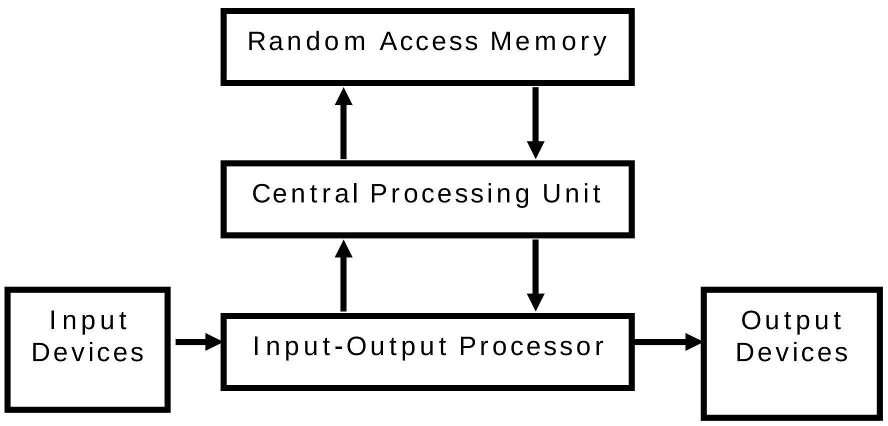
UNIT 1: DIGITAL LOGICAL CIRCUITS

What is Digital Computer? OR Explain the block diagram of digital computers.

- Digital computer is a digital system that performs various computational tasks.
- The word "DIGITAL" implies that the information in the computer is represented by digits.
- Digital computers use the binary number system, which has two digits, 0 and 1.
- A binary digit is called a *bit*.
- Information is represented in digital computers in the groups of bit.
- By using various coding techniques, groups of bits can be made to represent not only binary numbers but also other symbols, letters of alphabets and decimal digit.
- Bits are grouped together as *bytes* and *words* to form some type of representation within the computer.
- A sequence of instructions for the computer is known as *program*.

Block diagram of a digital computer



- The hardware of the computer is usually divided into three major parts.
- The Central processing Unit (CPU) contains an arithmetic and logic unit for manipulating data, a number of registers for storing data and control circuits for fetching and executing instructions.
- The memory of a computer contains storage for instructions and data, it is called a Random Access Memory (RAM) because the CPU can access any location in memory at random and retrieve the binary information within a fixed interval of time.
- The input and output processor contains electronic circuit for communication and controlling the transfer of information between the computer and the outside world.
- The input and output device connected to the computer include keyboards, printers, terminals, magnetic disk drives and other communication devices.

What is Gates? Explain the Logic Gates in brief.

- Binary information is represented in digital computers using electrical signals.
- These signals can be represented by voltage to specify one of two possible states.
- The two states represent a binary variable that can be equal to 1 or 0.
- The manipulation of binary information in a computer is done using logic circuits called *gates*.

- Gates are blocks of hardware that produce signals of binary 1 or 0 when input logic requirements are satisfied.
- There are various types of logic gates are commonly used in digital computer.
- Each gate has a different graphic symbols and operation.
- The input-output relationship of binary variables for each gate can be represented in tabular form by Truth-Table.
- There are three types of gates:
  - Basic / Fundamental Gates (AND, OR, NOT)
  - Universal Gates (NAND, NOR)
  - Exclusive Gates (EX-OR, EX-NOR)

LOGICAL GATES

Basic / Fundamental Gates **Universal Gates** **Exclusive Gates**

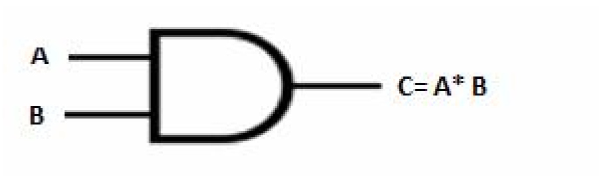
(AND, OR, NOT) **(NAND, NOR)** **(EX-OR, EX-NOR)**

Basic Gates

AND Gate:

- In this type of gate output is high only when all its inputs are high.
- If any single input is law then the output will remain low.
- So it is said that in AND gate the output is only high when the input is also high.

