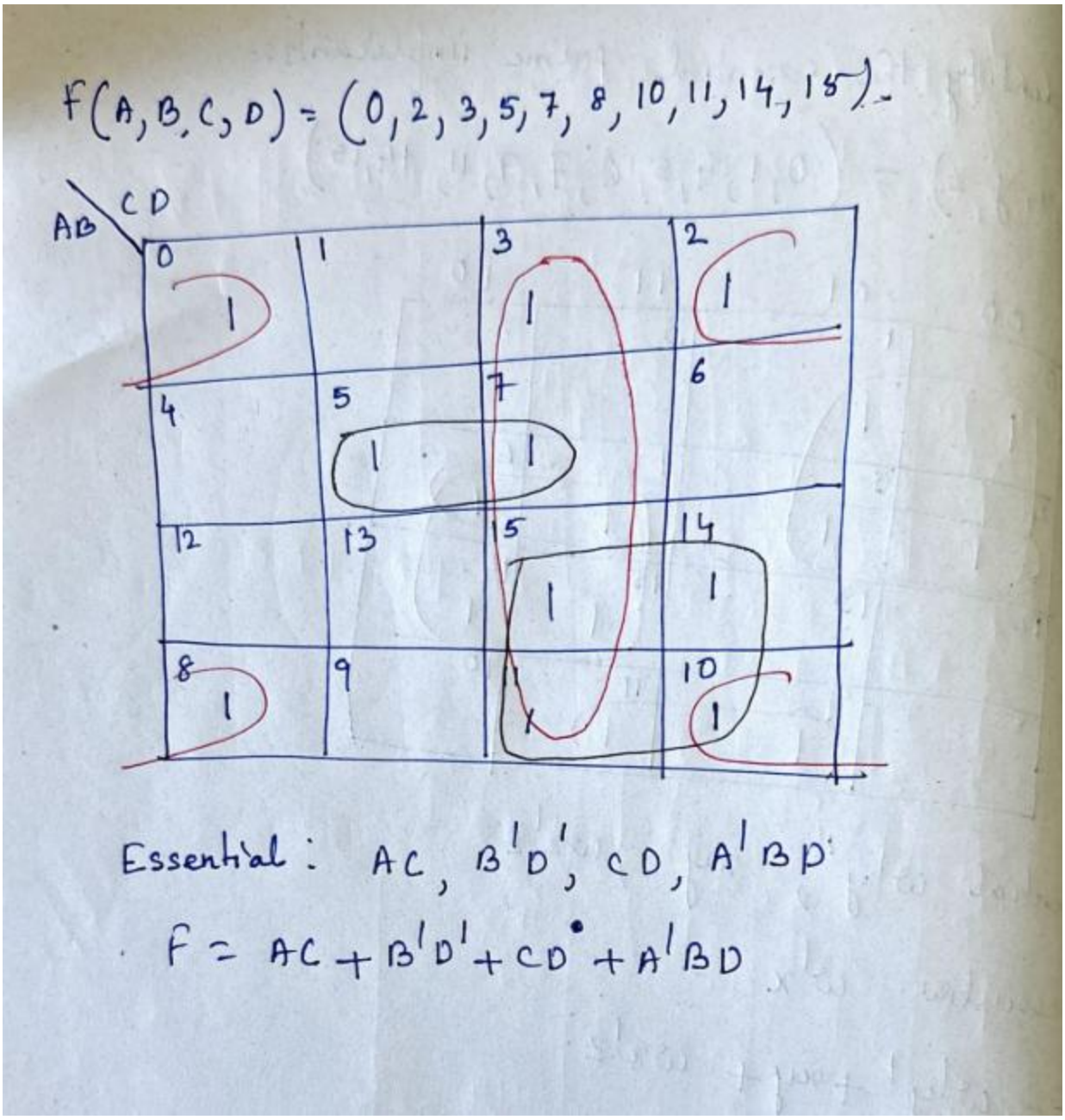
(f) Negative logic OR gate

Dataflow Modelling:

2b i)



ii)



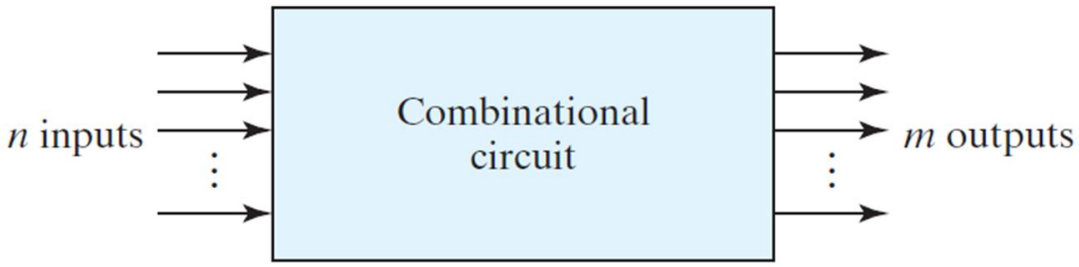
2c Dataaflow modeling of combinational logic uses a number of operators that act on binary
operands to produce a binary result.

```
// Dataflow description of four-bit adder
// Verilog 2001, 2005 module port syntax
module binary_adder (
  output [3: 0]          Sum,
  output                 C_out,
  input [3: 0]           A, B,
  input                  C_in
);
  assign {C_out, Sum} = A + B + C_in;
endmodule
```

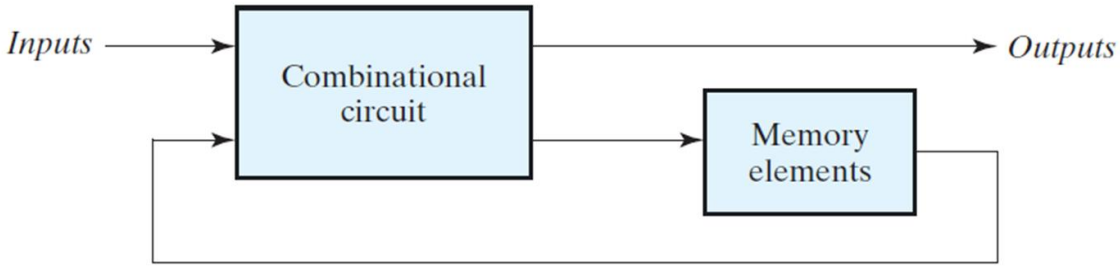
Module 2

3a

	Combinational Logic Circuits	Sequential Logic Circuits
Definition	At any instant of time, the output is only dependent on the current state of the inputs.	At any instant of time, the output is determined by inputs and previous outputs.
Time dependency	Time is not an important parameter.	Time is an important parameter. For timing and synchronizing of different circuit elements, a clock signal is necessary.
Memory	The output is solely dependent on inputs only. No need for memory.	Memory is required to store the previous state of the system.
Design	Easy to design and implement with the help of basic logic gates.	The design of these systems requires basic logic gates and flip flops.
Feedback	There is no feedback.	There is at least one memory element in the feedback path.
Hardware & cost	They are easier to implement but costly, due to hardware. Their implementation requires more hardware.	They are difficult to implement but less costly than sequential circuits.
Speed	They are faster since all inputs are applied at the same time.	They are slower, because of the secondary inputs. So, there is a delay in between inputs. And the output is gated by a clock signal.



