		OR			
11	a)	Discuss daisy-chaining priority interrupt with a neat diagram.	L2	CO2	5 M
	b)	Describe the working of a DMA controller with neat diagram.	L2	CO2	5 M

PVP 23

Code: 23ES1304

II B.Tech - I Semester – Regular / Supplementary Examinations
NOVEMBER 2025

DIGITAL LOGIC AND COMPUTER ORGANIZATION

(Common for CSE, IT)

Duration: 3 hours Max. Marks: 70

Note: 1. This question paper contains two Parts A and B.
2. Part-A contains 10 short answer questions. Each Question carries 2 Marks.
3. Part-B contains 5 essay questions with an internal choice from each unit. Each Question carries 10 marks.
4. All parts of Question paper must be answered in one place.

BL – Blooms Level CO – Course Outcome

PART – A

		BL	CO
1.a)	Represent $(-13)_{10}$ in 8-bit 2's complement form.	L2	CO1
1.b)	Prove the theorem $X\ .\ X = X$	L3	CO3
1.c)	Describe four-bit half-adder circuit.	L2	CO1
1.d)	Draw SR latch with control input logic diagram and truth table.	L2	CO4
1.e)	What is a stack pointer?	L2	CO2
1.f)	What happens when sum of two decimal numbers in BCD exceed 9?	L1	CO1
1.g)	Define the term "volatile memory."	L1	CO1
1.h)	What is the role of page table in virtual memory?	L1	CO1
1.i)	Differentiate between input and output devices.	L2	CO2
1.j)	What is a priority interrupt?	L1	CO2

PART – B

			BL	CO	Max. Marks
		UNIT-I			
2	a)	What is the use of Complements? Explain 9's and 10's Complements with examples.	L2	CO1	5 M
	b)	Explain boolean algebra theorems with proofs.	L2	CO3	5 M
		OR			
3	a)	Describe the signed binary numbers in digital design.	L2	CO1	5 M
	b)	What is meant by Sum of Minterms? Express the Boolean function $F = A + \bar{B}C$ in Sum of Minterms.	L3	CO3	5 M
		UNIT-II			
4	a)	What are the steps to be followed obtain the output Boolean functions of a combinational circuit from its logic diagram?	L2	CO1	5 M
	b)	Describe the universal shift register.	L2	CO4	5 M
		OR			
5	a)	What is the use of Decoder in digital system? Construct 3X8 decoder using its truth table.	L3	CO1	5 M
	b)	Explain the working of a 4-bit asynchronous ripple counter using T flip-flops with timing diagrams.	L2	CO4	5 M

		UNIT-III			
6	a)	Explain memory stack with suitable example.	L2	CO2	5 M
	b)	Explain Booth's multiplication algorithm with step-by-step example.	L3	CO1	5 M
		OR			
7	a)	Explain addressing modes in brief.	L2	CO2	5 M
	b)	Discuss decimal subtraction using 9's and 10's complements with examples.	L2	CO1	5 M
		UNIT-IV			
8	a)	Describe the read and write operations of associative memory.	L2	CO1	5 M
	b)	Explain the concept of virtual memory and its need in modern systems.	L2	CO1	5 M
		OR			
9	a)	Explain the need for auxiliary memory in computer systems.	L2	CO1	5 M
	b)	Explain the working of write-through and write-back policies in cache memory.	L2	CO1	5 M
		UNIT-V			
10	a)	Describe the functional block diagram of an I/O module and explain its operation.	L2	CO2	5 M
	b)	Explain asynchronous data transfer with a neat diagram.	L2	CO2	5 M

II. B.Tech- I Semester- Regular/ Supplementary Examinations

NOVEMBER 2025

DIGITAL LOGIC AND COMPUTER ORGANIZATION

(Common for CSE, IT)

Schema

PART A

1.a) Represent (-13)(10) in 8-bit 2's complement form. **10*2=20M**

1's complement -1M

2's complement-1M

1.b) Prove the theorem X.X = X

Proof in truth table-2M

1.c) Describe four-bit half-adder circuit.

Half adder truth table- 1M

K map simplification and Diagarm-1M

1.d) Draw SR latch with control input logic diagram and truth table.

SR Latch diagram-1 M

Truth Table-1M

1.e) What is a stack pointer?

Stack Pointer explanation-2M

1.f) What happens when sum of two decimal numbers in BCD exceed 9?

Explanation with example-2M

1.g) Define the term "volatile memory."

Explanation-2M

1.h) What is the role of page table in virtual memory?

Explanation-2M

1.i) Differentiate between input and output devices.

Any two differences-2M

1.j) What is a priority interrupt?

Explanation -2M

PART B

UNIT-I

2.a) What is the use of Complements? Explain 9's and 10's Complements with examples.

Use of complements-1M **5M**

9's complement example -2M

10's complement example-2M

2.b) Explain boolean algebra theorems with proofs. **5M**

Any two proofs 2 x 2.5=5M

OR

3.a) Describe the signed binary numbers in digital design. **5M**

Explanation -4M
Example -1M

3.b) What is meant by Sum of Minterms? Express the Boolean function $F = A + \overline{B}\,C$ in Sum of Minterms. **5M**

Definition of Minterms-1M
Problem Solving-4M

UNIT-II

4.a) What are the steps to be followed obtain the output Boolean functions of a combinational circuit from its logic diagram? **5M**

Explanation-3M
Example-2M

4.b) Describe the universal shift register. **5M**

Diagram-3M
Explanation-2M

OR

5.a) What is the use of Decoder in digital system? Construct 3X8 decoder using its truth table.

Use of decoder- 1M
Truth Table-2M
Diagarm-2M

5M

5.b) Explain the working of a 4-bit asynchronous ripple counter using T flip-flops with timing diagrams. **5M**

Diagram with explanation-3M
Timing Diagram-2M

UNIT-III

6.a) Explain memory stack with suitable example. **5M**

Memory stack explanation with diagram-5M

6.b) Explain Booth's multiplication algorithm with step-by-step example. **5M**

Algorithm- 3M
Example -2M

OR

7.a) Explain addressing modes in brief. **5M**

Any five addressing modes with example-5M

7.b) Discuss decimal subtraction using 9's and 10's complements with examples. 5M

9's complement example-2.5M
10's complement example-2.5M

UNIT-IV

8.a) Describe the read and write operations of associative memory. **5M**

Read operation-2.5M
Write operation-2.5M

8.b) Explain the concept of virtual memory and its need in modern systems. **5M**

Virtual Memory explanation-4M
Need of Virtual memory-1M

OR

9.a) Explain the need for auxiliary memory in computer systems. 5M

Auxiliary memory explanation-5M

9.b) Explain the working of write-through and write-back policies in cache memory. 5M

Write –through- 2.5M
Write-back- 2.5M

UNIT-V

10.a) Describe the functional block diagram of an I/O module and explain its operation.

I/O interface diagram-2M 5M
Explanation-3M

10.b) Explain asynchronous data transfer with a neat diagram. 5M

Strobe-2.5M
Handshaking-2.5M

OR

11.a) Discuss daisy-chaining priority interrupt with a neat diagram. **5M**

Diagarm-2M
Explanation-3M

11.b) Describe the working of a DMA controller with neat diagram. **5M**

DMA controller -2M

Explanation-3M

1.a) Represent $(-13)_{10}$ in 8-bit 2's complement form.

Sol: To represent the $(-13)_{10}$ in 2's complement form first take the $(+13)_{10}$

$(+13)_{10}$= 00001101

$(+13)_{10}$ in 1's Complement is = 11110010

To get the (-13) in 2's complement add 1 to the 1's complement of $(+13)_{10}$

So $(-13)_{10}$ 8 bit 2's complement for is 11110011

1.b) Prove the theorem X.X = X

Sol: From the boolean algebra

When X=0, X.X=0.0=0=X

When X=1, X.X=1.1=1=X

OR

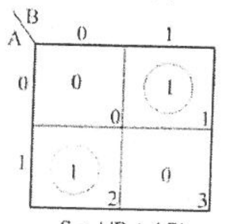
X	X.X	Result
0	0.0	0=X
1	1.1	1=X

1.c) Describe four-bit half-adder circuit.

Sol: Half adder is used for addition of 2 bits. When you add two bits we will get Sum and Carry. The following table shows the Half adder results

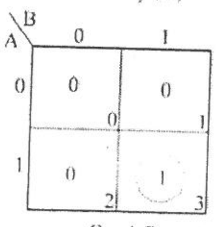
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

For Sum (S)



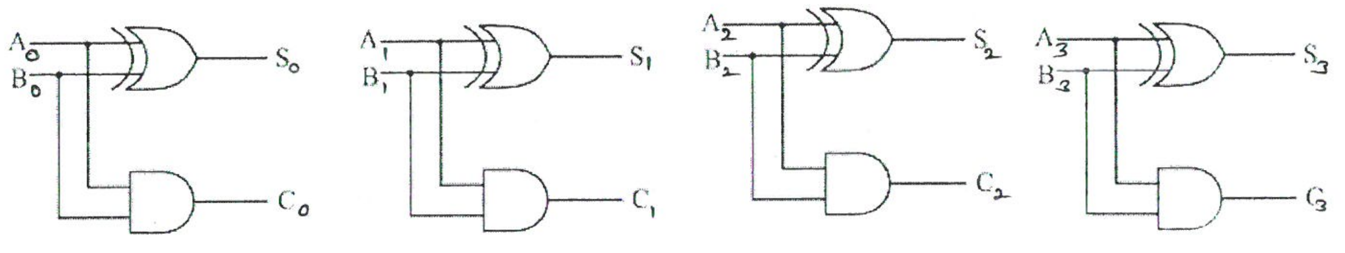
S = A'B + AB'

For Carry (C)



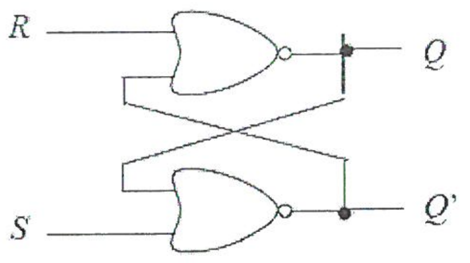
C = A B

S=A'B+AB' = $A \oplus B$



1.d) Draw SR latch with control input logic diagram and truth table.