SYMBOL:



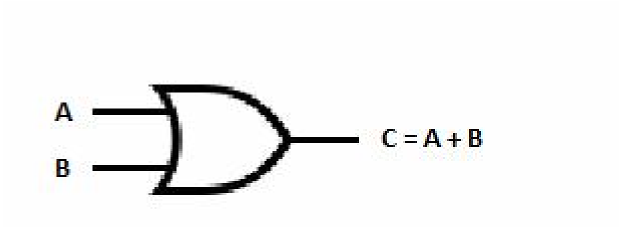
TRUTH-TABLE:

INPUT		OUTPUT
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate:

- In this type of gate if any input signal is high then the output will be high.
- The output is only low only when all the inputs are low

SYMBOL:



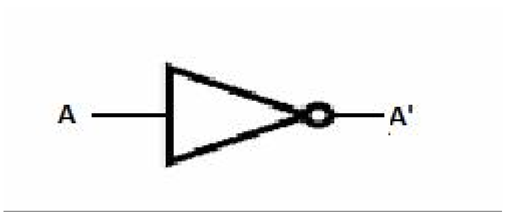
TRUTH-TABLE:

INPUT		OUTPUT
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

NOT Gate:

- This type of gate is also known as "Inverter".
- It is a gate that contains only one input and only one output.
- The output is always opposite than the input signals.

SYMBOL:



TRUTH-TABLE:

INPUT	OUTPUT
A	NOT A (A')
0	1
1	0

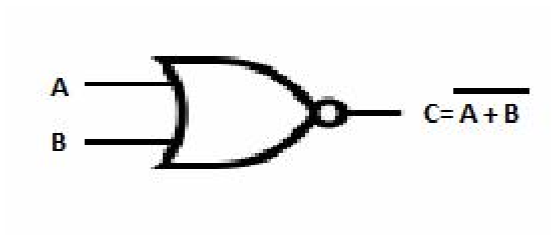
Universal Gates

NAND and NOR gates are known as universal gates because we can construct any gate using NAND & NOR gate.

NOR Gate:

- The NOR gate is the complement of the OR gate.
- As shown in the truth table that the output of NOR gate is exactly opposite than the output of OR gate.
- This means that the output will be high when all the input is low.

SYMBOL:



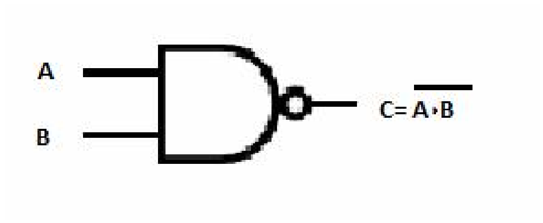
TRUTH-TABLE:

INPUT		OUTPUT
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

NAND Gate:

- The NAND gate is an AND gate followed by NOT gate.
- As shown in the truth table that the output of NAND gate is exactly opposite than the output of AND gate.
- This means that the output will be high when all the input is high.

SYMBOL:



TRUTH-TABLE:

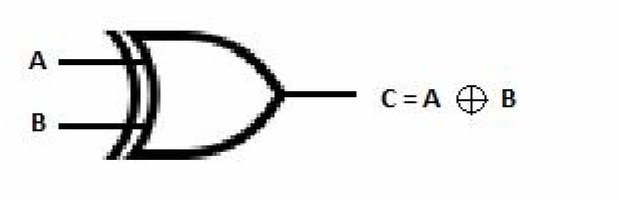
INPUT		OUTPUT
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

Exclusive Gates

EX-OR Gate:

- This gate is produces high output whenever the two inputs are at opposite level.
- The EX-OR gate is the gate that produces high output for Odd number of high inputs.

SYMBOL:



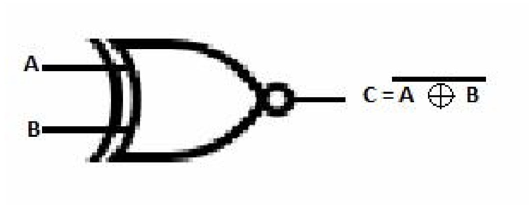
TRUTH-TABLE:

INPUT		OUTPUT
A	B	A EX-OR B
0	0	0
0	1	1
1	0	1
1	1	0

EX-NOR Gate:

- This gate is produces high output whenever the two inputs are at same level.
- The EX-OR gate is the gate that produces high output for Even number of high inputs.
- The truth table shows that output of this gate is exactly opposite of EX-OR gate.