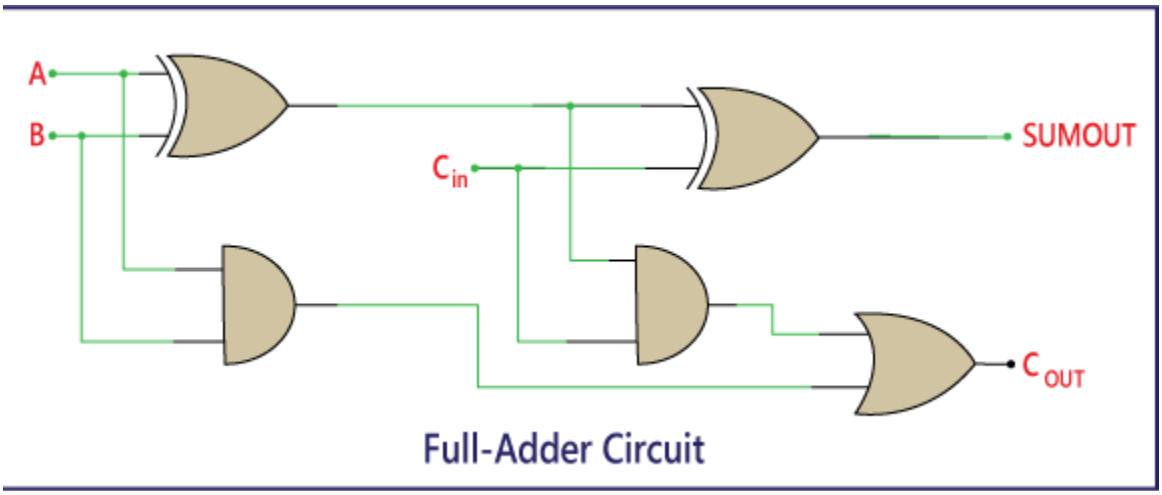
COMBINATIONAL CIRCUIT



SEQUENTIAL CIRCUIT

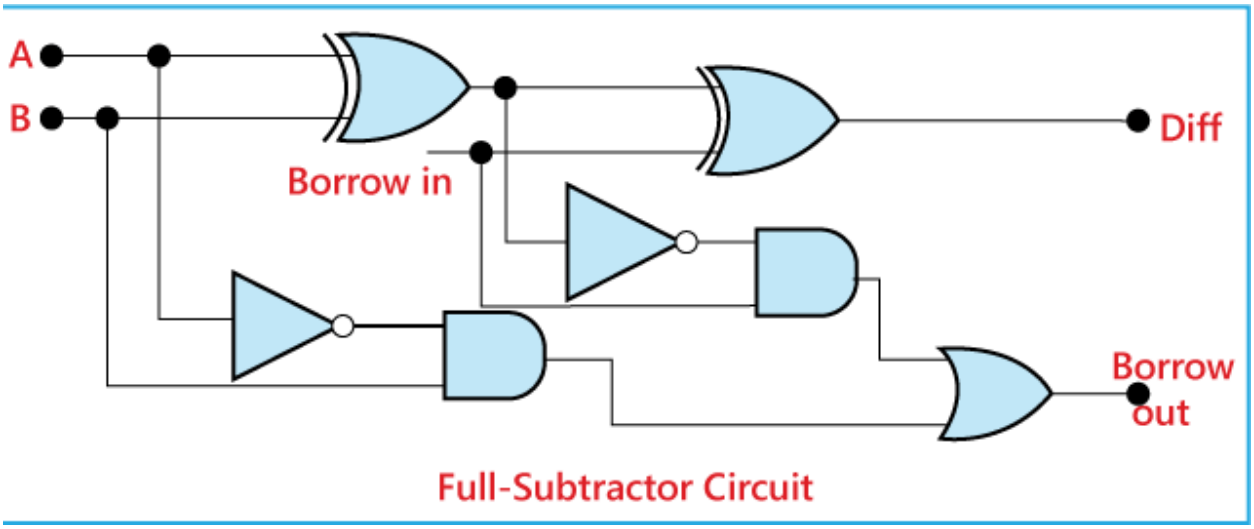
3b

```
module full_adder(input a, b, cin, output S, Cout);
  assign S = a ^ b ^ cin;
  assign Cout = (a & b) | (b & cin) | (a & cin);
endmodule
```



3c

```
module full_subtractor(input a, b, Bin, output D, Bout);
  assign D = a ^ b ^ Bin;
  assign Bout = (~a & b) | (~(a ^ b) & Bin);
endmodule
```



3c

•The carry propagation time is an important attribute of the adder because it limits the speed with which two numbers are added.

•Although the adder—or, for that matter,any combinational circuit—will always have some value at its output terminals,

•the outputs will not be correct unless the signals are given enough time to propagate through the gates connected from the inputs to the outputs.

•Since all other arithmetic operations are implemented by successive additions, the time consumed during the addition process is critical.

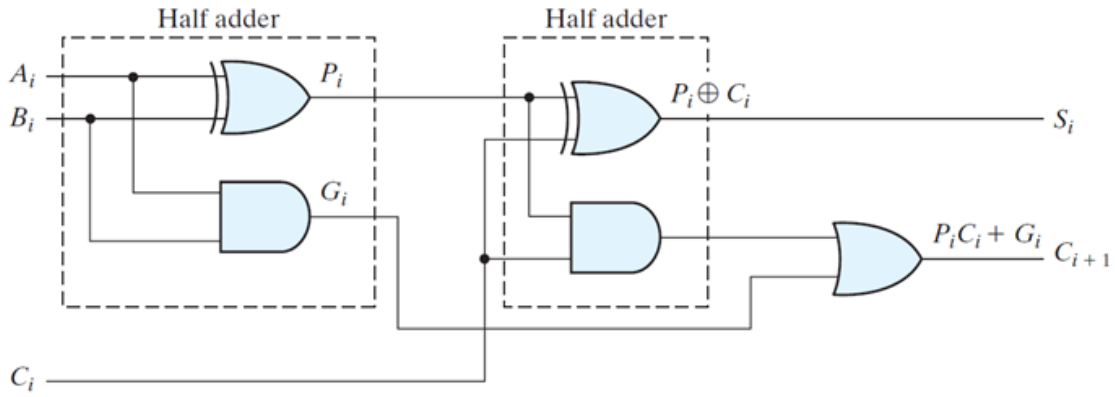
•An obvious solution for reducing the carry propagation delay time is to employ faster gates with reduced delays.

•However, physical circuits have a limit to their capability.

•Another solution is to increase the complexity of the equipment in such a way that the carry delay time is reduced.

•There are several techniques for reducing the carry propagation time in a parallel adder.

•The most widely used technique employs the principle of ***carry lookahead logic*** .



$$P_i = A_i \oplus B_i$$
$$G_i = A_iB_i$$

the output sum and carry can respectively be expressed as

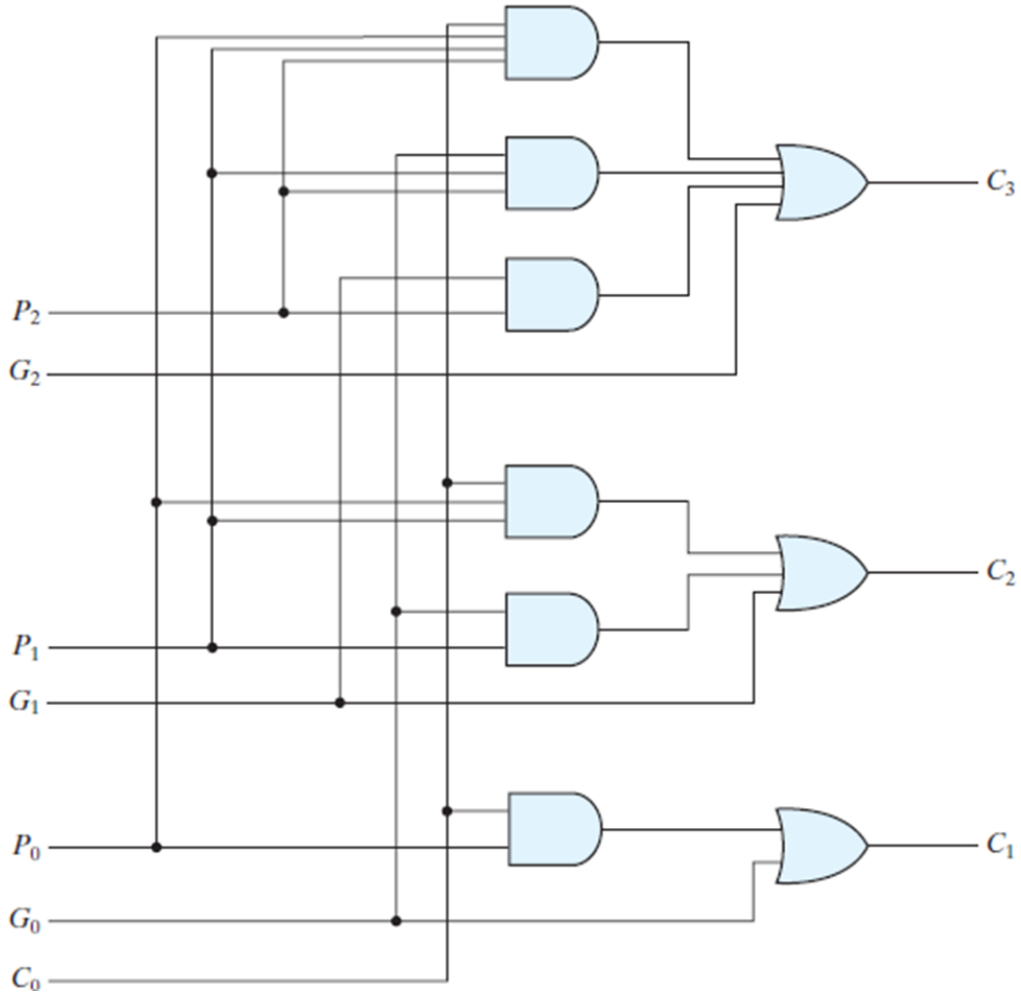
$$S_i = P_i \oplus C_i$$
$$C_{i+1} = G_i + P_iC_i$$