Sol: The *SR* latch can be implemented by NOR gates or NAND gates. SR Latch with NOR gate is shown below



S	*R*	*Q*	*Q'*	
1	0	1	0	(Set state)
0	0	1	0	(After *S*=1, *R*=0)
0	1	0	1	(Reset state)
0	0	0	1	(After *S*=0, *R*=1)
1	1	0	0	(Undefined)

1.e) What is a stack pointer?

Sol: A stack pointer (SP) is a special-purpose CPU register that holds the memory address of the top element of the stack.

The stack is a section of memory used for temporary data storage, structured as a Last-In, First-Out (LIFO) data structure. The stack pointer is critical for managing this area.

- Tracking the Top: The SP is the only way the CPU knows where the stack currently ends (or begins, depending on the architecture).

PUSH Operation: When data is pushed onto the stack (added), the SP is decremented (or incremented, depending on whether the stack grows up or down in memory) to point to the new location where the data is stored.

POP Operation: When data is popped off the stack (removed), the data is retrieved from the address currently held by the SP, and then the SP is incremented (or decremented) to point to the new top of the stack.

1.f) What happens when sum of two decimal numbers in BCD exceed 9?

Sol: When the sum of two BCD digits exceeds 9, the result becomes an invalid BCD code. To correct this, the value **0110_2 (decimal 6)** is added to the binary sum. This adjustment converts the invalid result into a valid BCD representation and generates a carry if needed, ensuring the final output remains a proper BCD digit.

Example:

Add 8+7 in BCD

8 in BCD = 1000

7 in BCD =0111.

The binary sum is 1111_2, which is greater than 9, so it is not valid BCD. To correct it, add 0110_2 (6):
$1111_2 + 0110_2 = 1\ 0101_2$.
Thus, the BCD result is 0001 0101, which represents 15 correctly.

1.g) Define the term "volatile memory."

Sol : RAM (Random Access Memory) is a fast, temporary storage used by a computer to hold data and instructions that the CPU needs while programs are running. It allows quick read and write operations, enabling smooth multitasking and faster program execution. However, RAM is volatile, meaning all stored information is lost when the power is turned off. RAM size and speed directly affect overall system performance, making it a crucial component in modern computing

1.h) What is the role of page table in virtual memory?

Sol: A page table's primary role in virtual memory is to translate virtual (or logical) addresses into physical addresses in RAM. It acts as a data structure maintained by the operating system to map the memory addresses

1.i) Differentiate between input and output devices.

Sol:

Input Devices	Output devices
It accepts data from user.	It reflects processed data to user.
Takes the data from the user and sends it to the processor for execution.	It takes the processed data from the processor and sends it back to the user.
It helps the computer is accepting the data.	It helps the computer is displaying the data.
Ex: Keyboard, Image Scanner, Microphone, Pointing device.	Ex: Monitor, Printers

1.j) What is a priority interrupt?

Sol: A priority interrupt is a system that establishes a priority over the various sources to determine which condition is to be serviced first when two or more requests arrive simultaneously. The system may also determine which conditions are permitted to interrupt the computer while another interrupt is being serviced. Higher-priority interrupt levels are assigned to requests which, if delayed or interrupted, could have serious consequences.

UNIT-I

2.a) What is the use of Complements? Explain 9's and 10's Complements with examples.

Sol: Complements are mainly used for representing the negative numbers.
When the number is negative, the sign is represented by 1 but the rest of the number may be represented in one of three possible ways:

1. Signed-magnitude representation
2. Signed-1' s complement representation
3. Signed 2' s complement representation

There are two types of complements for each base r system: the r's complement and the (r -l)'s complement. When the value of the base r is substituted in the name, the two types are referred to as the 2's and 1's complement for binary numbers and the 10's and 9's complement for decimal numbers.

<u>9's Complement</u>
Given a number N in base r having n digits, the (r - 1)'s complement of N is defined as
$(r^n - I) - N$.

For decimal numbers r = 10 and r - 1 = 9, so the 9's complement of N is
$(10^n - 1) - N$.

Now, 10^n represents a number that consists of a single 1 followed by n 0's. $10^n - 1$ is a number represented by n 9's

For example, with $n = 4$ we have $10^4 = 10000$ and $10^4 - 1 = 9999$. It follows that the 9' s complement of a decimal number is obtained by subtracting each digit from 9. For example, the 9's complement of 546700 is 999999- 546700 = 453299.

10's Complement:

The r's complement of an n-digit number N in base r is defined as $r^n - N$.
r's complement is obtained by adding 1 to the (r - 1)'s complement.

Example: Thus the 10's complement of the decimal 2389 is 7610 + I = 7611 and is obtained by adding 1 to the 9' s complement value.

2.b) Explain Boolean algebra theorems with proofs.

Sol: The following are the different types of Boolean algebra theorems and postulates

Postulates and Theorems of Boolean Algebra

Postulate 2	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
Postulate 5	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Theorem 1	(a)	$x + x = x$	(b)	$x \cdot x = x$
Theorem 2	(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$
Theorem 3, involution		$(x')' = x$		
Postulate 3, commutative	(a)	$x + y = y + x$	(b)	$xy = yx$
Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$	(b)	$x(yz) = (xy)z$
Postulate 4, distributive	(a)	$x(y+z) = xy + yz$	(b)	$x + yz = (x+y)(x+z)$
Theorem 5, DeMorgan	(a)	$(x+y)' = x'y'$	(b)	$(xy)' = x' + y'$
Theorem 6, absorption	(a)	$x + xy = x$	(b)	$x(x+y) = x$

THEOREM 1(a): $x + x = x$.

Statement	**Justification**
$x + x = (x + x) \cdot 1$	postulate 2(b)
$= (x + x)(x + x')$	5(a)
$= x + xx'$	4(b)
$= x + 0$	5(b)
$= x$	2(a)

THEOREM 1(b): $x \cdot x = x$.

Statement	**Justification**
$x \cdot x = xx + 0$	postulate 2(a)
$= xx + xx'$	5(b)
$= x(x + x')$	4(a)
$= x \cdot 1$	5(a)
$= x$	2(b)

THEOREM 2(a): $x + 1 = 1$.

Statement	**Justification**
$x + 1 = 1 \cdot (x + 1)$	postulate 2(b)
$= (x + x')(x + 1)$	5(a)
$= x + x' \cdot 1$	4(b)
$= x + x'$	2(b)
$= 1$	5(a)

THEOREM 2(b): $x \cdot 0 = 0$ by duality.

NOTE: Proof of any two theorem marks will be awarded

OR

3.a) Describe the signed binary numbers in digital design.

Sol:

- Positive integers (including zero) can be represented as unsigned numbers. However, to represent negative integers, we need a notation for negative values. The convention is to make the sign bit 0 for positive and 1 for negative.
- If the binary number is signed, then the leftmost bit represents the sign and the rest of the bits represent the number.

- If the binary number is assumed to be unsigned, then the leftmost bit is the most significant bit of the number
- For example, the string of bits 01001 can be considered as 9 (unsigned binary) or as +9 (signed binary) because the leftmost bit is 0. The string of bits 11001 represents the binary equivalent of 25 when considered as an unsigned number and the binary equivalent of -9 when considered as a signed number. This is because the 1 that is in the leftmost position designates a negative and the other four bits represent binary 9.

- The representation of the signed numbers in the last example is referred to as the signed-magnitude convention
- When arithmetic operations are implemented in a computer, it is more convenient to use a different system, referred to as the signed complement system, for representing negative numbers. In this system, a negative number is indicated by its complement. The signed-complement system can use either the 1's or the 2's complement, but the 2's complement is the most common

Negative Nunbers can be represented in Three ways

- signed-magnitude representation
- signed-1's-complement representation
- signed-2's-complement representation

As an example, consider the number 9, represented in binary with eight bits. +9 is represented with a sign bit of 0 in the leftmost position, followed by the binary equivalent of 9, which gives 00001001

There are three different ways to represent -9 with eight bits:

signed-magnitude representation: 10001001
signed-1's-complement representation: 11110110
signed-2's-complement representation: 11110111

3.b) What is meant by Sum of Minterms? Express the Boolean function $F = A + \overline{BC}$in Sum of Minterms.
Sol:

A binary variable may appear either in its normal form (x) or in its complement form (x'). Now consider two binary variables x and y combined with an AND operation. Since each variable may appear in either form, there are four possible combinations: xy , xy, xy , and xy. Each of these four AND terms is called a minterm or a or a standard product.