SYMBOL:



TRUTH-TABLE:

INPUT		OUTPUT
A	B	A EX-NOR B
0	0	1
0	1	0
1	0	0
1	1	1

Write a note on Boolean Algebra

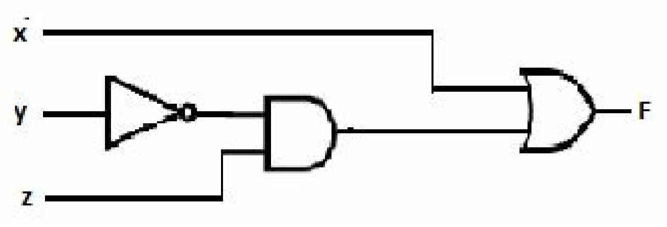
- In 1854 George Boole introduced a systemic treatment of logic and developed for this purpose an algebric system called Boolean Algebra.
- Boolean Algebra is an algebra that deals with binary variables and logic operations.
- The variables are designated by letters such as A,B, X ,Y etc.
- The three basic operations are AND, OR and complement.
- A Boolean function can be expressed with binary variable, the logic operation symbols, parentheses (rounded bracket) and equal to (=) sign.
- The result of a Boolean function is either 0 or 1.

- A Boolean function can be represented by either:

    a. Truth tables
    b. Logic diagrams
    c. Algebraic expression

- For example: F=x+y’z
    - F=1 only if x is 1 or if both y’ and z=1.
    - If y’(complement of y)=1 means that y=0 so we can say that F=1 only when x=1,y=0,z=1.
    - So we can say that function F equal to 1 for those combination where x=1 or yz=01
- A Boolean function can be transformed form algebraic expression into a logic diagram composed of AND,OR and NOT gates.
- **Truth table and logic diagram For above example :**

x	Y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



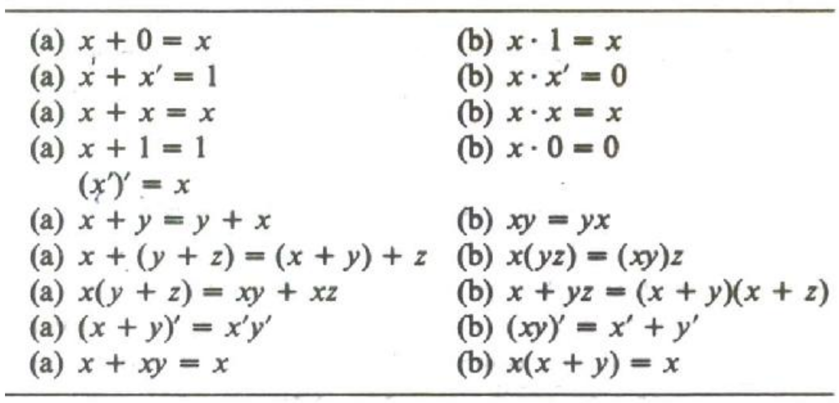
Boolean Operations

There are three basic logical operations:

- AND: This operation is represented by a dot or by the absence of an operator. For example, x.y = z or xy = z is read “x AND y is equal to z.” The logical operation AND is interpreted to mean that z=1 if and only if x=1 and y=1; otherwise z=0.
- OR: This operation is represented by a plus sign. For example, x+y = z is read “x OR y is equal to z”, meaning that z=1 if x=1 or if y=1 or if both x=1 and y=1. If both x=0 and y=0, then z=0.
- NOT: This operation is represented by a prime (sometimes by a bar). For example, x’ = z (or x = z) is read “x not is equal to z”, meaning that z is complement of x. In other words, if x=1, then z=0, but if
- x=0, then z=1.