•*Gi* is called a ***carry generate*** , and it produces a carry of 1 when both *Ai* and *Bi* are 1, regardless of the input carry *Ci*.
•*Pi* is called a ***carry propagate*** , because it determines whether a carry into stage *i* will propagate into stage *i* + 1
(i.e., whether an assertion of *Ci* will propagate to an assertion of *Ci*+1).

$$C_0 = \text{input carry}$$
$$C_1 = G_0 + P_0C_0$$

$$C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$$
$$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$

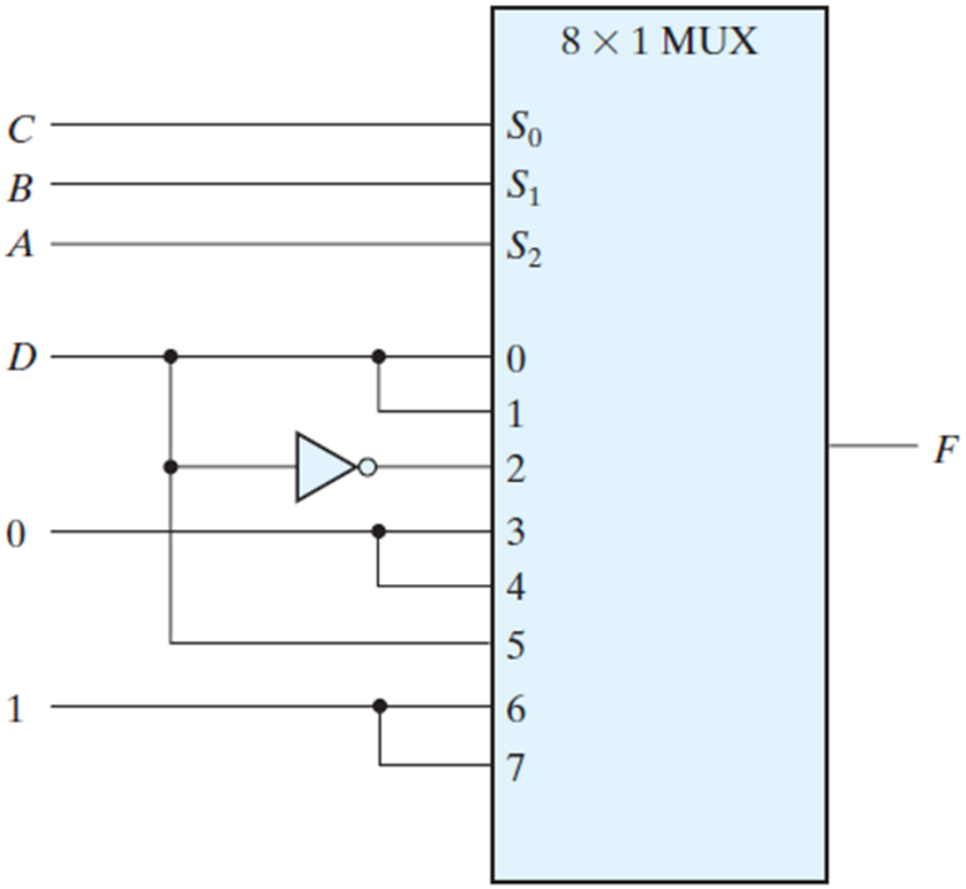
•Since the Boolean function for each output carry is expressed in sum-of-products form, each function can be implemented with one level of AND gates followed by an OR gate (or by a two-level NAND).
•This circuit can add in less time because *C*3 does not have to wait for *C*2 and *C*1 to propagate;
•*C*3 is propagated at the same time as *C*1 and *C*2.
•This gain in speed of operation is achieved at the expense of additional complexity (hardware).



4a

$$F(A, B, C, D) = \sum(1, 3, 4, 11, 12, 13, 14, 15)$$

A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = D'$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	



4b Encoder

•An encoder is a digital circuit that performs the inverse operation of a decoder.

• An encoder has 2^n (or fewer) input lines and *n* output lines.

•The output lines, as an aggregate, generate the binary code corresponding to the input value.

•An example of an encoder is the octal-to-binary encoder whose truth table is given in Table .

•It has eight inputs (one for each of the octal digits) and three outputs that generate the corresponding binary number.
It is assumed that only one input has a value of 1 at any given time

Truth Table of an Octal-to-Binary Encoder

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

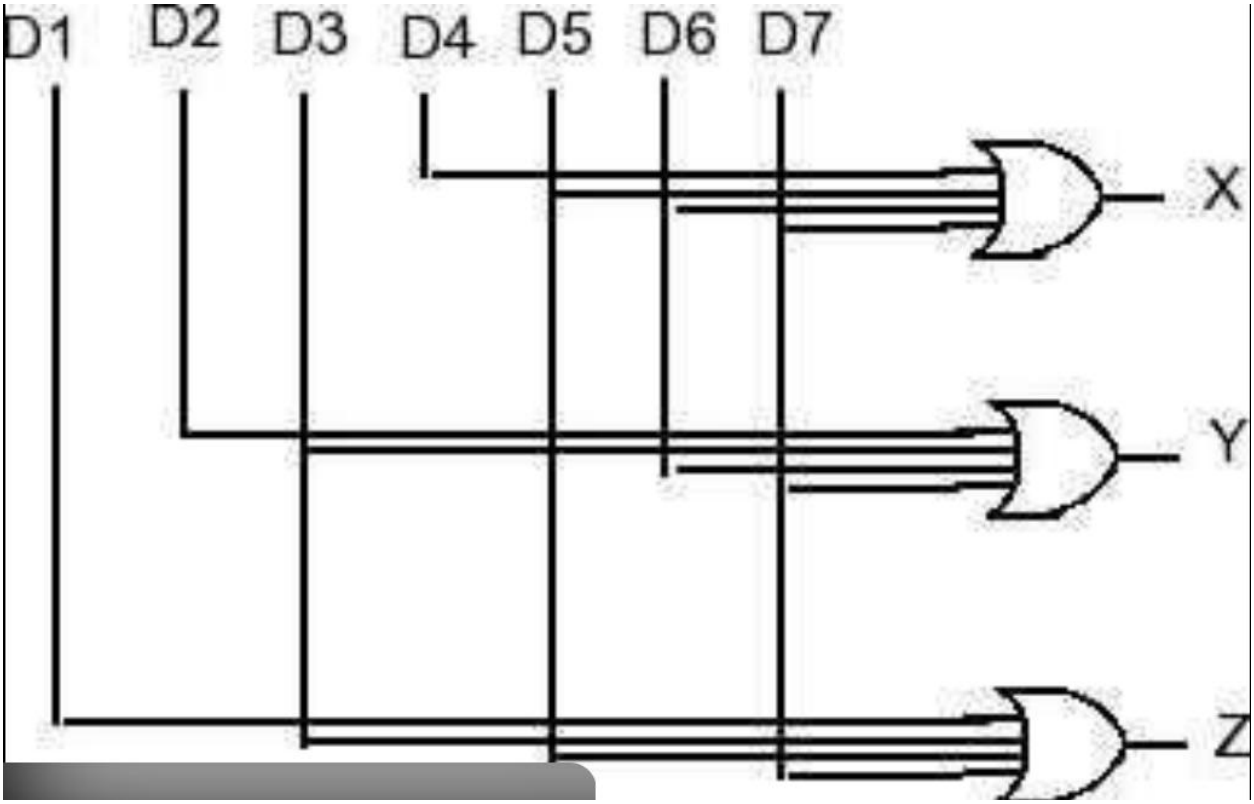
$$z = D_1 + D_3 + D_5 + D_7$$

$$y = D_2 + D_3 + D_6 + D_7$$

$$x = D_4 + D_5 + D_6 + D_7$$

•The encoder can be implemented with OR gates whose inputs are determined directly from the truth table.

