SCHEME OF VALUATION

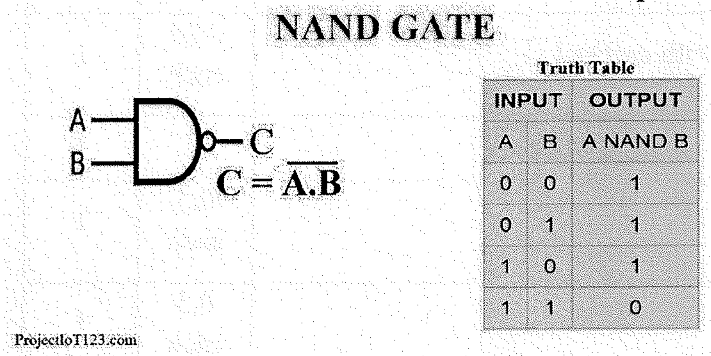
Scoring Indicators

Revision: 2021

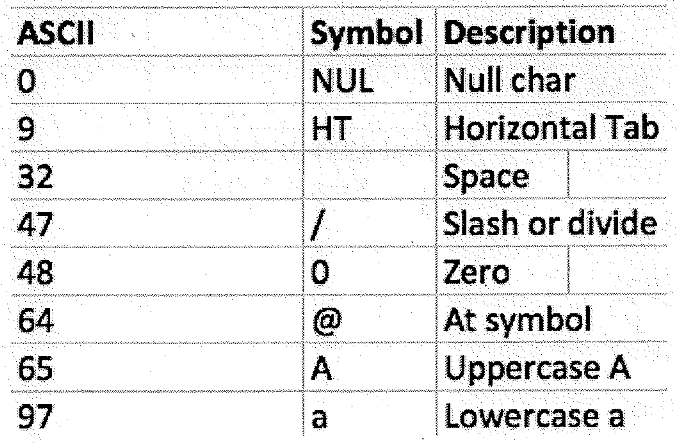
COURSE NAME: LOGIC SYSTEM DESIGN & COMPUTER ORGANIZATION

COURSE CODE: 3342 **QID: 2109230387**

Qst. No.	Scoring Indicators	Split score	Sub Total	Total score
	PART A			9
I.1	Find 1's complement and add 1 to it	1	1	
I.2	0+0=0; 0+1=1; 1+0=1; 1+1=0, with carry=1	0.25 each	1	
I.3	$\bar{A}.\bar{B}$	1	1	
I.4	Flipflops, counter	0.5 each	1	
I.5	NAND, NOR	0.5 each	1	
I.6	Processor, Memory, ALU, Control unit	0.25 each	1	
I.7	Address bus, Data bus, Control bus	1	1	
I.8	Hardwired control, microprogrammed, sequential control, pipelined control etc.	0.25 each	1	
I.9	SISD, SIMD, MISD, MIMD	0.25 each	1	
	PART B			24
II.1	**Final answer-1 mark** **Steps- 2 marks** 102 / 2 = 51, remainder = 0 (R1) 51 / 2 = 25, remainder = 1 (R2) 25 / 2 = 12, remainder = 1 (R3) 12 / 2 = 6, remainder = 0 (R4) 6 / 2 = 3, remainder = 0 (R5) 3 / 2 = 1, remainder = 1 (R6) 1 / 2 = 0, remainder = 1 (R7)	2	3	

<table>
<tr><td></td><td>Now, read the binary result by combining the remainders from bottom to top: 1100110.
So, the binary representation of 102 is 1100110.</td><td>1</td><td></td><td></td></tr>
<tr><td>II.2</td><td><b>Symbol – 1 mark</b>
<b>Truth table- 1 mark</b>
<b>Expression- 1 mark</b>


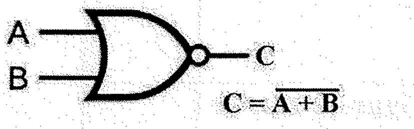
NAND GATE
A
B
C
C = $\overline{A.B}$
ProjectIoT123.com

Truth Table
<table><tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr><tr><td>A</td><td>B</td><td>A NAND B</td></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table></td><td>3</td><td>3</td><td></td></tr>
<tr><td>II.3</td><td><b>Explanation-1.5 marks</b>
<b>Examples-1.5 marks</b>
An alphanumeric code is a system that represents both letters and numbers using a combination of characters; for instance, the ASCII code assigns unique numeric values to letters, digits, and symbols to facilitate digital text and data representation.