A Boolean function can be expressed algebraically from a given truth table by form ing a minterm for each combination of the variables that produces a 1 in the function and then taking the OR of all those terms is called sum of minterms. For example, the function f1

$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$

Minterms for Three Binary Variables

			Minterms	
x	**y**	**z**	**Term**	**Designation**
0	0	0	$x'y'z'$	m_0
0	0	1	$x'y'z$	m_1
0	1	0	$x'yz'$	m_2
0	1	1	$x'yz$	m_3
1	0	0	$xy'z'$	m_4
1	0	1	$xy'z$	m_5
1	1	0	xyz'	m_6
1	1	1	xyz	m_7

Consider the function F= A+B'C. The function has three variables A, B,C. The first term A is missing two variables; therefore,

$$A = A(B + B') = AB + AB'$$

This function is still missing one variable, so

$$\begin{aligned} A &= AB(C + C') + AB'(C + C') \\ &= ABC + ABC' + AB'C + AB'C' \end{aligned}$$

The second term $B'C$ is missing one variable; hence,

$$B'C = B'C(A + A') = AB'C + A'B'C$$

Combining all terms, we have

$$\begin{aligned} F &= A + B'C \\ &= ABC + ABC' + AB'C + AB'C' + A'B'C \end{aligned}$$

But $AB'C$ appears twice, and according to theorem 1 ($x + x = x$), it is possible to remove one of those occurrences. Rearranging the minterms in ascending order, we finally obtain

$$\begin{aligned} F &= A'B'C + AB'C' + AB'C + ABC' + ABC \\ &= m_1 + m_4 + m_5 + m_6 + m_7 \end{aligned}$$

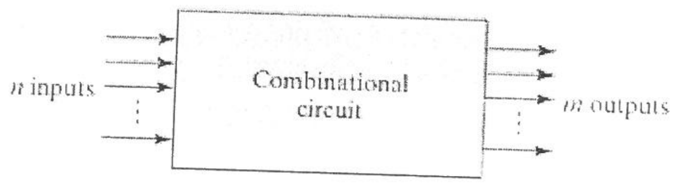
When a Boolean function is in its sum-of-minterms form, it is sometimes convenient to express the function in the following brief notation:

$$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$$

UNIT-II

4.a) What are the steps to be followed obtain the output Boolean functions of a combinational circuit from its logic diagram?

Sol: A combinational circuit consists of an interconnection of logic gates. Combinational logic gates react to the values of the signals at their inputs and produce the value of the output signal, transforming binary information from the given input data to a required output data. A block diagram of a combinational circuit is shown in Fig.



To obtain the output Boolean functions from a logic diagram, we proceed as follows:

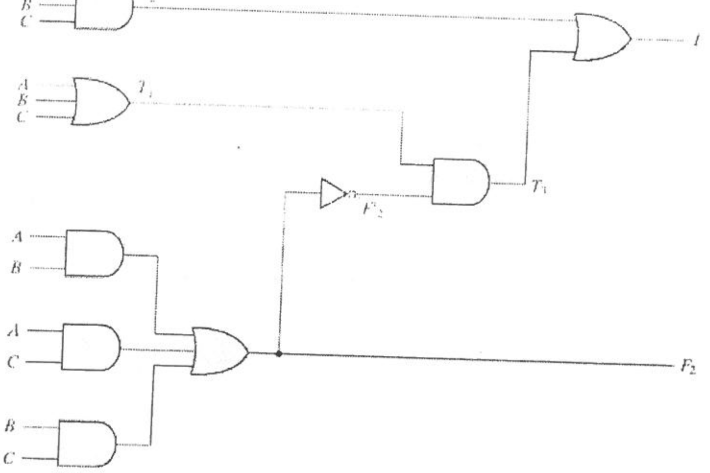
1. Label all gate outputs that are a function of input variables with arbitrary symbols— but with meaningful names. Determine the Boolean functions for each gate output.

2. Label the gates that are a function of input variables and previously labeled gates with other arbitrary symbols. Find the Boolean functions for these gates.

3. Repeat the process outlined in step 2 until the outputs of the circuit are obtained.

4. By repeated substitution of previously defined functions, obtain the output Boolean functions in terms of input variables.

Example :



We note that the circuit has three binary inputs— A , B , and C —and two binary outputs— F1 and F2. The outputs of various gates are labeled with intermediate symbols. The outputs of gates that are a function only of input variables are T1 and T2. Output F2 can easily be derived from the input variables. The Boolean functions for these three outputs are

F2 = AB + AC + BC

T1 = A + B + C

T2 = ABC

Next, we consider outputs of gates that are a function of already defined symbols:

T3 = F2'T1

F1 = T3 + T2

To obtain F1 as a function of A , B and C , we form a series of substitutions as follows:

F1 = T3 + T2 = F2'T1 + ABC = (AB + AC + BC)' (A + B + C) + ABC

= (A' + B')(A' + C')(B' + C')(A + B + C) + ABC

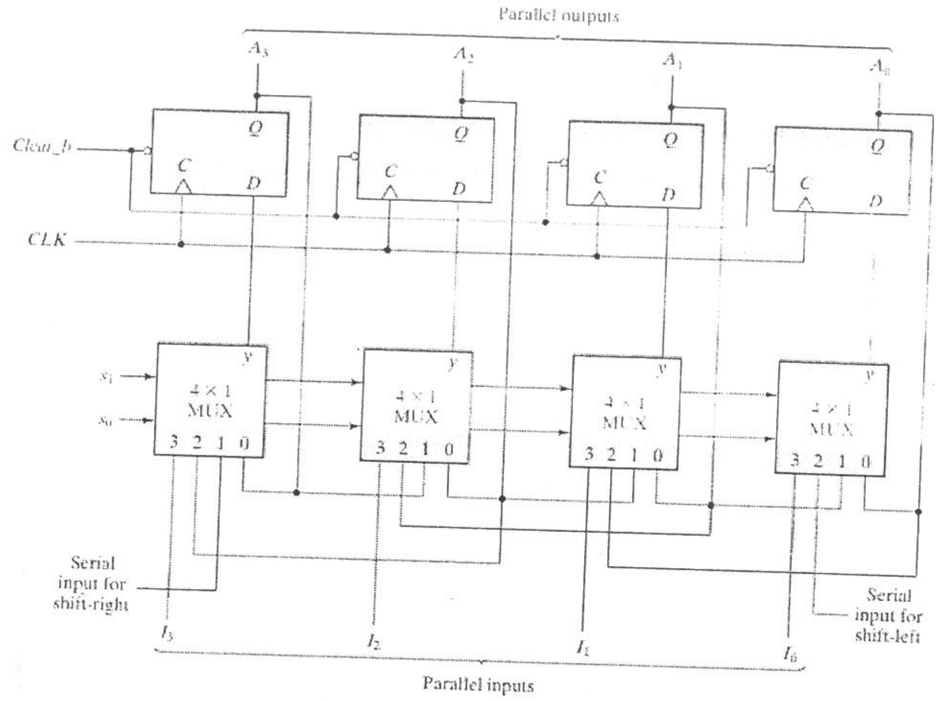
= (A' + B'C')(AB' + AC' + BC' + B'C) + ABC

= A'BC' + A'B'C + AB'C' + ABC

4.b) Describe the universal shift register.

Sol:

If the register has both shifts and parallel-load capabilities, it is referred to as a universal shift register. The following Diagram shows the universal shift registers



The circuit consists of four D flip-flops and four multiplexers. The four multiplexers have two common selection inputs s1 and s0. Input 0 in each multiplexer is selected when s1s0 = 00. input 1 is selected when s1s0 = 01, and similarly for the other two inputs. The selection inputs control the mode of operation of the register according to the function entries in the given table

Mode Control		
s_1	s_0	**Register Operation**
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

When s1s0 = 00, the present value of the register is applied to the D inputs of the flip-flops. This condition forms a path from the output of each flip-flop into the input of the same flip-flop, so that the output recirculates to the input in this mode of operation The next clock edge transfers into each flip-flop the binary value it held previously, and no change of state occurs. When s1s0 = 01, terminal 1 of the multiplexer inputs has a path to the D inputs of the flip-flops. This causes a shift-right operation, with the serial input transferred into flip-flop A3. When s1s0 = 10, a shift-left operation results, with the other serial input going into flip-flop A0. Finally, when s1s0 = 11, the binary information on the parallel input lines is transferred into the register simultaneously during the next clock edge.

OR

5.a) What is the use of Decoder in digital system? Construct 3X8 decoder using its truth table.

Sol: A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. Decoders are used to convert coded input into a readable output, performing functions like memory addressing, data demultiplexing, seven-segment displays, and code conversion