•Output *z* is equal to 1 when the input octal digit is 1, 3, 5,or 7.
Output *y* is 1 for octal digits 2, 3, 6, or 7, and output *x* is 1 for digits 4, 5, 6, or 7



APPLICATION

4c

Storage Elements:

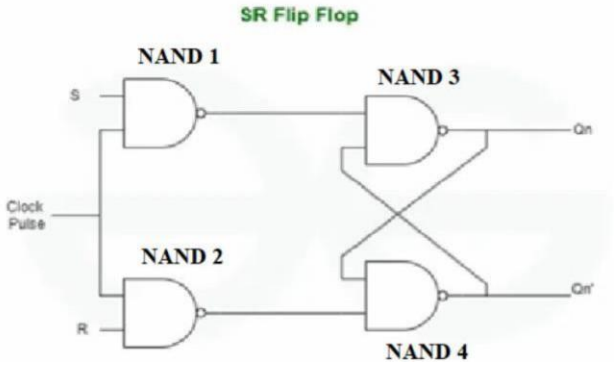
1)Latches:

- Latches are digital circuits that serve as basic building blocks in the construction of sequential logic circuits.
- They are **bistable**, meaning they **have two stable states** and can be used to store binary information. Latches are often used for temporary storage of data within a digital system.
- There are several types of latches, with the most common being the 1)SR latch (Set-Reset latch), 2)D latch (Data latch),3) JK latch.
- Storage elements that operate with signal **levels** (rather than signal transitions) are referred to as **latches**; those **controlled by a clock transition** are **flip-flops**.Latches are said to be level sensitive devices; flip-flops are edge-sensitive devices. The two types of storage elements are related because latches are the basic circuits from which all flip-flops are constructed.

S-R f/f

S	R	Qn+1	State
0	0	Qn	Hold
0	1	0	Reset
1	0	1	Set
1	1	X	Invalid

Working of SR Flip Flop



Case 1 : when S=0 and R=0, Clock is Positive Edge triggered (Value 1) then output of NAND gate 1 and NAND gate 2 will be 1. Considering previous output Q_n as 0, will make NAND 4 output which is Q'_{n+1} as 1 and this value will turn NAND gate 3 output which is Q_{n+1} to be 0.

Considering previous output Q_n as 1, will make NAND 4 output which is Q'_{n+1} as 0 and this value will turn NAND gate 3 output which is Q_{n+1} to be 1.

This is how the characteristic table is updated with values.

In this case Q_{n+1} is found to be equal to Q_n. Hence the state is called Hold state.

Case 2 : When S=0 and R=1 , Clock is Positive Edge triggered, then output of NAND gate 1 will be 1 and NAND gate 2 will be 0.

Considering previous output Q_n as 0, will make NAND 4 output which is Q'_{n+1} as 1 and this value will turn NAND gate 3 output which is Q_{n+1} to be 0.

Considering previous output Q_n as 1, will make NAND 4 output which is Q'_{n+1} as 1 and this value will turn NAND gate 3 output which is Q_{n+1} to be 0.

In this case for all the values of Q_n, the value of Q_{n+1} is found to be equal to 0.

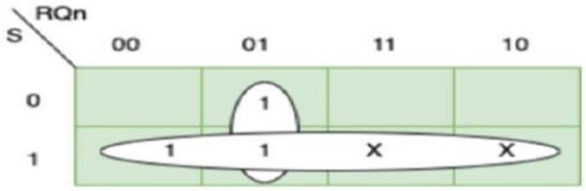
Case 3 :When S=1, R=0, Clock is Positive Edge triggered, for all values of Q_n, the value of Q_{n+1} is found to be equal to 1. Hence this state is called Set state.

Case 4 : When S=1 and R=1, Clock is Positive Edge triggered, for all vaues of Q_n, the values of Q_{n+1} and Q'_{n+1} are found to be equal which is indeterminate or impossible or invalid. Hence this state is called **Invalid State.**

Here, S is the Set input, R is the reset input, Q_n is the previous output and Q_{n+1} is the present output.

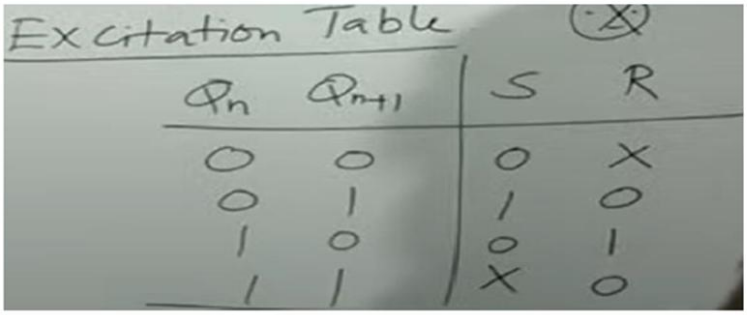
S	R	Qn	Qn+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	Invalid
1	1	1	Invalid

Characteristic Equation:
The characteristic equation tells us about what will be the next state of flip flop in terms of present state.



The characteristic equation for SR Flipflop will be

$Q_{n+1} = S + Q_nR'$



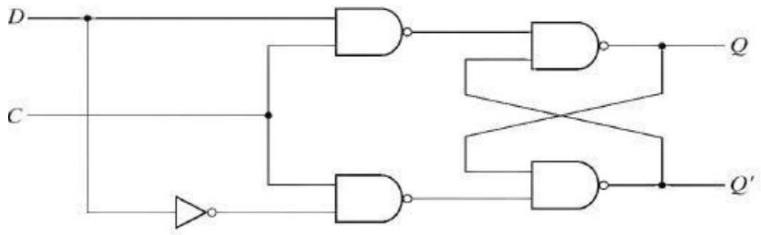
3. D Flip Flop

- D flip flop is actually a slight modification of the above explained clocked SR flip-flop. From the figure you can see that the D input is connected to the S input and the complement of the D input is connected to the R input.
- The D input is passed on to the flip flop when the value of CP is '1'.
- When CP is HIGH, the flip flop moves to the SET state. If it is '0', the flip flop switches to the CLEAR state.
- As long as the clock input C = 0, the SR latch has both inputs equal to 0 and it can't change its state regardless of the value of D
- When C is 1, the latch is placed in the set or reset state based on the value of D.

  If D = 1, the Q output goes to 1.

  If D = 0, the Q output goes to 0.

Circuit Diagram, Truth table
Characteristic Table, K-map and Characteristic Equation



(a) Logic diagram

C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

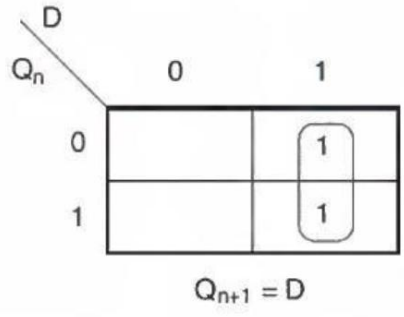
(b) Function table

Fig. 5-6 D Latch

Excitation Table:

Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

K- Map for Q_{n+1}:



5. a.)What do you mean by an addressing mode? Explain any 5 addressing modes?
Definition of Addressing mode 2 marks

An addressing mode is the method used by the CPU to specify the location of the operand (data) required for an instruction. It tells the processor where the data is stored.

- Any Five Addressing Modes (8 Marks)
- 1. Immediate Addressing Mode
- Data is given directly in the instruction.
  Example: MOV A, #10
- 2. Register Addressing Mode
- Operand is stored in a CPU register.
  Example: MOV A, R1
- 3. Direct Addressing Mode
- Address of operand is directly specified in the instruction.
  Example: MOV A, 2000H
- 4. Indirect Addressing Mode
- Address of operand is stored in a register.
  Example: MOV A, @R1
- 5. Indexed Addressing Mode

Effective address = Base address + Index value.
Used for arrays.
Example: MOV A, 2000H(R1)

b) Describe the Big-endian and Little-endian address assignment

1. Big-Endian (1 Marks)

- The **Most Significant Byte (MSB)** is stored at the lowest memory address.
  Used in some processors like Motorola, and network protocols.

2. Little-Endian (1 Marks)

- The **Least Significant Byte (LSB)** is stored at the lowest memory address.
  Used in Intel processors.
- Explanation 1marks

  Diagram- 1marks

C. A program with 5000 machine instructions, need an average of 3 basic steps to execute one instruction. Find the performance of the computer having a clock speed of 500 k.

5marks

Given:
Number of instructions = 5000
Average basic steps per instruction = 3
Clock speed = 500 kHz (500,000 cycles/second)
Total cycles = 5000 × 3 = 15000 cycles
Execution Time = Total cycles / Clock speed
= 15000 / 500000
= 0.03 seconds
Performance = 1 / Execution Time
= 1 / 0.03
= 33.33 programs per second
Final Answer:
Execution Time = 0.03 seconds
Performance ≈ 33.33 programs per second

6. a) demonstrate the branching operational using loop to add n numbers with block diagram 8 marks

Defintion : 1 Marks

Branching operation allows the control of a program to jump from one instruction to another based on a condition.
Loops use branching instructions to repeat a set of instructions until a condition is satisfied.