Basic Identities of Boolean Algebra

Postulates and theorems of Boolean Algebra



(a) $x + 0 = x$	(b) $x \cdot 1 = x$
(a) $x + x' = 1$	(b) $x \cdot x' = 0$
(a) $x + x = x$	(b) $x \cdot x = x$
(a) $x + 1 = 1$	(b) $x \cdot 0 = 0$
$(x')' = x$	
(a) $x + y = y + x$	(b) $xy = yx$
(a) $x + (y + z) = (x + y) + z$	(b) $x(yz) = (xy)z$
(a) $x(y + z) = xy + xz$	(b) $x + yz = (x + y)(x + z)$
(a) $(x + y)' = x'y'$	(b) $(xy)' = x' + y'$
(a) $x + xy = x$	(b) $x(x + y) = x$

Write a note on DeMorgan's theorem

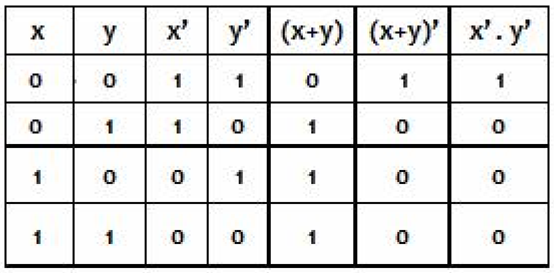
- It is developed greater mathematician and logician named De-Morgan.
- He developed two theorems which makes the complement of questions and product(**incomplete line).**
- It is very important in dealing with NOR and NAND gates.
- The 2 most important theorems of De-Morgan are as follows:

  1) The complement of sum equal to the product of the complement.

  $$\overline{x + y} = \overline{x} \cdot \overline{y}$$

  2) The complement of product equal to the sum of the complement.

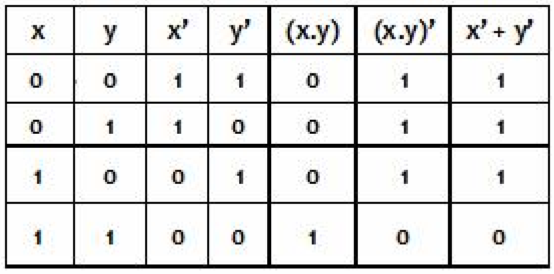
  $$\overline{x \cdot y} = \overline{x} + \overline{y}$$
- We can prove the theorems with the help of truth table.
- **THEOREM 1:** $\overline{x + y} = \overline{x} \cdot \overline{y}$



x	y	x'	y'	(x+y)	(x+y)'	x' . y'
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

- Last 2 columns gives same output so LHS=RHS

THEOREM 2: $\overline{x \cdot y} = \overline{x} + \overline{y}$

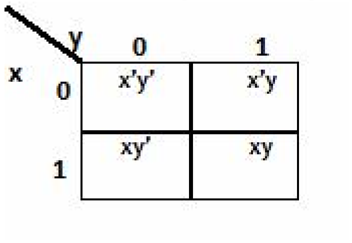


x	y	x'	y'	(x.y)	(x.y)'	x' + y'
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

- Last 2 columns gives same output so LHS=RHS

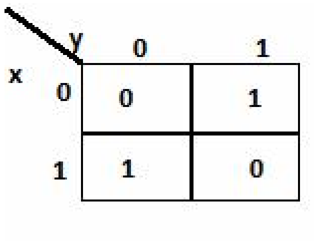
Write a note on K- Maps.

- The Karnaugh map is also referred as Veitch Diagrams, KV maps or K-maps.
- K-map is a method to minimizes the Boolean function.
- K-map provides a simple and straight forward method to minimizing Boolean expression.
- With the help of K-map we can simplified Boolean expression up to 4 and 6 variables.
- K-map diagram represents squares and each square represents 1 minterm.
- In K-map values of the variables are written in binary form & the logic function can be expressed in one of the following form
    - **SUM OF PRODUCTS (SOP)**
    - **PRODUCT OF SUM (POS)**
- A K-map for n variables is made up of **2^n** squares and each squares designed a product term of Boolean expression.
- For product terms which are present in expression, 1s are written in correspondence squares and 0 will be written in blank square.
- **For example: K-map for 2 variables:**