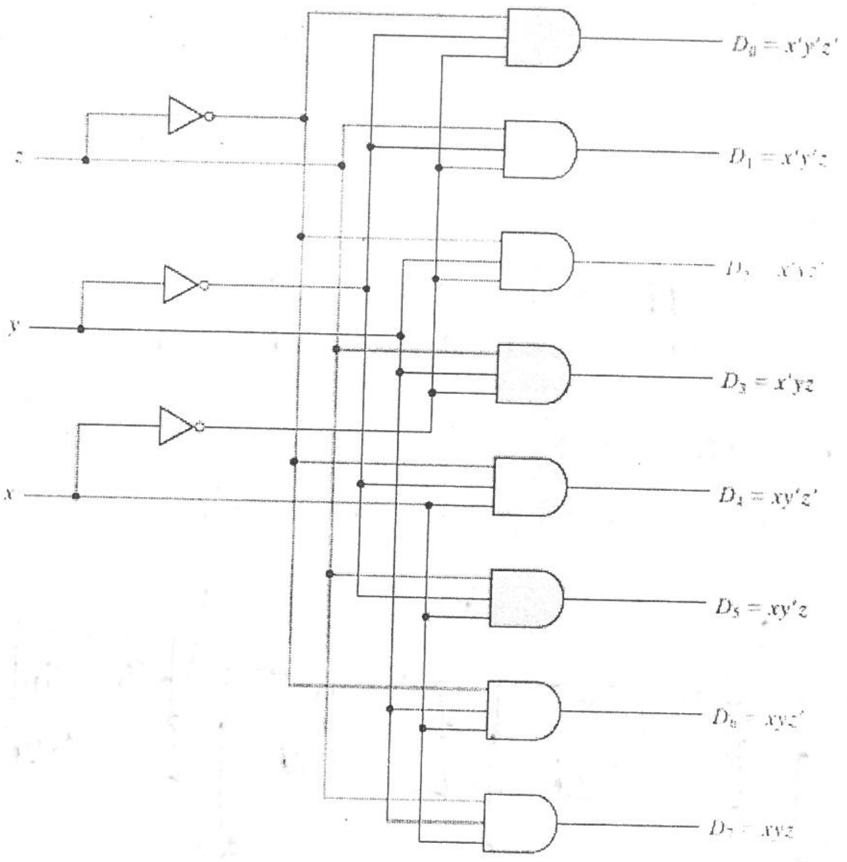
3 X 8 line Decoder

The three inputs are decoded into eight outputs, each representing one of the minterms of the three input variables. The three inverters provide the complement of the inputs, and each one of the eight AND gates generates one of the minterms. A particular application of this decoder is binary-to-octal conversion. The input variables represent a binary number, and the outputs represent the eight digits of a number in the octal number system. However, a three-to-eight-line decoder can be used for decoding any three-bit code to provide eight outputs, one for each element of the code. The following table shows the 3 x 8 line Decoder

Truth Table of a Three-to-Eight-Line Decoder

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

The following diagram shows the 3 x8 line decoder



z
y
x
$D_0 = x'y'z'$
$D_1 = x'y'z$
$D_2 = x'yz'$
$D_3 = x'yz$
$D_4 = xy'z'$
$D_5 = xy'z$
$D_6 = xyz'$
$D_7 = xyz$

5.b) Explain the working of a 4-bit asynchronous ripple counter using T flip-flops with timing diagrams.

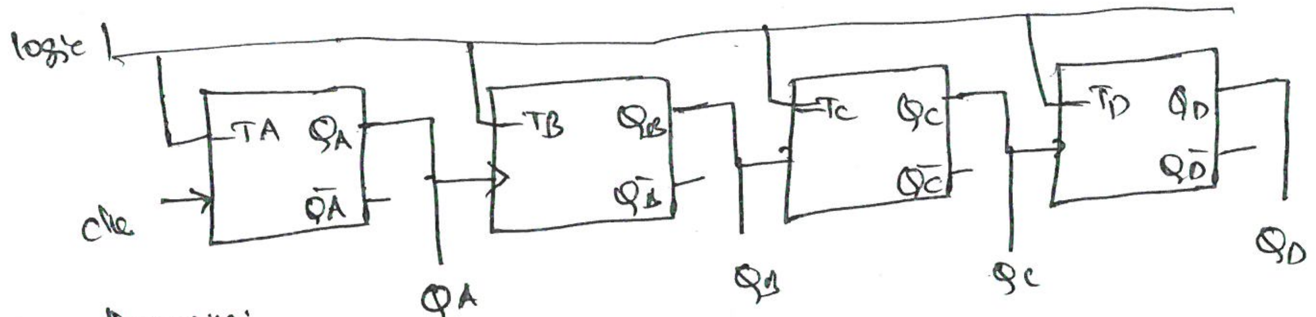
Sol: No.of flip flops required $2^n \geq N$.

where N = no.of states = 16 ie 0000 to 1111

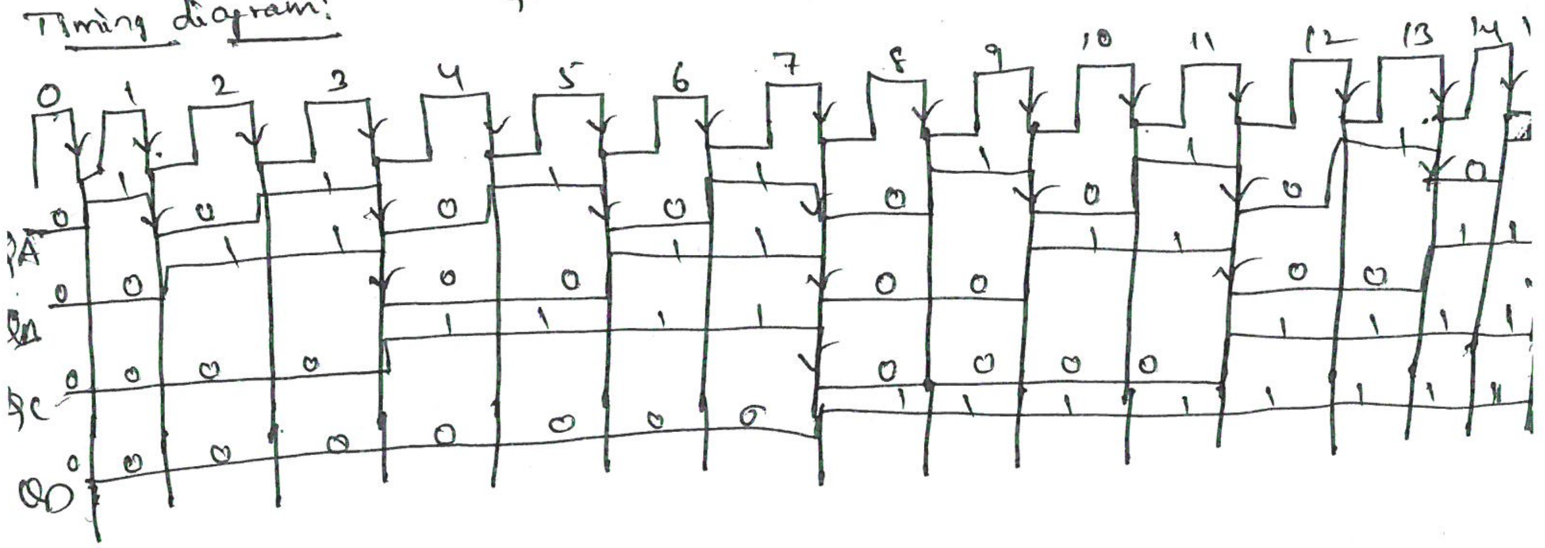
n = no.of flipflops. Required

$2^n \geq 16 \Rightarrow 2^4 \geq 16$

$\therefore$ 4-T flipflops required



Timing diagram:

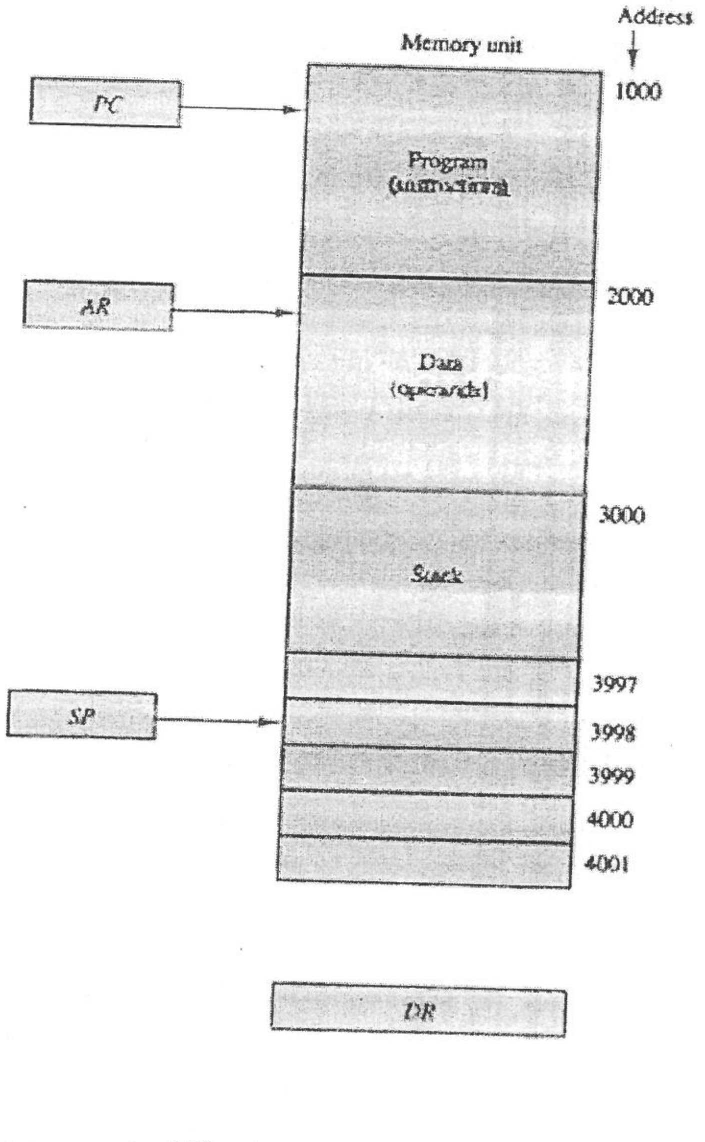


S.No	Q_A	Q_B	Q_C	Q_D	output
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3.
⋮	⋮	⋮	⋮	⋮	
15	1	1	1	1	15

UNIT-III

6.a) Explain memory stack with suitable example.

Sol: A stack can exist as a stand-alone unit that can be implemented in a random-access memory attached to a CPU. The implementation of a stack in the CPU is done by assigning a portion of memory to a stack operation and using a processor register as a stack pointer. The following diagram shows the a portion of computer memory partitioned into three segments: program, data, and stack.