<table><tr><th>ASCII</th><th>Symbol</th><th>Description</th></tr><tr><td>0</td><td>NUL</td><td>Null char</td></tr><tr><td>9</td><td>HT</td><td>Horizontal Tab</td></tr><tr><td>32</td><td></td><td>Space</td></tr><tr><td>47</td><td>/</td><td>Slash or divide</td></tr><tr><td>48</td><td>0</td><td>Zero</td></tr><tr><td>64</td><td>@</td><td>At symbol</td></tr><tr><td>65</td><td>A</td><td>Uppercase A</td></tr><tr><td>97</td><td>a</td><td>Lowercase a</td></tr></table></td><td>1.5
1.5</td><td>3</td><td></td></tr>
<tr><td>II.4</td><td></td><td></td><td></td><td></td></tr>
</table>

<table>
<tr><td></td><td>

Symbol – 1 mark
Truth table- 1 mark
Expression- 1 mark

NOR GATE



TRUTH TABLE

<table>
<tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr>
<tr><td>A</td><td>B</td><td>A NOR B</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>

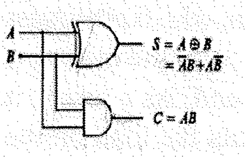
ProjectIoT123.com

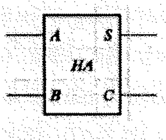
</td><td>3</td><td>3</td><td></td></tr>
<tr><td>II.5</td><td>

Symbol – 1 mark
Truth table- 1 mark
Expression- 1 mark

- A combinational circuit which adds two one-bit binary numbers is called a half-adder.

<table>
<tr><th colspan="2">Inputs</th><th colspan="2">Outputs</th></tr>
<tr><td>A</td><td>B</td><td>Carry</td><td>Sum</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td></tr>
</table>





- The sum column resembles like an output of the XOR gate.
- The carry column resembles like an output of the AND gate.

</td><td>3</td><td>3</td><td></td></tr>
<tr><td>II.6</td><td>

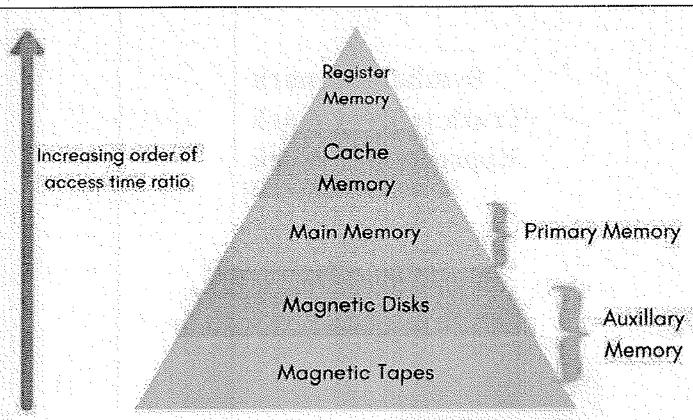
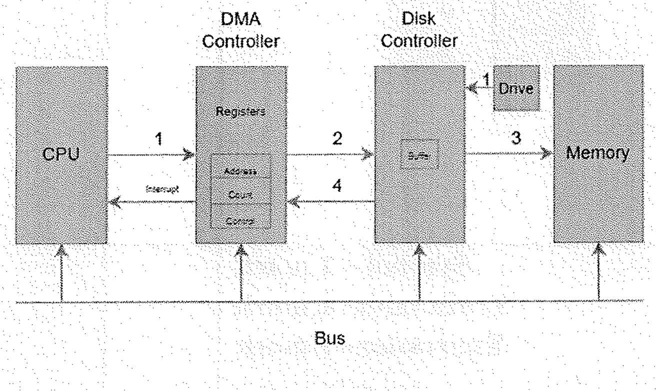
Expansion-1 mark