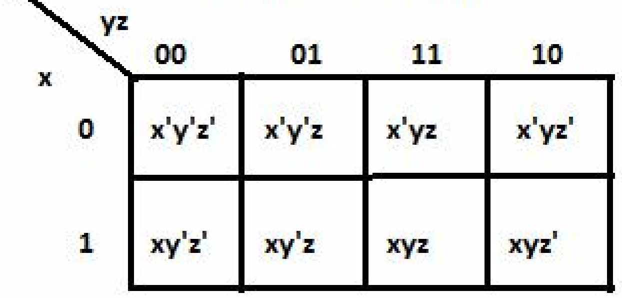
x \ y	0	1
0	x'y'	x'y
1	xy'	xy

- F =xy' + x'y



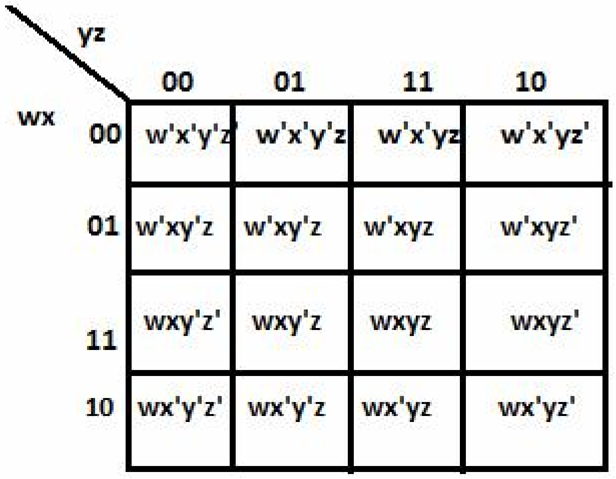
x \ y	0	1
0	0	1
1	1	0

k- map for 3 variables



x \ yz	00	01	11	10
0	x'y'z'	x'y'z	x'yz	x'yz'
1	xy'z'	xy'z	xyz	xyz'

k-map for 4 variables



wx \ yz	00	01	11	10
00	w'x'y'z'	w'x'y'z	w'x'yz	w'x'yz'
01	w'xy'z	w'xy'z	w'xyz	w'xyz'
11	wxy'z'	wxy'z	wxyz	wxyz'
10	wx'y'z'	wx'y'z	wx'yz	wx'yz'

- **<u>RULES FOR K- MAP:</u>**
- Each cell with 1 must be included in at list 1 group.
- Try to form the largest possible groups.
- Try to end up with as few groups as possible.
- Groups may be in sizes that are powered of 2.
- Groups may be square or rectangular only.
- Groups may be horizontal or vertical but not diagonal.
- Groups may wrap around the table.
- Groups may overleap.
- The larger a group is, the more redundant inputs there are:
    - Group of 1 has no redundant input.
    - Group of 2 known as pair has 1 redundant input.
    - Group of 4 known as quad has 2 redundant input.
    - Group of 8 known as octet has 3 redundant input.

Sum-of-Products Simplification

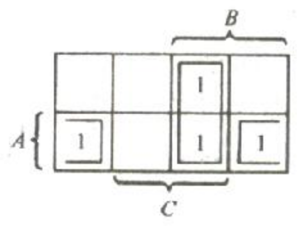
- A Boolean function represented by a truth table is plotted into the map by inserting 1's into those squares where the function is 1.
- Boolean functions can then be simplified by identifying adjacent squares in the Karnaugh map that contain a 1.
- A square is considered adjacent to another square if it is next to, above, or below it. In addition, squares at the extreme ends of the same horizontal row are also considered

adjacent. The same applies to the top and bottom squares of a column. The objective to identify adjacent squares containing 1's and group them together.

- Groups must contain a number of squares that is an integral power of 2.
- Groups of combined adjacent squares may share one or more squares with one or more groups.
- Each group of squares represents an algebraic term, and the **OR** of those terms gives the simplified algebraic expression for the function.
- To find the most simplified algebraic expression, the goal of map simplification is to identify the least number of groups with the largest number of members.

We will simplify the Boolean function.
F (A,B,C) = Σ(3,4,6,7)

Map for F(A,B,C) = p(3,4,6,7)



- The three variable maps for this function is shown in the figure 2.4
- There are four squares marked with 1's, one for each minterm that produces 1 for the function. These squares belong to minterm 3,4,6,7 and are recognized from the figure b.
- Two adjacent squares are combined in the third column. This column belongs to both B and C produces the term BC.
- The remaining two squares with 1's in the two corner of the second row are adjacent and belong to row columns of C', so they produce the term AC'.
- The simplified expression for the function is the or of the two terms:
  F = BC + AC'

The second example simplifies the following Boolean function:
F(A,B,C) = Σ (0,2,4,5,6)

- The five minterms are marked with 1's in the corresponding squares of the three variable maps.
- The four squares in the first and the fourth columns are adjacent and represent the term C'.
- The remaining square marked with a 1 belongs to minterm 5 and can be combined with the square of minterm 4 to produce the term AB'.