The program counter PC points at the address of the next instruction in the program. The address register AR points at an array of data. The stack pointer SP points at the top of the stack. The three registers are connected to a common address bus, and either one can provide an address for memory. PC is used during the fetch phase to read an instruction. AR is used during the execute phase to read an operand. SP is used to push or pop items into or from the stack. the initial value of SP is 4001 and the stack grows with decreasing addresses. Thus the first item stored in the stack is at address 4 000, the second item is stored at address 3999, and the last address that can be used for the stack is 3000. No provisions are available for stack limit checks.

We assume that the items in the stack communicate with a data register DR. A new item is inserted with the push operation as follows:

$SP \leftarrow SP - 1$
$M[SP] \leftarrow DR$

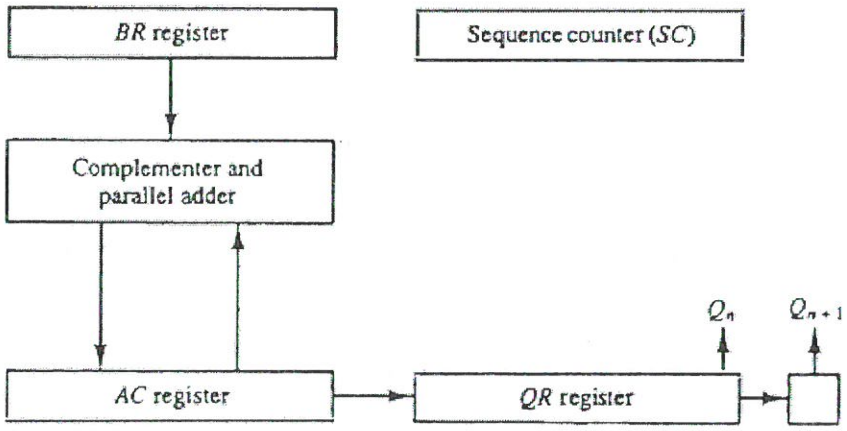
The stack pointer is decremented so that it points at the address of the next word. A memory write operation inserts the word from DR into the top of the stack. A new item is deleted with a pop operation as follows:

$DR \leftarrow M[SP]$
$SP \leftarrow SP + 1$

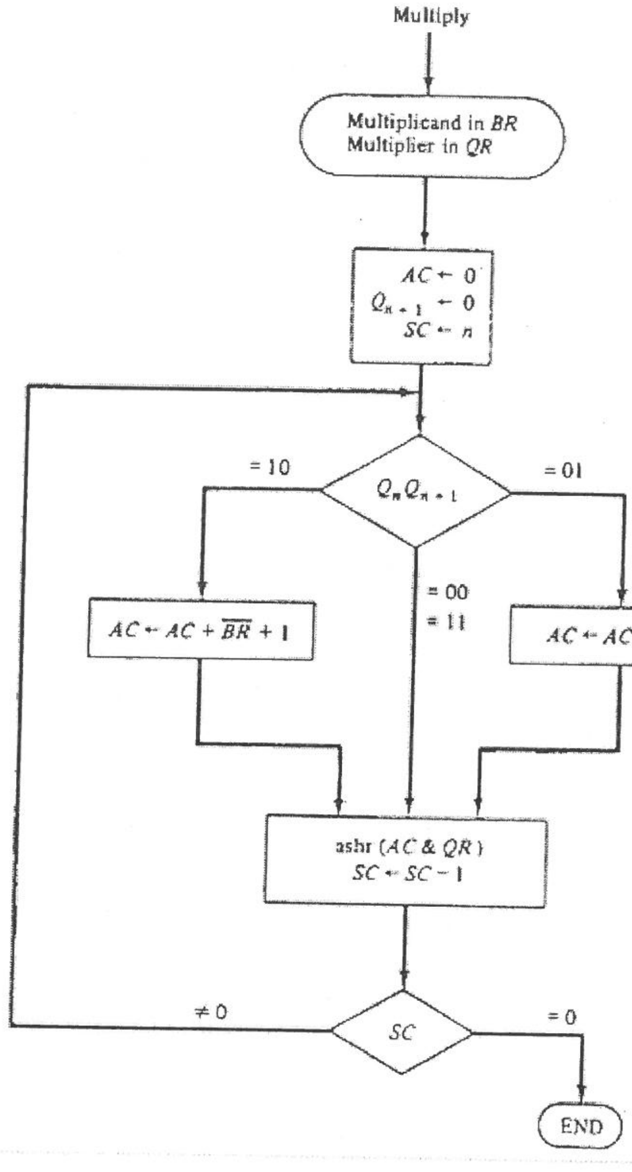
6.b) Explain Booth's multiplication algorithm with step-by-step example.

Sol: Booth algorithm gives a procedure for multiplying binary integers in signed-2's complement representation. It operates on the fact that strings of O's in the multiplier require no addition but just shifting, and a string of 1's in the multiplier. The hardware implementation of Booth algorithm requires the register configuration shown in the following diagram

Hardware for Booth algorithm.



We rename registers A, B, and Q, as AC, BR, and QR, respectively. Qn, designates the least significant bit of the multiplier in register QR. An extra flip-flop Q_{n+1} is appended to QR to facilitate a double bit inspection of the multiplier. The flowchart for Booth algorithm is shown as follows



Bit Q_{n+1} are initially cleared to 0 and the sequence counter SC is set to a number n equal to the number of bits in the multiplier. The two bits of the multiplier in Q_n and Qn+1 are inspected. If the two bits are equal to 10, it means that the first 1 in a string of 1' s has been encountered. This requires a subtraction of the multiplicand from the partial product in AC. If the two bits are equal to 01, it means that the first 0 in a string of 0' s has been encountered. This requires the addition of the multiplicand to the partial product in AC. When the two bits are equal, the partial product does not change. An overflow cannot occur because the addition and subtraction of the multiplicand follow each other. As a consequence, the two numbers that are added always have opposite signs, a condition that excludes an overflow. The next step is to shift right the partial product and the multiplier (including bit Qn+1). This is an arithmetic shift right (ashr) operation which shifts AC and QR to the right and leaves the sign bit in AC unchanged. The sequence counter is decremented and the computational loop is repeated n times

A numerical example of Booth algorithm for n = 5. It shows the step-by-step multiplication of (-9) x (-13) = + 117

Example of Multiplication with Booth Algorithm

Q_n Q_{n+1}	BR = 10111 $\overline{BR}$ + 1 = 01001	AC	QR	Q_{n+1}	SC
	Initial	00000	10011	0	101
1 0	Subtract BR	01001			
		01001			
	ashr	00100	11001	1	100
1 1	ashr	00010	01100	1	011
0 1	Add BR	10111			
		11001			
	ashr	11100	10110	0	010
0 0	ashr	11110	01011	0	001
1 0	Subtract BR	01001			
		00111			
	ashr	00011	10101	1	000

OR

7a) Explain addressing modes in brief.

Sol: An addressing mode in computer organization defines how the operand (the data) required for an instruction is specified or determined during program execution.

The following are the different types of Addressing Modes

Implied Mode: In this mode the operands are specified implicitly in the definition of the instruction.

Immediate Mode: In this mode the operand is specified in the instruction itself. In other words, an immediate-mode instruction has an operand field rather than an address field.

Register Mode: In this mode the operands are in registers that reside within the CPU. The particular register is selected from a register field in the instruct ion.

Register Indirect Mode: In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory. In other words, the selected register contains the address of the operand rather than the operand itself.

Autoincrement or Autodecrement Mode:

This is similar to the register indirect mode except that the register is incremented or decremented after (or before) its value is used to access memory.

Direct Address Mode: In this mode the effective address is equal to the address part of the instruction. The operand resides in memory and its address is given directly by the address field of the instruction.

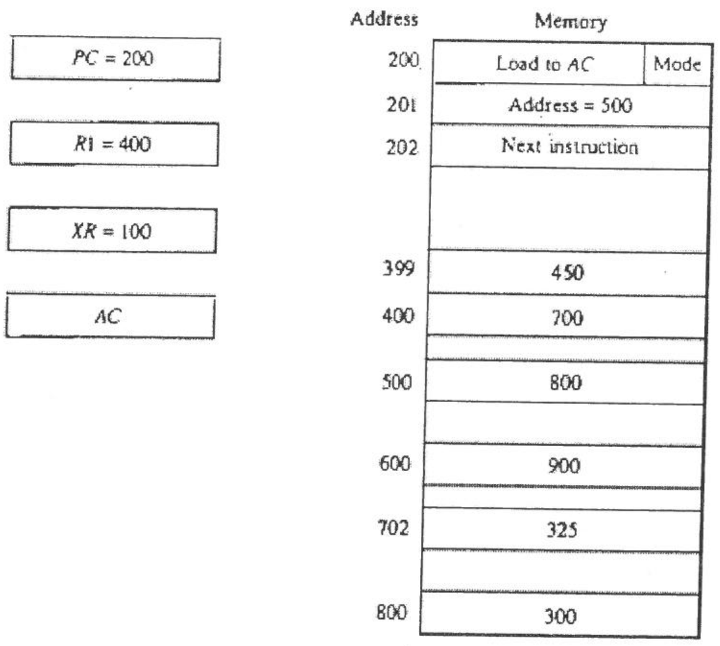
Indirect Address Mode: In this mode the address field of the instruction gives the address where the effective address is stored in memory. Control fetches the instruction from memory and uses its address part to access memory again to read the effective address.

Relative Address Mode: In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address.