Example- 2 marks

Sum of Products=A·B+A·B

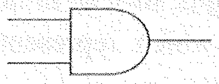
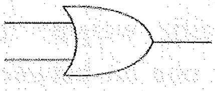
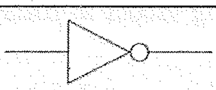
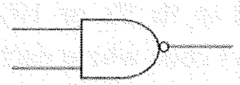
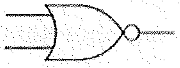
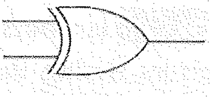
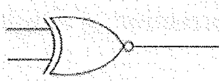
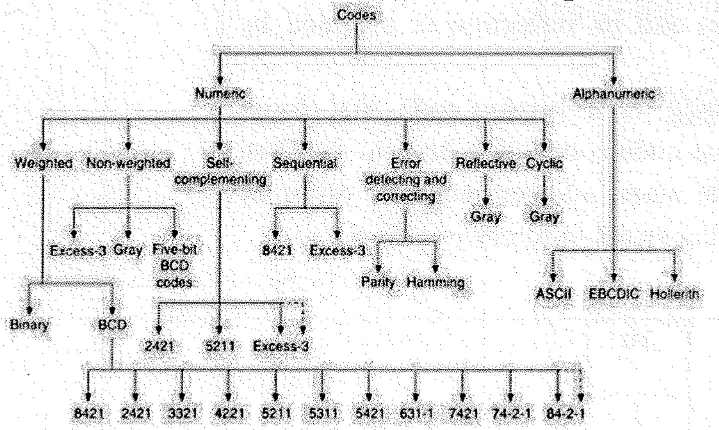
Product of Sum=(A+B)·(A+B)

</td><td>3</td><td>3</td><td></td></tr>
</table>

II.7	 Increasing order of access time ratio Register Memory Cache Memory Main Memory Primary Memory Magnetic Disks Auxillary Memory Magnetic Tapes	3	3	
II.8	**Figure-2 marks** **Explanation- 1 marks**  DMA Controller Disk Controller CPU Registers Address Count Control Buffer Drive Memory 1 2 3 4 1 Interrupt Bus DMA structure A **direct memory access (DMA)** controller is a device within the system that can facilitate data transfer between input/output devices within the system and the main memory without the CPU's intervention. This is done by the operating system, which programs the DMA controller by telling where the data lives on the memory, how much to copy, and which device it should send. As a result, it completely bypasses the CPU and, in turn, lessens the load on the CPU.	2 1	3	
II.9	**Definition- 1 mark** **Stages- 2 marks** Instruction pipelining is a technique used in computer architecture to improve the overall performance of instruction execution. • Fetch (IF): Fetch the instruction from memory. • Decode (ID): Decode the instruction to determine the operation to be performed. • Execute (EX): Perform the operation or calculate the effective address. • Memory (MEM): Access memory, if needed.	1 2	3	

	- Write Back (WB): Write the result back to a register.			
II.10	**Hardwired Control** Hardwired control involves using combinational logic circuits to generate control signals. **Characteristics:** 1. Direct mapping between instructions and control signals. 2. Fixed and inflexible, requires hardware modification for changes. **Advantages:** 1. Simple and fast execution. 2. Well-suited for simple instruction sets. **Disadvantages:** 1. Limited flexibility. 2. Complex for larger instruction sets. **Microprogramming** Microprogramming involves using a control memory that stores microinstructions for each machine language instruction. **Characteristics:** 1. Control unit executes a sequence of microinstructions for each instruction. 2. Microinstructions are stored in a control memory. **Advantages:** 1. More flexible than hardwired control. 2. Easier to modify and update. **Disadvantages:** 1. Slower execution compared to hardwired control. 2. Requires additional memory for the control store.	1.5 each	3	
	PART C			42
III	i. Convert 'A' to binary: $A_{16}=10_{10}=1010_2$ Convert 'B' to binary: $B_{16}=11_{10}= 1011_2$ Convert 'C' to binary: $C_{16}= 12_{10}= 1100_2$ Combine these binary representations: So, the hexadecimal number 0xABC is equal to the binary number 101010111100. ii. $123_8 =$ 1 2 3 001 010 011 On combining 001010011, then group it by 4 0101 0011 5 3 That is 0x53	3.5 each	7	
IV	i.			

	$0110.011_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}$ $= 16 + 4 + 1 + 0.25 + 0.125$ $= 22.375_{10}$ ii. $53.43_{10}=$ $53 \div 2 = 26, remainder\ 1$ $26 \div 2 = 13, remainder\ 0$ $13 \div 2 = 6, remainder\ 1$ $6 \div 2 = 3, remainder\ 0$ $3 \div 2 = 1, remainder\ 1$ That is $53_{10}=110101_2$ $0.43 \times 2 = 0.86$ $0.86 \times 2 = 1.72$ $0.72 \times 2 = 1.44$ $0.44 \times 2 = 0.88$ That is $0.43_{10}=.0110_2$ $53.43_{10}=110101.0110_2$	3.5 each	7	