The simplified function is
F = C'+AB'
Map for F(A,B,C) = p(0,2,4,5,6)

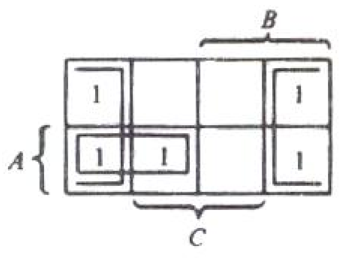
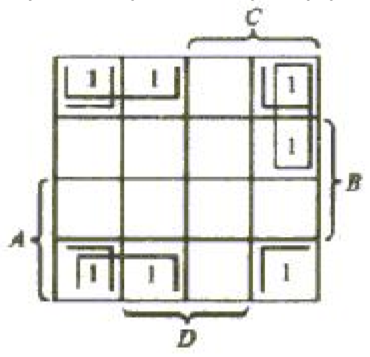


Figure 2.6 Map for F(A,B,C,D) = p(0,1,2,6,8,9,10)



- The area in the map covered by this four variable consists of the squares marked with 1's in fig 1.10. The function contains 1's in the four corners that when taken as groups give the term B'D'. This is possible because these four squares are adjacent when the map is considered with the top and bottom or left and right edges touching.
- The two 1's on the bottom row are combined with the two 1's on the left of the bottom row to give the term B'C'.
- The remaining 1 in the square of minterm 6 is combined with the minterm 2 to give the term A'CD'.

  The simplified function is:
  F = B'D' + B'C' + A'CD'

Product-of-Sums Simplification

- Another method for simplifying Boolean expressions can be to represent the function as a product of sums.
- This approach is similar to the Sum-of-Products simplification, but identifying adjacent squares containing 0's instead of 1's forms the groups of adjacent squares.
- Then, instead of representing the function as a sum of products, the function is represented as a product of sums.

  <u>Examples</u>
  F(A,B,C,D) = Σ (0,1,2,5,8,9,10)

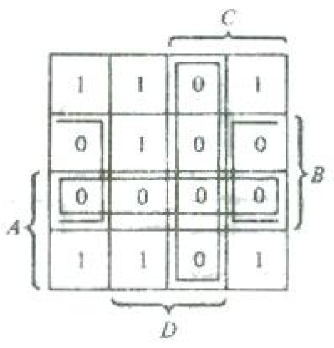
The 1's marked in the map of figure 2.7 represents the minterms that produces a 1 for the function.
The squares marked with 0's represent the minterm not included in F and therefore denote the complement of F.
Combining the squares with 1's gives the simplified function in sum-of-products form:

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

If the squares marked with 0's are combined as shown in the diagram, we obtain the simplified complement function:

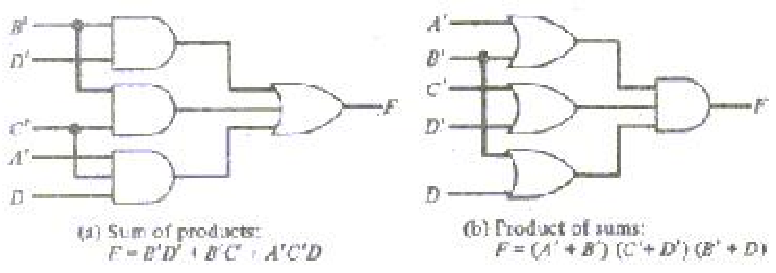
F’=(A’+B’)(C’D’)(B’+D)

Figure 2.7 Map for F(A,B,C,D) = p (0,1,2,5,8,9,10)