Indexed Addressing Mode: In this mode the content of an index register is added to the address part of the instruction to obtain the effective address.

Base Register Addressing Mode: In this mode the content of a base register is added to the address part of the instruction to obtain the effective address.

Example :



Addressing Mode	Effective Address	Content of AC
Direct address	500	800
Immediate operand	201	500
Indirect address	800	300
Relative address	702	325
Indexed address	600	900
Register	—	400
Register indirect	400	700
Autoincrement	400	700
Autodecrement	399	450

7.b) Discuss decimal subtraction using 9's and 10's complements with examples.

Sol:

<u>Decimal subtraction using 10's complement</u>

Ex: M=72532 N=03250

Now we need to perform M-N. It can be written as M+(-N)

10's complement of 03250 = r^n-N=10^5-03250

=96750

```
Now given              M=72532
10's complement of     N=96750
                       ------------
                        169282
                       -----------
```

Discard the carry we get +**69282**

<u>Decimal subtraction using 9's complement</u>

Ex: M=72532 N=03250

Now we need to perform M-N. It can be written as M+(-N)

9's complement of 03250 = 99999-03250=96749

```
Now given              M=72532
9's complement of      N=96749
                       ------------
                        169281
                       -----------
```

Discard the carry and add 1 to the LSB bit of the answer 69282+1= +69282

Note: The above example is for M>N. in the same way for M<N. If anyone did in M<N consider that also.

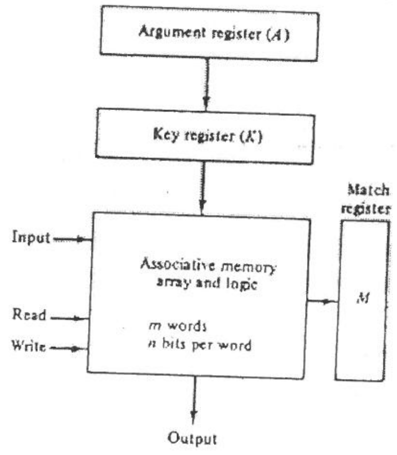
UNIT-IV

8.a) Describe the read and write operations of associative memory.

Sol: A memory unit accessed by content is called an associative memory or content addressable memory (CAM). This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location. When a word is written in an associative memory, no address is given. The memory is capable of finding an empty unused location to store the word. When a word is to be read from an associative memory, the content of the word, or part of the word, is specified. The memory locates all words which match the specified content and marks them for reading.

The key register provides a mask for choosing a particular field or key in the argument word. The entire argument is compared with each memory word if the key register contains all l' s. Otherwise, only those bits in the argument that have l's in their corresponding position of the key register are compared.

Block diagram of associative memory.



Read Operation:

If more than one word in memory matches the unmasked argument field, all the matched words will have 1's in the corresponding bit position of the match register. It is then necessary to scan the bits of the match register one at a time. The matched words are read in sequence by applying a read signal to each word line whose corresponding M, bit is a 1.

In most applications, the associative memory stores a table with no two identical items' under a given key. In this case, only one word may match the unmasked argument field. By connecting output M; directly to the read line in the same word position (instead of the M register), the content of the matched word will be presented automatically at the output lines and no special read command signal is needed. Furthermore, if we exclude words having a zero content, an all-zero output will indicate that no match occurred and that the searched item is not available in memory.

Write Operation:

An associative memory must have a write capability for storing the information to be searched. Writing in an associative memory can take different forms, depending on the application. If the entire memory is loaded with new information at once prior to a search operation then the writing can be done by addressing each location in sequence. This will make the device a random access memory for writing and a content addressable memory for reading. The advantage here is that the address for input can be decoded as in a random access memory.

If unwanted words have to be deleted and new words inserted one at a time, there is a need for a special register to distinguish between active and inactive words. This register, sometimes called a tag register, would have as many bits as there are words in the memory. For every active word stored in memory, the corresponding bit in the tag register is set to 1. A word is deleted from memory by clearing its tag bit to 0. Words are stored in memory by scanning the tag register until the first 0 bit is encountered. This gives the first available inactive word and a position for writing a new word. After the new word is stored in memory it is made active by setting its tag bit to 1. An unwanted word when deleted from memory can be cleared to all 0' s if this value is used to specify an empty location.

8.b) Explain the concept of virtual memory and its need in modern systems.

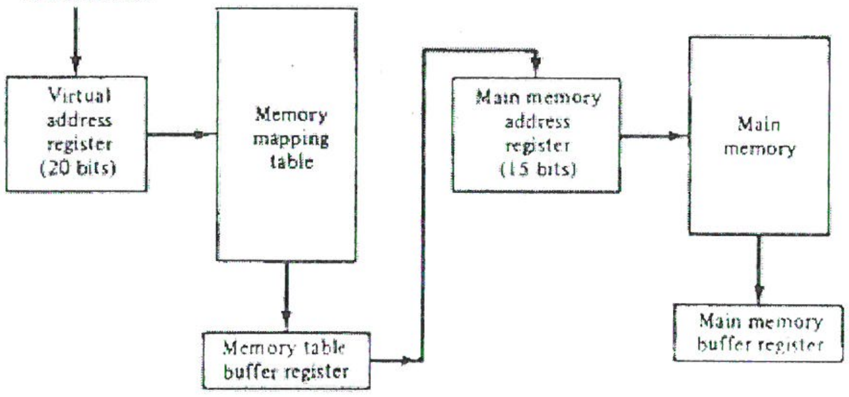
Sol: Virtual memory is a concept used in some large computer systems that permit the user to construct programs as though a large memory space were available, equal to the totality of auxiliary memory. Each address that is referenced by the CPU goes through an address mapping from the so-called virtual address to a physical address in main memory. Virtual memory is used to give programmers the illusion that they have a very large memory at their disposal, even though the computer actually has a relatively small main memory.

Address Space and Memory Space

An address used by a programmer will be called a **virtual address**, and the set of such addresses the address space. An address in main memory is called a location or **physical address.** The set of such locations is called the memory space.

The mapping table may be stored in a separate memory as shown in the following figure or in main memory.

Memory table for mapping a virtual address.


Virtual address
Virtual address register (20 bits)
Memory mapping table
Memory table buffer register
Main memory address register (15 bits)
Main memory
Main memory buffer register

In the first case, an additional memory unit is required as well as one extra memory access time. In the second case, the table takes space from main memory and two accesses to memory are required with the program running at half speed. A third alternative is to use an associative memory as explained below.

The table implementation of the address mapping is simplified if the information in the address space and the memory space are each divided into groups of fixed size. The physical memory is broken down into groups of equal size called blocks, which may range from 64 to 4096 words each. The term **page** refers to groups of address space of the same size. The programs are also considered to be split into pages. Portions of programs are moved from auxiliary memory to main memory in records equal to the size of a page. The term "**page frame**" is sometimes used to denote a block

Virtual memory Need in Modern systems

Modern systems need virtual memory to enable multitasking, run large applications, and improve system security and stability. It allows the computer to run more programs simultaneously by using a portion of the hard disk as an extension of physical RAM, and it provides memory protection to prevent one program from interfering with another.

OR

9.a) Explain the need for auxiliary memory in computer systems.

Sol:

Need for the Auxiliary memory

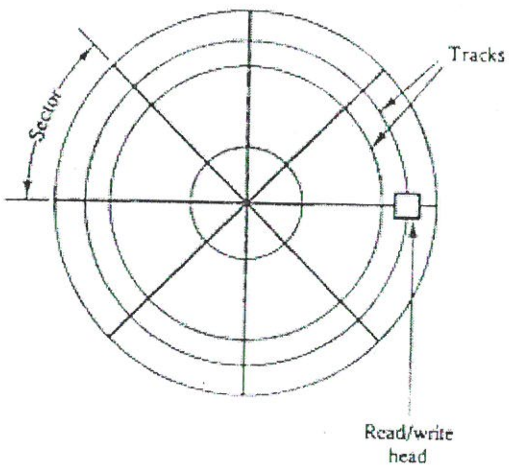
- **Permanent storage:** Auxiliary memory is non-volatile, meaning data remains stored even when the computer is turned off. This is essential for long-term storage of the operating system, user files, and applications.
- **Large capacity:** Auxiliary memory devices like hard drives and SSDs have a much larger storage capacity than primary(RAM) allowing them to store vast amounts of data that are too large to fit into RAM.
- **Cost-effectiveness:** Auxiliary memory is significantly cheaper per gigabyte than primary memory **Cost-**, making it an economical choice for storing large volumes of data.
- **Data backup and transfer:** Auxiliary memory is vital for backing up important data to prevent loss and for easily transferring data between computers using devices like external hard drives and Data USB drives.

The most common auxiliary memory devices used in computer systems are magnetic disks and tapes. Magnetic drums and disks are quite similar in operation. Both consist of high-speed rotating surfaces coated with a magnetic recording medium. The rotating surface of the drum is a cylinder and that of the disk, a round flat plate. The recording surface rotates at uniform speed and is not started or stopped during access operations. Bits are recorded as magnetic spots on the surface as it passes a stationary mechanism called a write head. Stored bits are detected by a change in magnetic field produced by a recorded spot on the surface as it passes through a read head. The amount of surface available for recording in a disk is greater than in a drum of equal physical size. Therefore, more information can be stored on a disk than on a drum of comparable size. For this reason, disks have replaced drums in more recent computers.