<table>
<tr><td>V</td><td>
<table>
<tr><th>Logical Gates</th><th>Symbol</th><th colspan="3">Truth Table</th></tr>
<tr><td rowspan="5">AND</td><td rowspan="5"></td><td>A</td><td>B</td><td>AB</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td rowspan="5">OR</td><td rowspan="5"></td><td>A</td><td>B</td><td>A + B</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td rowspan="3">NOT</td><td rowspan="3"></td><td>A</td><td>$\overline{A}$</td><td></td></tr>
<tr><td>0</td><td>1</td><td></td></tr>
<tr><td>1</td><td>0</td><td></td></tr>
<tr><td rowspan="5">NAND</td><td rowspan="5"></td><td>A</td><td>B</td><td>$\overline{A B}$</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
<tr><td rowspan="5">NOR</td><td rowspan="5"></td><td>A</td><td>B</td><td>$\overline{A + B}$</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
<tr><td rowspan="5">XOR</td><td rowspan="5"></td><td>A</td><td>B</td><td>$A \oplus B$</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
<tr><td rowspan="5">XNOR</td><td rowspan="5"></td><td>A</td><td>B</td><td>$\overline{A \oplus B}$</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td><td>7</td><td>7</td><td></td></tr>
<tr><td>VI</td><td>
<b>Types of code- 3.5 marks</b>
<b>Explanation-3.5 marks</b>


➢ An alphanumeric code is a system that represents both letters and numbers using a combination of characters; for instance, the ASCII code assigns unique numeric values to letters, digits, and symbols to facilitate digital text and data representation.
</td><td>3.5</td><td>7</td><td></td></tr>
</table>

<table>
<tr>
<td></td>
<td>

- ➢ A weighted numeric code is a binary coding scheme where each bit's position carries a specific weight, such as in the Binary-Coded Decimal (BCD) code, which represents decimal digits using groups of 4 bits each.
- ➢ A non-weighted numeric code is a binary code where each bit's position doesn't represent a specific weight; for example, the Gray Code changes only one bit between consecutive values, useful for minimizing errors in rotary encoders.
- ➢ A self-complementing numeric code is a binary code where the complement of a number is equal to its 9's or 10's complement; an example is the Excess-3 code where each decimal digit is represented by adding 3 to its value.
- ➢ A sequential numeric code is a binary code where consecutive numbers are encoded in an incremental sequence, such as the natural binary code where each successive number increments by one.
- ➢ An error detecting and correcting numeric code is a binary code designed to identify and rectify errors during data transmission or storage, such as the Hamming Code that adds redundancy for the detection and correction of single-bit errors.
- ➢ A reflective numeric code, also known as a self-complementary code, is a binary code where a number and its binary reflection are equivalent, for instance, the Excess-3 code where each decimal digit is represented by adding 3 to its value, and its reflection is obtained by subtracting it from 9.
- ➢ A cyclic numeric code is a binary code that loops cyclically, with the last code connecting back to the first, such as the Gray Code where adjacent numbers differ by only one bit, minimizing errors in rotary encoders.

</td>
<td>3.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VII</td>
<td>
<table>
<tr><th>A\B
C</th><th>00</th><th>01</th><th>11</th><th>10</th></tr>
<tr><td><b>0</b></td><td></td><td>1</td><td>1</td><td>1</td></tr>
<tr><td><b>1</b></td><td></td><td></td><td>1</td><td>1</td></tr>
</table>

$f = \bar{A}\bar{C} + B$
</td>
<td>3.5
3.5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>VIII</td>
<td>
<table>
<tr><th>A\B
C</th><th>00</th><th>01</th><th>11</th><th>10</th></tr>
<tr><td><b>0</b></td><td>1</td><td>1</td><td>1</td><td></td></tr>
<tr><td><b>1</b></td><td></td><td>1</td><td>1</td><td>1</td></tr>
</table>
</td>
<td>3.5</td>
<td>7</td>
<td></td>
</tr>
</table>

	$f = \overline{A}\overline{B} + AB + C$	3.5		
IX	➢ **Commutative Law** Any binary operation which satisfies the following expression is referred to as a commutative operation. Commutative law states that changing the sequence of the variables does not have any effect on the output of a logic circuit. ➢ A. B = B. A ➢ A + B = B + A ➢ **Associative Law** It states that the order in which the logic operations are performed is irrelevant as their effect is the same. • (A. B). C = A . (B . C) • (A + B) + C = A + (B + C) ➢ **Distributive Law** Distributive law states the following conditions: • A. (B + C) = (A. B) + (A. C) • A + (B. C) = (A + B) . (A + C) ➢ **AND Law** These laws use the AND operation. Therefore, they are called AND laws. • A .0 = 0 • A . 1 = A • A. A = A • $A.\bar{A}$=0 ➢ **OR Law** These laws use the OR operation. Therefore, they are called OR laws. • A + 0 = A	1 each	7	