Algorithm:(2marks)
1. Start
2. Initialize SUM = 0
3. Read n
4. Initialize i = 1
5. Read number
6. SUM = SUM + number
7. i = i + 1
8. If i ≤ n, branch to Step 5
9. Display SUM
10. Stop

Block diagram (3 marks)

Explanation (2marks)

6.b) show how below expression will be executed in one address and three address processor in accumulator organization X= (A*B)+(C*D) 7 marks

Expression: X = (A * B) + (C * D)

One-Address (Accumulator Organization): (3marks)

```
LOAD A
MUL B
STORE T
LOAD C
MUL D
ADD T
STORE X
```

Three-Address Processor: (4marks)
MUL T1, A, B
MUL T2, C, D
ADD X, T1, T2

6 C. what are condition code flags? mention the significance of the flag N,Z,V and C ?

Condition code flags are special bits in the status register that indicate the result of arithmetic or logical operations. (2 Marks)

N (Negative Flag): Set to 1 if result is negative.
Z (Zero Flag): Set to 1 if result is zero.
V (Overflow Flag): Set to 1 if signed overflow occurs.
C (Carry Flag): Set to 1 if carry or borrow occurs.

Explanation: 3marks

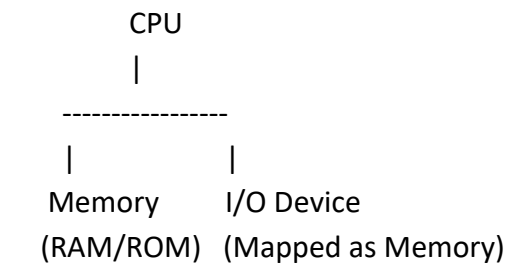
7. a) Explain memory mapped I/O and I/O interface for the input device with a diagram?

Memory Mapped I/O (5marks)

Memory Mapped I/O is a technique in which input/output devices are assigned specific memory addresses. The CPU treats I/O devices as memory locations and uses normal memory instructions to access them.

Features:
- I/O devices share the same address space as memory.
- No separate I/O instructions are required.
- Uses memory control signals.

Diagram:



I/O Interface for Input Device (5marks)

An I/O interface is a hardware module that connects input devices to the CPU.

Functions:
- Data buffering
- Status reporting
- Control signal generation

Block Diagram:
Input Device → Input Buffer → Status Register → Control Logic → CPU

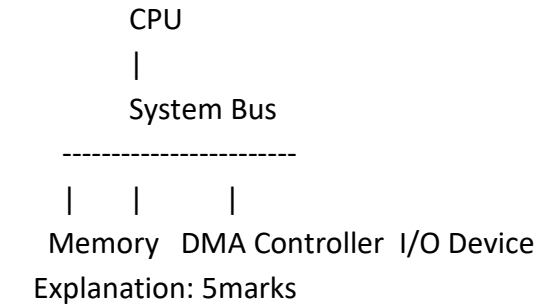
Working:
1. Input device sends data to buffer.
2. Status register indicates data ready.
3. CPU reads data from buffer.

7b. explain DMA with a neat diagram

Direct Memory Access (DMA) is a technique that allows an I/O device to transfer data directly to or from main memory without continuous CPU involvement. (2 Marks)

Need for DMA:
- Reduces CPU workload
- Increases transfer speed
- Efficient for large data transfers

Block Diagram: (3marks)



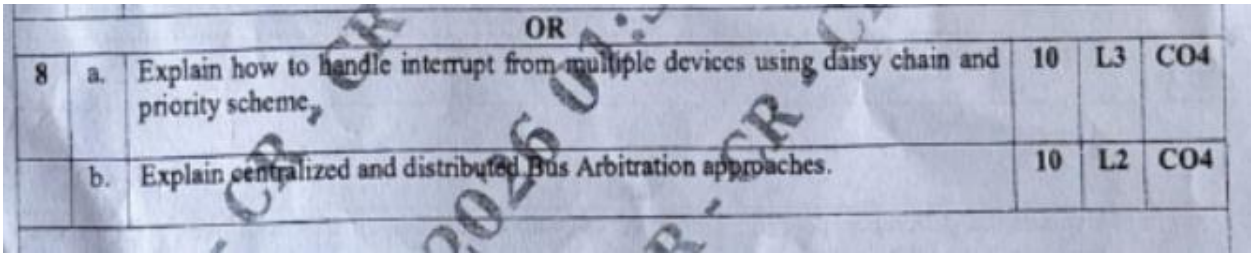
Explanation: 5marks

Working:
1. CPU initializes DMA controller with address and count.
2. DMA requests bus control.
3. CPU grants bus.
4. DMA transfers data directly.

5. DMA interrupts CPU after completion.

DMA Modes:

- Burst Mode
- Cycle Stealing Mode
- Transparent Mode



		OR			
8	a.	Explain how to handle interrupt from multiple devices using daisy chain and priority scheme.	10	L3	CO4
	b.	Explain centralized and distributed Bus Arbitration approaches.	10	L2	CO4

Schemes:

8 a.

Definition: 2 Marks

Classification: 1 Marks

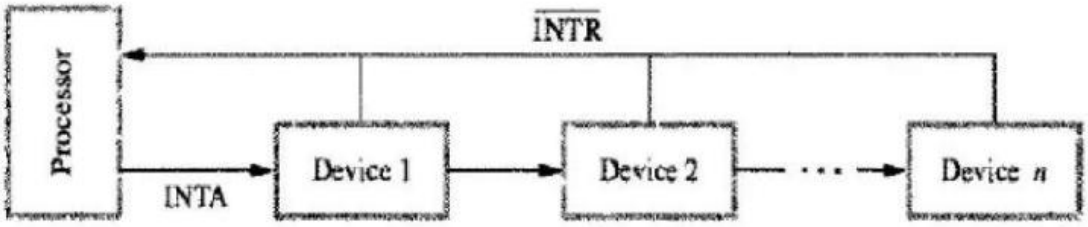
Explanation: 5 Marks

Diagram : 2marks

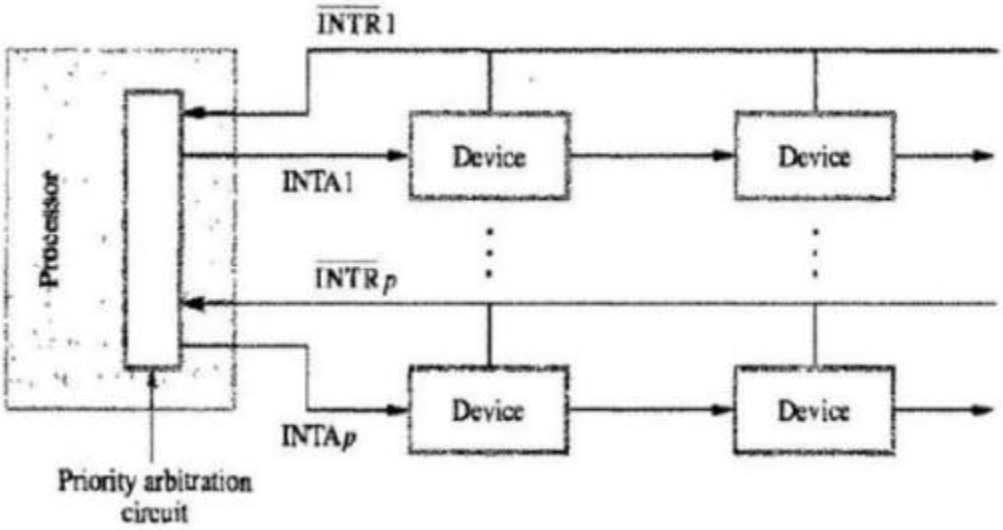
Let us now consider the situation where a number of devices capable of initiating interrupts are connected to the processor. Because these devices are operationally independent, there is no definite order in which they will generate interrupts. For example, device X may request an interrupt while an interrupt caused by device Y is being serviced, or several devices may request interrupts at exactly the same time. When an interrupt request is received it is necessary to identify the particular device that raised the request. Furthermore, if two devices raise interrupt requests at the same time it must be possible to break the tie and select one of the two requests for service. When the interrupt-service routine for the selected device has been completed, the second request can be serviced.

Daisy Chain

We also need to consider the problem of simultaneous arrivals of interrupt requests from two or more devices. The processor must have some means of deciding which request to service first. Polling the status registers of the I/O devices is the simplest such mechanism. In this case, priority is determined by the order in which the devices are polled. When vectored interrupts are used, we must ensure that only one device is selected to send its interrupt vector code. A scheme to connect device to form daisy chain and its arrangement are shown in Figure.



Here, the interrupt request line INTR is common to all devices. The interrupt acknowledge line INTA is connected in a daisy chain fashion such that INTA signal propagates serially through the devices. When several devices raise an interrupt request, the INTR line is activated & the processor responds by setting INTA line to 1. This signal is received by device 1 Devicel passes the signal on to device2 only if it does not require any service. If devices1 has a pending request for interrupt blocks that INTA signal & proceeds to put its identification code on the data lines. Therefore, the device that is electrically closest to the processor has the highest priority. The advantage of using this scheme is that it requires fewer wires than the individual connections.