Magnetic Disks

A magnetic disk is a circular plate constructed of metal or plastic coated with magnetized material. Often both sides of the disk are used and several disks may be stacked on one spindle with read/write heads available on each surface. All disks rotate together at high speed and are not stopped or started for access purposes. Bits are stored in the magnetized surface in spots along concentric circles called tracks. The tracks are commonly divided into sections called sectors. In most systems, the minimum quantity of information which can be transferred is a sector.

The subdivision of one disk surface into tracks and sectors is shown in the following figure



Some units use a single read/write head for each disk surface. In this type of unit, the track address bits are used by a mechanical assembly to move the head into the specified track position before reading or writing. In other disk systems, separate read/write heads are provided for each track in each surface. The address bits can then select a particular track electronically through a decoder circuit. This type of unit is more expensive and is found only in very large computer systems. Permanent timing tracks are used in disks to synchronize the bits and recognize the sectors. A disk system is addressed by address bits that specify the disk number, the disk surface, the sector number and the track within the sector. After the read/write heads are positioned in the specified track, the system has to wait until the rotating disk reaches the specified sector under the read/write head. Information transfer is very fast once the beginning of a sector has been reached. Disks may have multiple heads and simultaneous transfer of bits from several tracks at the same time.

Magnetic Tape

A magnetic tape transport consists of the electrical, mechanical, and electronic components to provide the parts and control mechanism for a magnetic-tape unit. The tape itself is a strip of plastic coated with a magnetic recording

A magnetic tape transport consists of the electrical, mechanical, and electronic components to provide the parts and control mechanism for a magnetic-tape unit. The tape itself is a strip of plastic coated with a magnetic recording medium. Bits are recorded as magnetic spots on the tape along several tracks. Usually, seven or nine bits are recorded simultaneously to form a character together with a parity bit. Read/write heads are mounted one in each track so that data can be recorded and read as a sequence of characters. Magnetic tape units can be stopped, started to move forward or in re verse, or can be rewound.

However, they cannot be started or stopped fast enough between individual characters. For this reason, information is recorded i n blocks referred to as records. Gaps of unrecorded tape are inserted between records where the tape can be stopped. The tape starts moving while in a gap and attains its constant speed by the time it reaches the next record. Each record on tape has an identification bit pattern at the beginning and end. By reading the bit pattern at the beginning, the tape control identifies the record number.

9.b) Explain the working of write-through and write-back policies in cache memory.

Sol: An important aspect of cache organization is concerned with memory write requests. When the CPU finds a word in cache during a read operation, the main memory is not involved in the transfer. However, if the operation is a write, there are two ways that the system can proceed.

Write-through

The simplest and most commonly used procedure is to update main memory with every memory write operation, with cache memory being up dated in parallel if it contains the word at the specified address. This is called the write-through method. This method has the advantage that main memory always contains the same data as the cache. This characteristic is important in systems with direct memory access transfers.

Write-back

The second procedure is called the write-back method. In this method only the cache location is updated during a write operation. The location is then marked by a flag so that later when the word is removed from the cache it is copied into main memory. The reason for the write-back method is that during the time a word resides in the cache, it may be updated several times; however, as long as the word remains in the cache, it does not matter whether the copy i n main memory is out of date, since requests from the word are filled from the cache. It is only when the word is displaced from the cache that an accurate copy need be rewritten into main memory.

UNIT-V

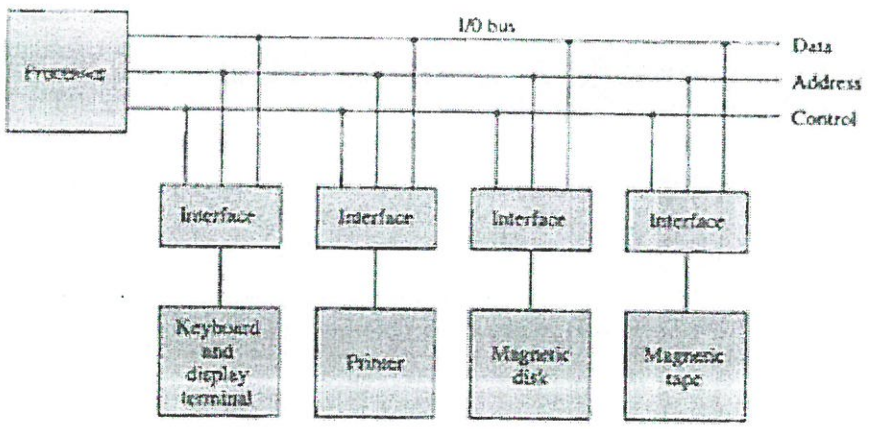
10.a) Describe the functional block diagram of an I/O module and explain its operation.

Sol: To resolve these differences, computer systems include special hardware components between the CPU and peripherals to supervise and synchronize all input and output transfers. These components are called interface units because they interface between the processor bus and the peripheral device.

I/O Bus and Interface Modules

A typical communication link between the processor and several peripherals is shown in the following diagram

Connection of I/O bus to input-output devices



The I/O bus consists of data lines, address lines, and control lines.. Each peripheral device has associated with it an interface unit. Each interface deoodes the address and control received from the I/O bus, interprets them for the peripheral, and provides signals for the periphe.ral controller. It also synchronizes the data flow and supervises the transfer between peripheral and processor. Each peripheral has its own controller that operates the particular electromechanical device. A controller may be housed separately or may be physically integrated with the peripheral.

The I/O bus from the processor is attached to all peripheral interfaces. To communicate with a particular device, the processor places a device address on the address lines. Each interface attached to the I/O bus contains an address decoder that monitors the address lines. When the interface detects its own address, it activates the path between the bus lines and the device that it controls. All peripherals whose address does not correspond to the address in the bus are disabled by their interface.

- A control command is issued to activate the peripheral and to inform it what to do
- A status command is used to test various status conditions in the interface and the peripheral.

- A data output command causes the interface to respond by transferring data from the bus into one of its registers.
- The data input command is the opposite of the data output. In this case the interface receives an item of data from the peripheral and places it in its buffer register

10b) Explain asynchronous data transfer with a neat diagram.

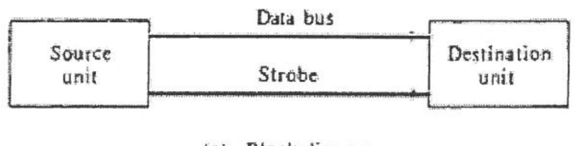
Sol: In most cases, the internal timing in each unit is independent from the other in that each uses its own private clock for internal registers. In that case, the two units are said to be asynchronous to each other.

Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted. One way of achieving this is by means of a **strobe pulse** supplied by one of the units to indicate to the other unit when the transfer has to occur.

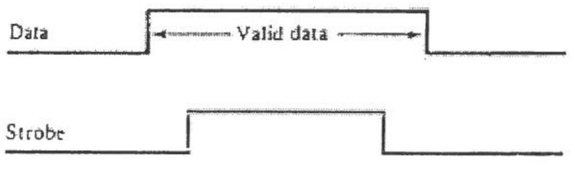
Another method commonly used is to accompany each data item being transferred with a control signal that indicates the presence of data in the bus. The unit receiving the data item responds with another control signal to acknowledge receipt of the data. This type of agreement between two independent units is referred to as **Handshaking.**

Strobe Control

The strobe control method of asynchronous data transfer employs a single control line to time each transfer. The strobe may be activated by either the source or the destination unit. The following shows a **source-initiated transfer**



(a) Block diagram

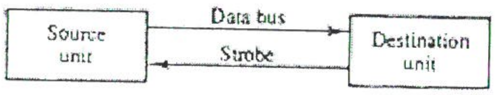


(b) Timing diagram

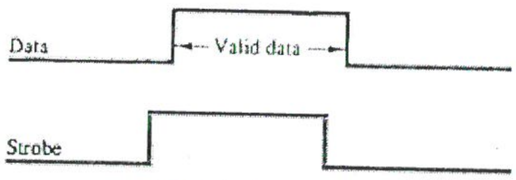
The strobe is a single line that informs the destination unit when a valid data word is available in the bus. the source unit first places the data on the data bus. After a brief delay to ensure that the data settle to a steady value, the source activates the strobe pulse. The information on the data bus and the strobe signal remain in the active state for a sufficient time period to allow the destination unit to receive the data. Often, the destination unit uses the falling edge of the strobe pulse to transfer the contents of the data bus into one of its internal registers. The source removes the data from the bus a brief period after it disables its strobe pulse. Actually, the source does not have to change the information in the data bus. The fact that the strobe signal is disabled indicates that the data bus does not contain valid data. New valid data will be available only after the strobe is enabled again.

Destination initiated strobe

In this case the destination unit activates the strobe pulse, informing the source to provide the data. The source unit responds by placing the requested binary information on the data bus. The data must be valid and remain in the bus long enough for the destination unit to accept it. The falling edge of the strobe pulse can be used again to trigger a destination register. The destination unit then disables the strobe. The source removes the data from the bus after a predetermined time interval



(a) Block diagram

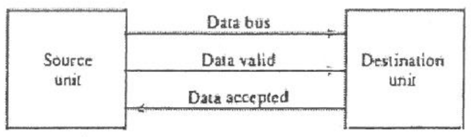


(b) Timing diagram

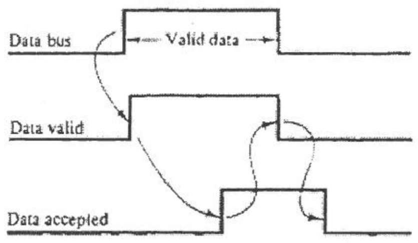
Handshaking:

The disadvantage of the strobe method is that the source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was placed in the bus. The handshake method solves this problem by introducing a second control signal that provides a reply to the unit that initiates the transfer

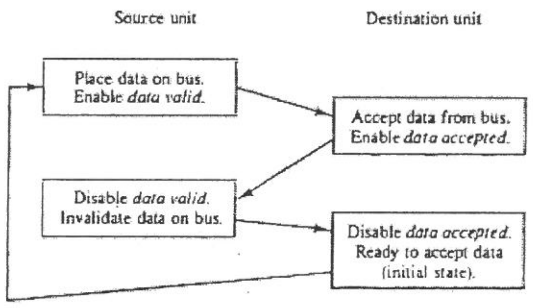
The following figure shows the **data transfer procedure when initiated by the source**. The two handshaking lines are data valid, which is generated by the source unit, and data accepted, generated by the destination unit. The timing diagram shows the exchange of signals between the two units.