- A + 1 = 1
- A + A = A
- $A + \bar{A}=1$

➢ **Inversion Law**

In Boolean algebra, the inversion law states that double inversion of variable results in the original variable itself.

$$\bar{\bar{A}} = A$$

➢ **DE-MORGAN's theorem**

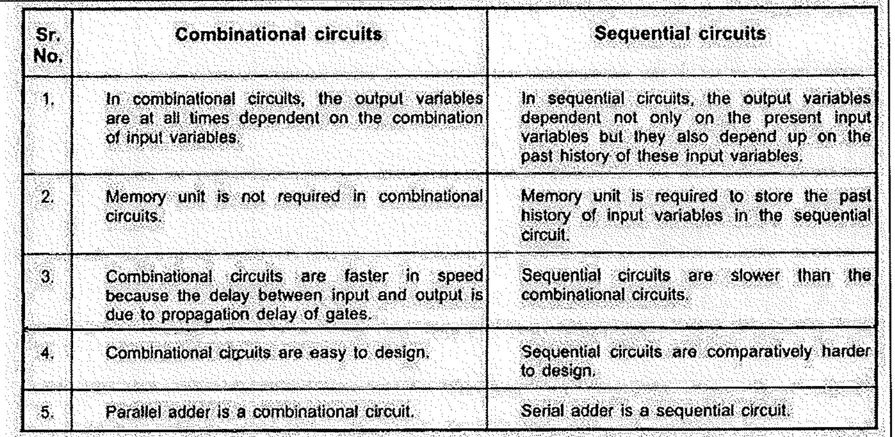
Theorem 1

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

Theorem 2

$\overline{A + B} = \overline{A}\,.\overline{B}$

X

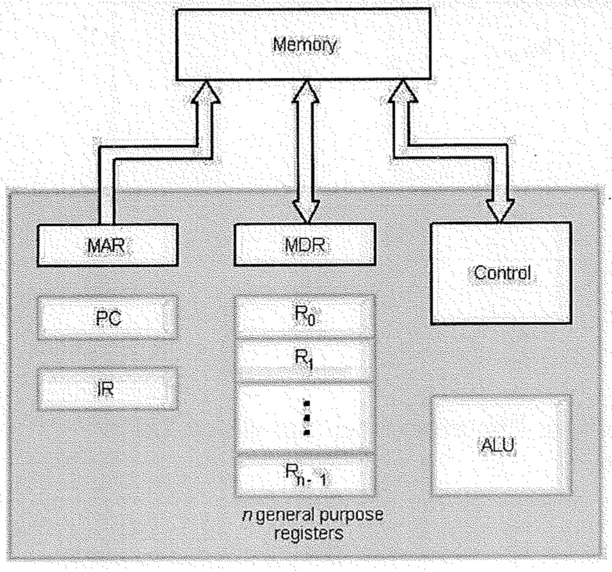


Sr. No.	Combinational circuits	Sequential circuits
1.	In combinational circuits, the output variables are at all times dependent on the combination of input variables.	In sequential circuits, the output variables dependent not only on the present input variables but they also depend up on the past history of these input variables.
2.	Memory unit is not required in combinational circuits.	Memory unit is required to store the past history of input variables in the sequential circuit.
3.	Combinational circuits are faster in speed because the delay between input and output is due to propagation delay of gates.	Sequential circuits are slower than the combinational circuits.
4.	Combinational circuits are easy to design.	Sequential circuits are comparatively harder to design.
5.	Parallel adder is a combinational circuit.	Serial adder is a sequential circuit.