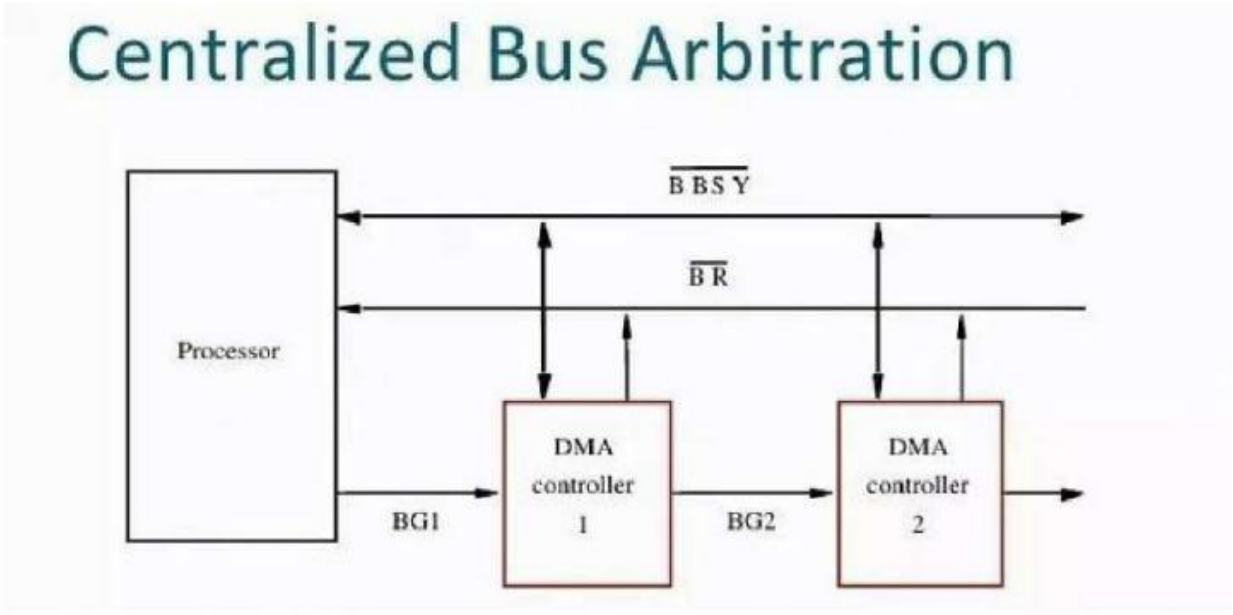
8 b.

Definition: 2 Marks
Classification: 1 Marks
Explanation: 5 Marks
Diagram : 2marks

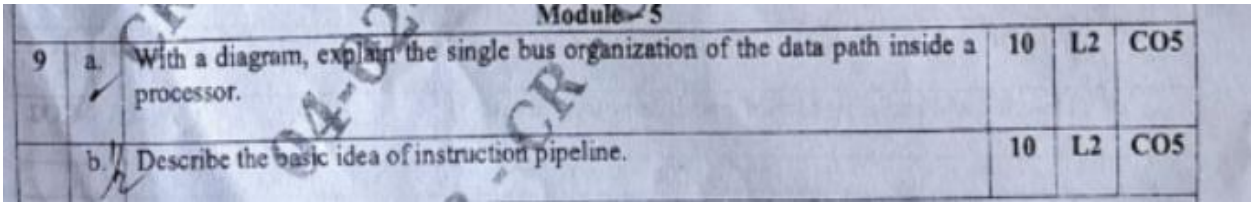
A device that initiates data transfers on the bus at any given time is called a bus master.In computer system, there may be more than one bus master such as a DMA controller or rocessor etc. These devices share the system bus and when a current master bus relinquishes another bus can acquire the control of the processor. Bus arbitration is a process by which next device becomes the bus controller by transferring bus mastership to another bus. Types of Bus Arbitration There are two types of bus arbitration namely centralized and distributed arbitration In centralized arbitration, a single bus-arbiter performs the required arbitration. In distributed arbitration, all devices participate in selection of next bus-master.

Centralized arbitration

A single bus-arbiter performs the required arbitration as shown in the figure. Normally, processor is the bus master unless it grants bus mastership to one of the DMA controllers. DMA controller indicates that it needs to become bus-master by activating the Bus-Request line(BR). The signal on the BR line is the logical OR of bus-requests from all devices connected to it. When BR is activated, processor activates Bus-Grant signal (BG1) indicating to DMA controllers that they may use bus when it becomes free. (This signal is connected to all DMA controllers using a daisy-chain arrangement). If DMA controller-1 is requesting the bus, it blocks propagation of grant-signal to other devices. Otherwise, it passes the grant downstream by asserting BG2. Current bus-master indicates to all devices that it is using the bus by activating Bus-Busy line (BBSY).Arbiter circuit ensures that only one request is granted at any given time according to a predefined priority scheme.



Distributed Arbitration: All devices participate in the selection of next bus-master as shown in the figure. Each device on bus is assigned a 4-bit identification number (ID). When 1 or more devices request bus, they assert Start-Arbitration signal & place their 4-bit ID numbers on four open-collector lines ARB 0 through ARB 3. A winner is selected as a result of interaction among signals transmitted over these lines by all contenders. Net outcome is that the code on 4 lines represents request that has the highest ID number. Main advantage: This approach offers higher reliability since operation of bus is not dependent on any single device.



		Module-5			
9	a.	With a diagram, explain the single bus organization of the data path inside a processor.	10	L2	CO5
	b.	Describe the basic idea of instruction pipeline.	10	L2	CO5

9 a.
Definition: 2 Marks
Explanation: 5 Marks
Diagram : 3marks

To execute an instruction, processor has to perform following 3 steps: 1) Fetch contents of memory-location pointed to by PC. Content of this location is an instruction to be executed. The instructions are loaded into IR, Symbolically, this operation can be written as IR←[[PC]] 2) Increment PC by 4 P,← [PC] +4 3) Carry out the actions specified by instruction (in the IR). The

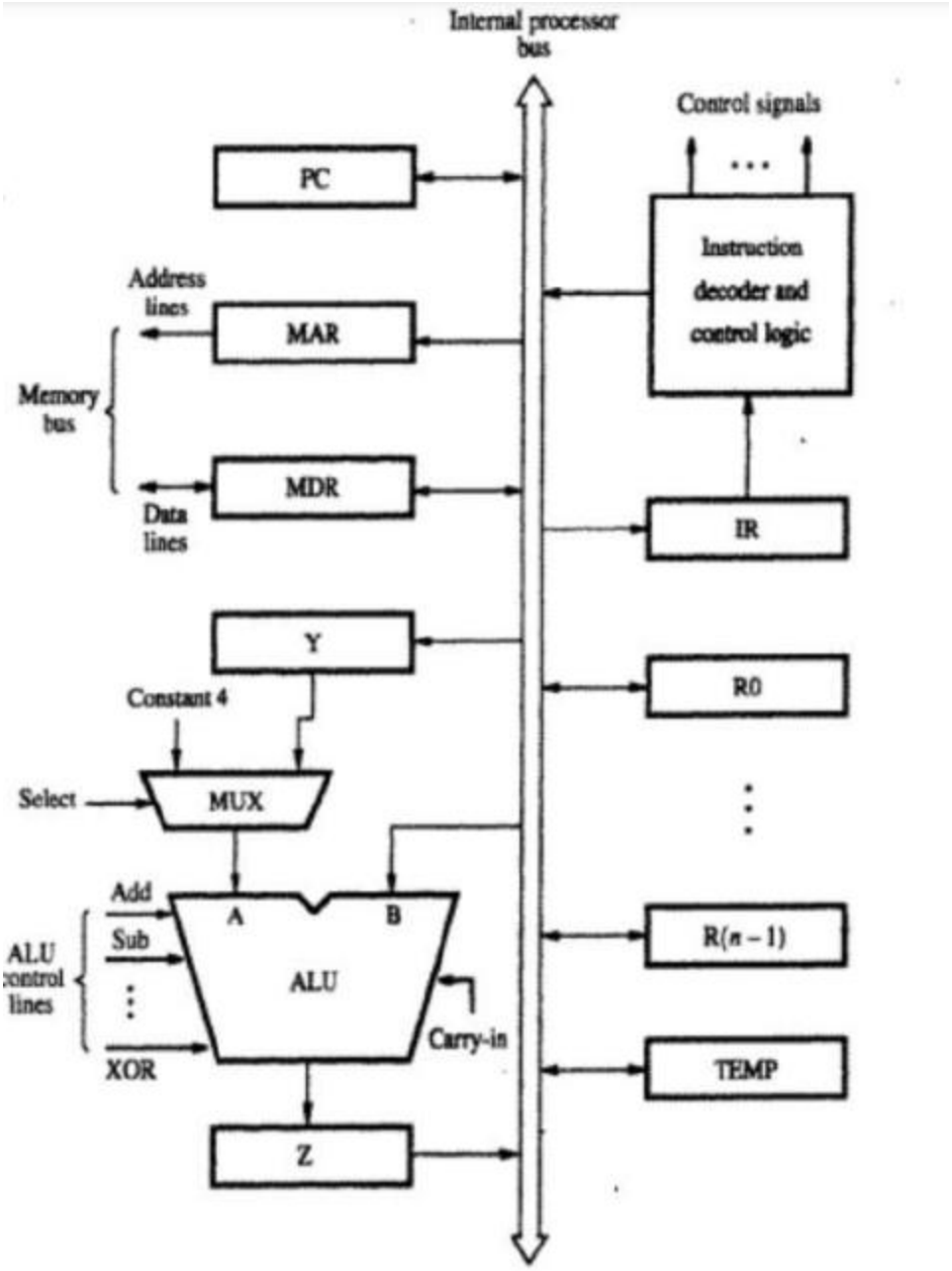
first 2 steps are referred to as fetch phase; Step 3 is referred to as execution phase. The operations specified by an instruction can be carried out by performing one or more of the following actions:

1.Read the content of a given memory-location and load them into a register.
2.Read data from one or more register
3.Perform ALU operations and place the result into the register.
4.Store data from a register into a given memory location.

Below figure shows the single bus organization. ALU and all the registers are interconnected via a single common bus. Data and address line of the external memory bus is connected to the internal processor bus via MDR and MAR respectively(MDR-Memory Data Register and MAR-Memory Address Register). MDR has 2 inputs and 2 outputs. Data may be loaded → into MDR either from memory-bus (external) or → from processor-bus (internal). MAR's input is connected to internal-bus, and MAR"s output is connected to externalbus.Instruction-decoder & control-unit is responsible for → issuing the signals that control the operation of all the units inside the processor (and for interacting with memory bus).

→ implementing the actions specified by the instruction (loaded in the IR) Registers R0 through R(n-1) are provided for general purpose use by programmer. Three registers Y, Z & TEMP are used by processor for temporary storage during execution of some instructions. These are transparent to the programmer i.e. programmer need not be concerned with them because they are never referenced explicitly by any instruction. MUX(Multiplexer) selects either → output of Y or → constant-value 4(is used to increment PC content).This is provided as input A of ALU. B input of ALU is obtained directly from processor-bus. As instruction execution progresses, data are transferred from one register to another, often passing through ALU to perform arithmetic or logic operation. An instruction can be executed by performing one or more of the following operations:

1) Transfer a word of data from one processor-register to another or to the ALU.
2) Perform arithmetic or a logic operation and store the result ina processor-register.
3) Fetch the contents of a given memory-location and load them into a processorregister.
4) Store a word of data from a processor-register into a given memory-location.


Internal processor bus
Control signals
PC
Instruction decoder and control logic
Address lines
MAR
Memory bus
MDR
Data lines
IR
Y
R0
Constant 4
Select
MUX
Add
Sub
ALU control lines
XOR
A
B
ALU
Carry-in
R(n – 1)
TEMP
Z

9 b.
Definition: 2 Marks
Classification: 1 Marks
Explanation: 5 Marks
Diagram : 2marks

The speed of execution of programs is influenced by many factors.
1. One way to improve performance is to use faster circuit technology to implement the processor and the main memory.
2. Another possibility is to arrange the hardware so that more than one operation can be performed at the same time. In this way, the number of operations performed per second is increased, even though the time needed to perform any one operation is not changed. Pipelining is a particularly effective way of organizing concurrent activity in a computer system. Consider how the idea of pipelining can be used in a computer. The processor executes a program by fetching and executing instructions, one after the other. Let Fi and Ei refer to the fetch and execute steps for instruction Ii.

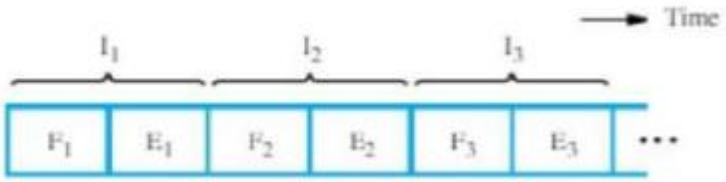
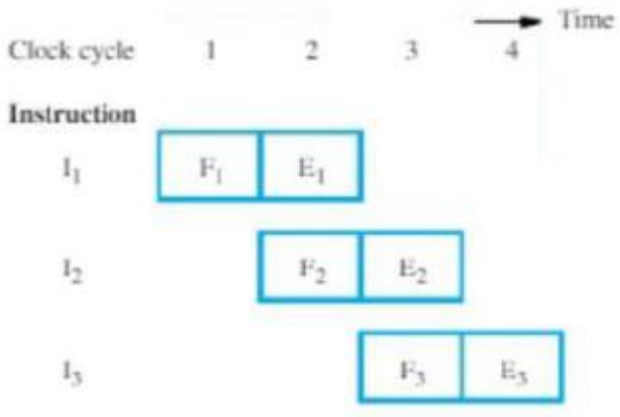


Fig: Sequential execution



The instruction fetched by the fetch unit is deposited in an intermediate storage buffer, B1. This buffer is needed to enable the execution unit to execute the instruction while the fetch unit is fetching the next instruction. The results of execution are deposited in the destination location specified by the instruction. Operation of the computer proceeds as in Figure 5.9. In the first clock cycle, the fetch unit fetches an instruction I1 (step F1) and stores it in buffer B1 at the end of the

clock cycle. In the second clock cycle, the instruction fetch unit proceeds with the fetch operation for instruction 12 (step F2). Meanwhile, the execution unit performs the operation specified by instruction I1, which is available to it in buffer B1 (step E1). By the end of second clock cycle, the execution of instruction Il is completed and instruction 12 is available. Instruction 12 is stored in B1,replacing I1,which is no longer needed. StepE2 is performed by the execution unit during the third clock eycle, while instruction 13 is being fetched by the fetch unit. In this manner, both the fetch and execute units are kept busy all the time. In summary, the fetch and execute units in Figure 5.3 constitute a two-stage pipeline in which each stage performs one step in processing an instruction. An inter-stage storage buffer, B1, is needed to hold the information being passed from one stage to the next. New information is loaded into this buffer at the end of each clock cycle. The processing of an instruction need not be divided into only two steps. For example, a pipelined processor may process each instruction in four steps, as follows: F Fetch: read the instruction from the memory. D Decode: decode the instruction and fetch the source operand(s). E Execute: perform the operation specified by the instruction. WWrite: store the result in the destination location.

		OR			
10	a.	Explain the process of fetching word from memory in processor.	10	L4	CO5
	b.	Explain the pipeline performance of a processor and pipeline stalls.	10	L2	CO5

10 a.
Definition: 2 Marks
Sequence: 3 Marks
Explanation: 3 Marks
Diagram : 2marks

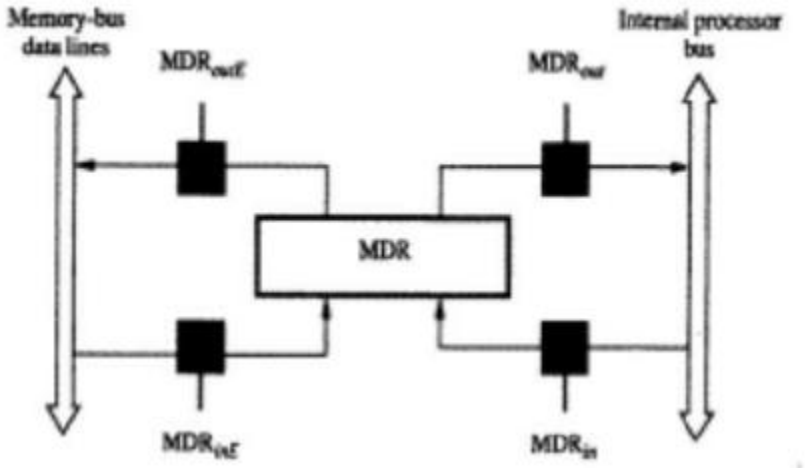
To fetch instruction/data from memory, processor transfers required address to MAR (whose output is connected to address-lines of memory-bus). At the same time, processor issues Read signal on control-lines of memory-bus. When requested-data are received from memory, they are stored in MDR. From MDR, they are transferred to other registers. The response time of each memory access varies. For this MFC (Memory Function Completed): is used. It is the signal sent from Addressed-device to the processor. MFC informs the processor that the requested operation is completed by addressed device. Thus MFC is set to 1 to indicate that the contents of the specified location
Move (R1),R2.