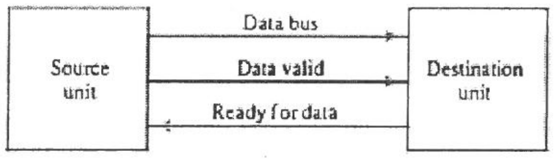
(a) Block diagram



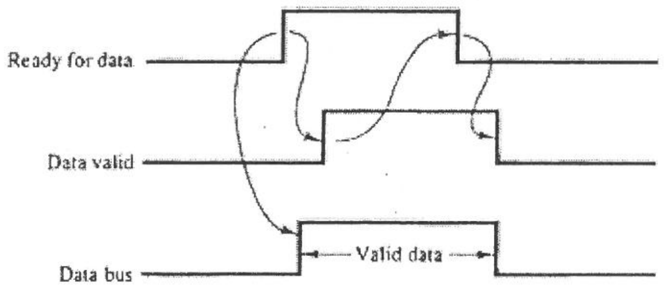
(b) Timing diagram


Source unit
Destination unit
Place data on bus. Enable data valid.
Accept data from bus. Enable data accepted.
Disable data valid. Invalidate data on bus.
Disable data accepted. Ready to accept data (initial state).

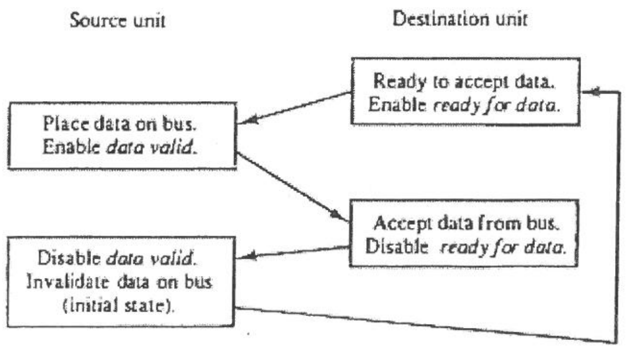
The destination-initiated transfer using handshaking lines is shown in the following figure



(a) Block diagram



(b) Timing diagram

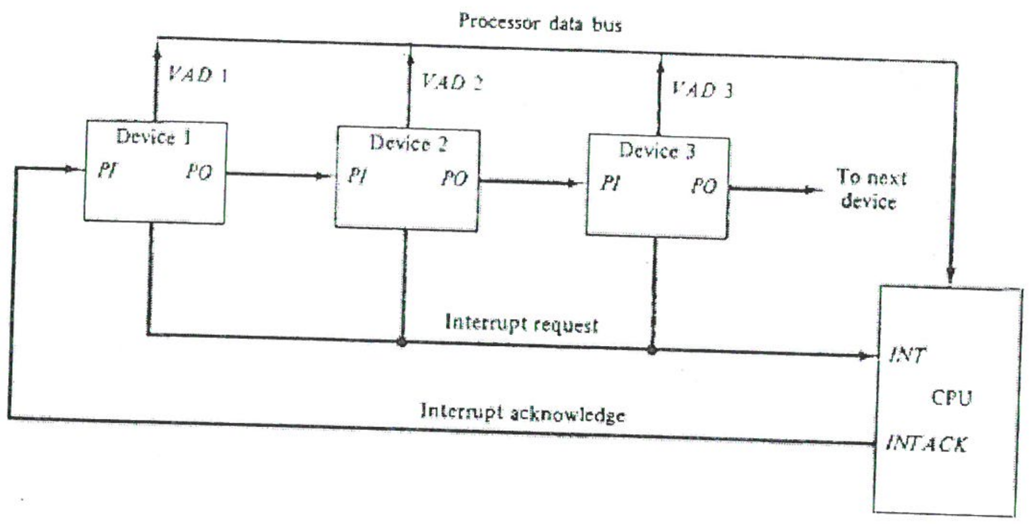

Source unit
Destination unit
Ready to accept data. Enable ready for data.
Place data on bus. Enable data valid.
Accept data from bus. Disable ready for data.
Disable data valid. Invalidate data on bus (initial state).

The signal generated by the destination unit has been changed to ready for data to reflect its new meaning. The source unit in this case does not place data on the bus until after it receives the ready for data signal from the destination unit. From there on, the handshaking procedure follows the same pattern as in the source-initiated case.

OR

11.a) Discuss daisy-chaining priority interrupt with a neat diagram.

Sol: The daisy-chaining method of establishing priority consists of a serial connection of all devices that request an interrupt. The device with the highest priority is placed in the first position, followed by lower-priority devices up to the device with the lowest priority, which is placed last in the chain. This method of connection between three devices and the CPU is shown in the following diagram



The interrupt request line is common to all devices and forms a wired logic connection. If any device has its interrupt signal in the low-level state, the interrupt line goes to the low-level state and enables the interrupt input in the CPU. W hen no interrupts are pending, the interrupt line stays in the high-level state and no interrupts are recognized by the CPU. This is equivalent to a negative logic OR operation. The CPU responds to an interrupt request by enabling the interrupt acknowledge line. This signal is received by device 1 at its PI (priority in) input. The acknowledge signal passes on to the next device through the PO (priority out) output only if device 1 is not requesting an interrupt. If device 1 has a pending interrupt, it blocks the acknowledge signal from the next device by placing a 0 in the PO output. It then proceeds to insert its own interrupt vector address (VAD) vector address (VAD) into the data bus for the CPU to use during the interrupt cycle.

A device with a 0 in its PI input generates a 0 in its PO output to inform the next-lower-priority device that the acknowledge signal has been blocked. A device that is requesting an interrupt and has a 1 in its PI input will intercept the acknowledge signal by placing a 0 in its PO output. If the device does not have pending interrupts, it transmits the acknowledge signal to the next device. by placing a 1 in its PO output. Thus the device with PI = 1 and PO = 0 is the one with the highest priority that is requesting an interrupt, and this device places its VAD on the data bus. The daisy chain arrangement gives the highest priority to the device that receives the interrupt acknowledge signal from the CPU. The farther the device is from the first position, the lower is its priority

11.b) Describe the working of a DMA controller with neat diagram.

Sol: The transfer of data between the peripheral and memory without the interaction of CPU and letting the peripheral device manage the memory bus directly is termed as Direct Memory Access (DMA).

DMA Controller
The DMA controller communicates with the CPU through the data bus and control lines. DMA select signal is used for selecting the controller, the register select is for selecting the register. When the bus grant signal is zero, the CPU communicates through the data bus to read or write into the DMA register. When bus grant is one, the DMA controller takes the control of buses and transfers the data between the memory and I/O.

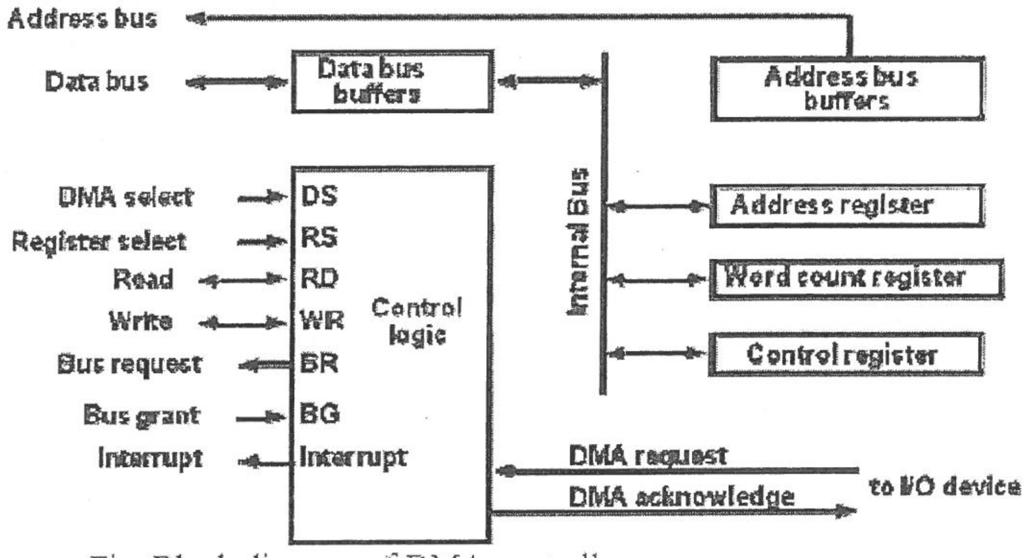


Fig: Block diagram of DMA controller

The address register specifies the desired location of the memory which is incremented after each word is transferred to the memory. The word count register holds the number of words to be transferred which is decremented after each transfer until it is zero. When it is zero, it indicates the end of transfer. After which the bus grant signal from CPU is made low and CPU returns to its normal operation. The control register specifies the mode of transfer which is Read or Write.