7

7

XI

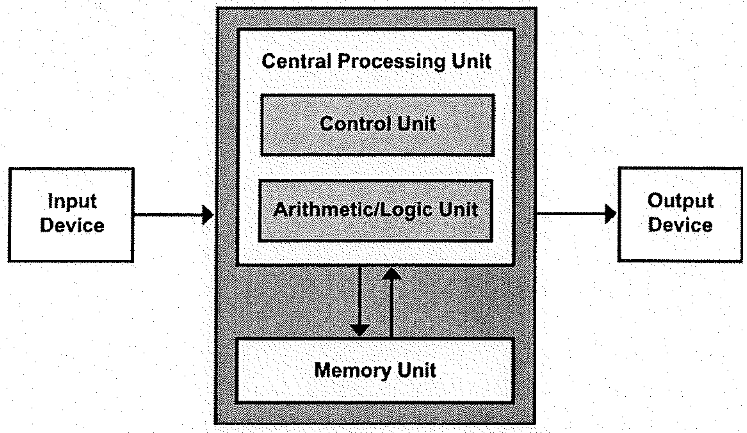


2

7

	Processor contains number of registers. ⬇ **Instruction register (IR)** holds instruction that is currently being executed. ⬇ **Program counter** (another register) keeps track of execution of program and points to the next instruction to be executed (Contains address of the next information to be executed). ⬇ **N-general purpose registers** ($R_0 - R_{n-1}$) also known as CPU registers. There are 2 registers communicate with memory. They are: 1. **MAR (Memory Address Register)** holds the address of location to be accessed. 2. **MDR (Memory Data Register)** contains data to be written into or read out of addressed location. OPERATION: Load, R0, LOC Add R4, R0 Store R4, LOC 1. IR holds the information about the instructions. 2. Processor register R0 loads the contents from the memory address 'LOC'. 3. Performs the addition between the contents in registers R4 and R0 with the help of ALU and the sum is available ate register R4. 4. The sum in the register R4 is moved to the memory location 'LOC'.	5		
XII	• Programmed I/O : In this mode the data transfer is initiated by the instructions written in a computer program. An input instruction is required to store the data from the device to the CPU and a store instruction is required to transfer the data from the CPU to the device. Data transfer through this mode requires constant monitoring of the peripheral	3.5 each	7	

<table>
<tr>
<td></td>
<td>device by the CPU and also monitor the possibility of new transfer once the transfer has been initiated. Thus CPU stays in a loop until the I/O device indicates that it is ready for data transfer. Thus programmed I/O is a time consuming process that keeps the processor busy needlessly and leads to wastage of the CPU cycles. This can be overcome by the use of an interrupt facility. This forms the basis for the Interrupt Initiated I/O.
• Interrupt Initiated I/O :
This mode uses an interrupt facility and special commands to inform the interface to issue the interrupt command when data becomes available and interface is ready for the data transfer. In the meantime CPU keeps on executing other tasks and need not check for the flag. When the flag is set, the interface is informed and an interrupt is initiated. This interrupt causes the CPU to deviate from what it is doing to respond to the I/O transfer. The CPU responds to the signal by storing the return address from the program counter (PC) into the memory stack and then branches to service that processes the I/O request. After the transfer is complete, CPU returns to the previous task it was executing. The branch address of the service can be chosen in two ways known as vectored and non-vectored interrupt. In vectored interrupt, the source that interrupts, supplies the branch information to the CPU while in case of non-vectored interrupt the branch address is assigned to a fixed location in memory.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XIII</td>
<td><b>Diagram-3.5</b>
<b>Explanation-3.5</b>
Time-Space diagram
<table>
<tr><th>CLOCK →
SEGMENT ↓</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th></tr>
<tr><td>1</td><td>T1</td><td>T2</td><td>T3</td><td>T4</td><td>T5</td><td>T6</td><td>T7</td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>T1</td><td>T2</td><td>T3</td><td>T4</td><td>T5</td><td>T6</td><td>T7</td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td>T1</td><td>T2</td><td>T3</td><td>T4</td><td>T5</td><td>T6</td><td>T7</td><td></td></tr>
<tr><td>4</td><td></td><td></td><td></td><td>T1</td><td>T2</td><td>T3</td><td>T4</td><td>T5</td><td>T6</td><td>T7</td></tr>
</table>
Here it is 4 stage/segment pipeline is shown. In each clock cycle only one instruction is processed</td>
<td>7</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>XIV</td>
<td><b>Figure-3.5 marks</b>
<b>Explanation-3.5 marks</b></td>
<td></td>
<td></td>
<td></td>
</tr>
</table>

	 Input Device Central Processing Unit Control Unit Arithmetic/Logic Unit Memory Unit Output Device	3.5	7	
	It is a design model for modern computers which has a Central Processing Unit (CPU) and the concept of Memory which is used for storing both data and instructions. This architecture implemented the stored program concept in which the data and instructions are stored in the same memory. This architecture consists of a Control Unit, CPU, Arithmetic and logic unit(ALU), Register, I/O(Input Output Devices), and Memory unit.	3.5		