The sequence of steps is:

1) R1out MARin Read ;desired address is loaded into MAR & Read command is issued

2) MDRinE, WMFC ;load MDR from memory bus & Wait for MFC response from memory.

3) MDRout, R2in ;load R2 from MDR where WMFC=control signal that causes processor's control circuitry to wait for arrival of MFC signal .



10 b.

Definition: 2 Marks

Classification: 1 Marks

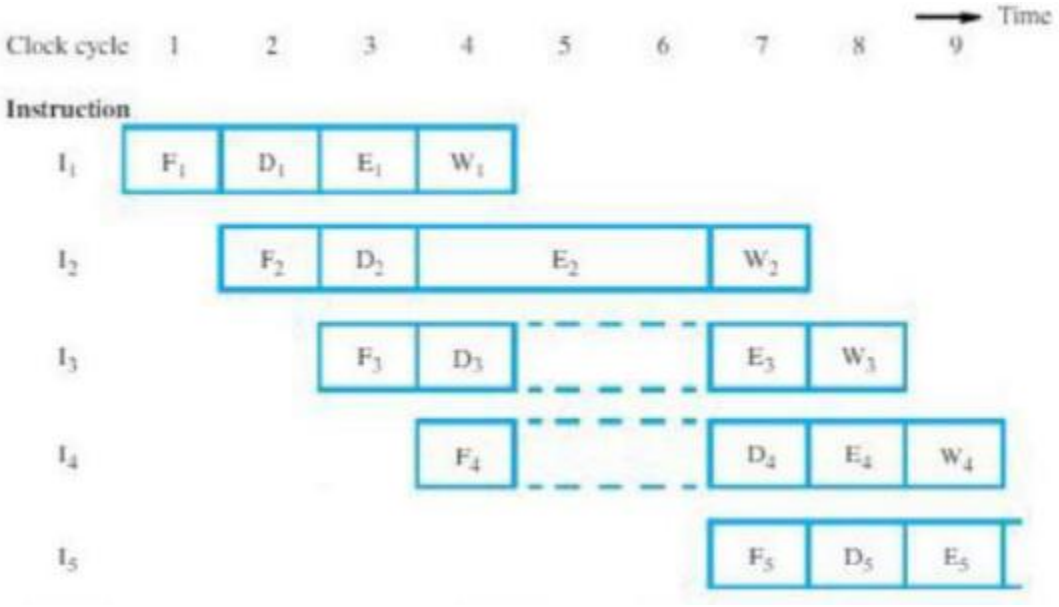
Explanation: 5 Marks

Diagram : 2marks

The pipelined processor in Figure 5.6 completes the processing of one instruction in each clock cycle, which means that the rate of instruction processing is four times that of sequential operation. The potential increase in performance resulting from pipelining is proportional to the number of pipeline stages. Let us consider an example of, one of the pipeline stages may not be able to complete its processing task for a given instruction in the time allotted as in Figure .

Here instruction 12 requires three cycles to complete, from cycle 4 through cycle 6. Thus, in cycles 5 and 6, the Write stage must be told to do nothing, because it has no data to work with. Meanwhile, the information in buffer B2 must remain intact until the Execute stage has completed its operation. This means that stage 2 and, in turn, stagel are blocked from accepting new instructions because the infornnation in B1 cannot be overwritten. Thus, steps D4 and F5 must be postponed. Pipelined operation in Figure is said to have been stalled for two clock cycles.

Normal pipelined operation resumes in cycle 7.



Any condition that causes the pipeline to stall is called a hazard. There are three types of Hazards:

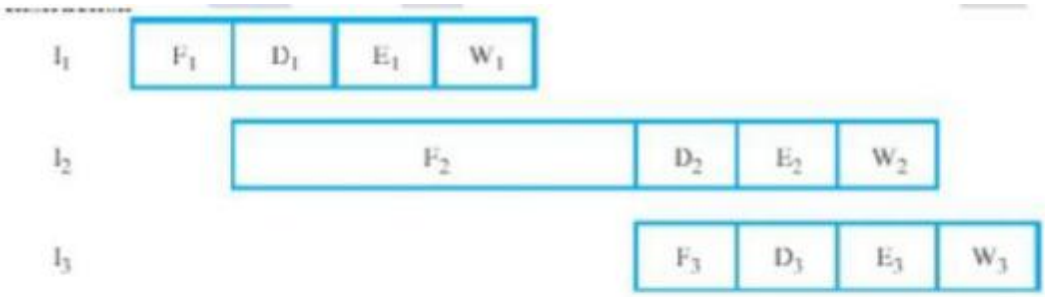
1. Data hazard
2. Instruction or control hazard
3. Structural hazard

Data hazard

A data hazard is any condition in which either the source or the destination operands of an instruction are not available at the time expected in the pipeline. As a result some operation has to be delayed, and the pipeline stalls.

Control hazards or instruction hazards

The pipeline may also be stalled because of a delay in the availability of an instruction. For example, this may be a result of a miss in the cache, requiring the instruction to be fetched from the main memory. Such hazards are often called control hazards or instruction hazards. Figure has an instruction hazard with it. Instruction Il is fetched from the cache in cycle, and its execution proceeds normally. However, the fetch operation for instruction I2, which is started in cycle 2,results in a cache miss. The instruction fetch unit must now suspend any further fetch requests and wait for 12 to arrive. We assume that instruction 12 is received and loaded into buffer B1 at the end of cycle 5. The pipeline resumes its normal operation at that point.



(a) Instruction execution steps in successive clock cycles

Clock cycle	1	2	3	4	5	6	7	8	9 → Time
Stage									
F: Fetch	F_1	F_2	F_2	F_2	F_2	F_3			
D: Decode		D_1	idle	idle	idle	D_2	D_3		
E: Execute			E_1	idle	idle	idle	E_2	E_3	
W: Write				W_1	idle	idle	idle	W_2	W_3

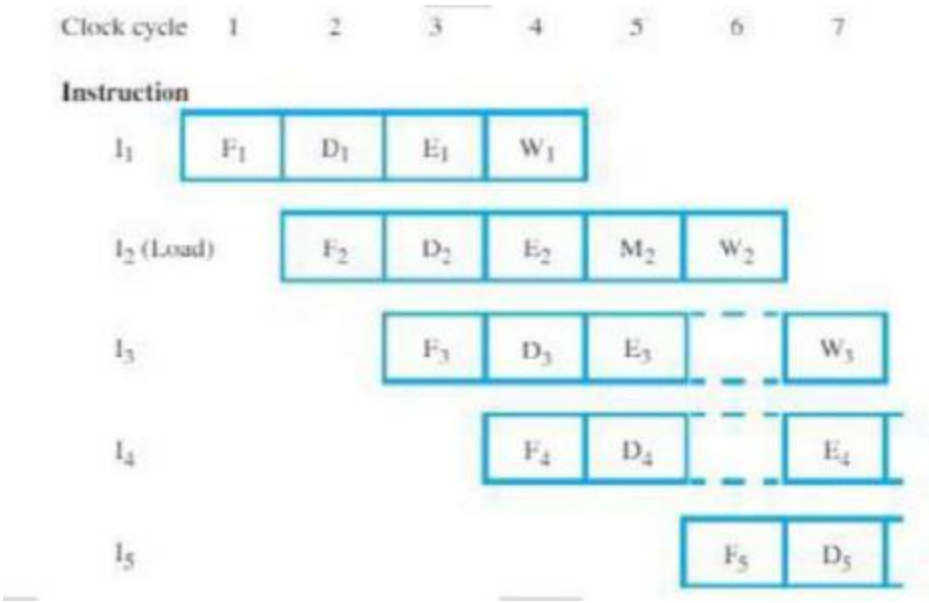
(b) Function performed by each processor stage in successive clock cycles

Structural hazard

A third type of hazard that may be encountered in pipelined operation is known as a structural hazard. This is the situation when two instructions require the use of a given hardware resource at the same time.

Example: Load X(R1),R2

The memory address, X+[R1], is computed in step E2 in cycle4, then memory access takes place in cycle5.The operand read from memory is written into register R2 in cycle 6. This means that the execution step of this instruction takes two clock cycles (cycles 4 and 5). It causes the pipeline to stall for one cycle, because both instructions 12 and 13 require access to the register file in cycle 6 which is shown in Figure


Clock cycle 1 2 3 4 5 6 7
Instruction
I1
F1 D1 E1 W1
I2 (Load)
F2 D2 E2 M2 W2
I3
F3 D3 E3 W3
I4
F4 D4 E4
I5
F5 D5