The logic diagram of the two simplified expression are shown in fig 2.8

Logic Diagram with AND and OR gates



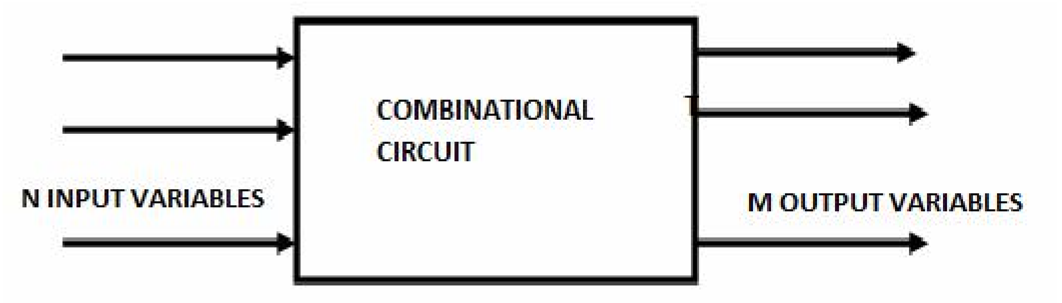
- The sum of product expression us implemented in fig 2.8(a) with a group of of AND gates, one for each AND term.
- The output of the AND gates are connected to the inputs of a single OR gate. The same function is implemented in fig 2.8(b) in product of sum form with a group of OR gates, one for each OR term, the outputs of the OR gates are connected to the inputs of a single And gate.
- In each case it is assumed that the input variable are directly available in their complement, so inverter are not included.

Write a note on Combinational Circuits

- A combinational circuit is the circuit where more than 1 circuit is designed into single component.
- It has N no of inputs and M no of outputs.
- It is basically used to design digital applications and it transforms the data into the digital manner.
- A combinational circuit is a connected arrangement of logic gates with a set of inputs and outputs.

- At any given time, the binary values of the outputs are a function of the binary values of the inputs.
- The design of a combinational circuit starts from a verbal outline of the problem and ends in a logic circuit diagram. The procedure involves the following steps:

  1. The problem is stated.
  2. The input and output variables are assigned letter symbols.
  3. The truth table that defines the relationship between inputs and outputs is derived.
  4. The simplified Boolean functions for each output are obtained.
  5. The logic diagram is drawn.

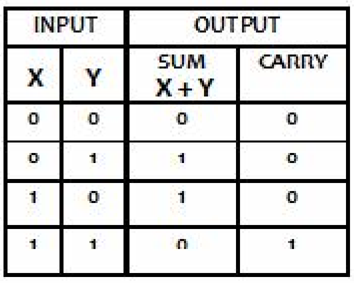


Arithmatic circuits:

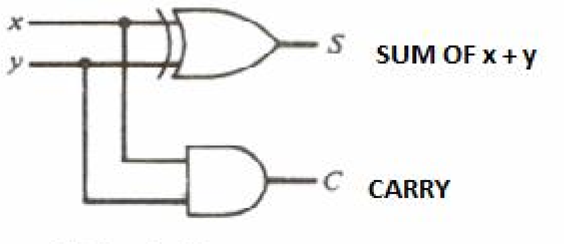
- It is made of different arithmetic operators. There will be addition, substraction, division, modules and any other arithmetic operations.

Half-Adder

- Half-Adder is a part of combinational circuit.
- It is basically designed for arithmetic addition.
- It is most basic digital arithmetic circuit.
- Performs the addition of two binary digits.
- The input variables of a half-adder are called the *augend* and the *addend*.
- The output variables of a half-adder are called the *sum* and the *carry*.



INPUT		OUTPUT	
X	Y	SUM X + Y	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



(b) Logic diagram

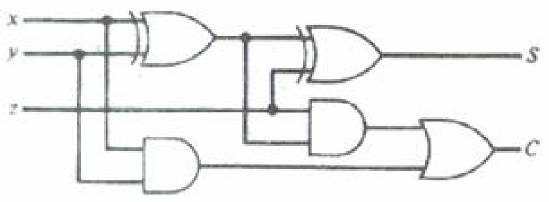
Full-Adder

- A full-adder performs the addition of three binary digits.
- Two half-adders can be combined to for a full-adder..
- Although a full adder has three inputs, it still only has two outputs since the largest number is 1+1+1 = 3, and 3 can be represented by two bits.

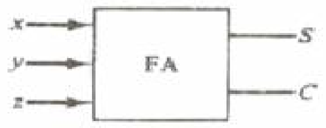
TRUTH TABLE OF FULL - ADDER

INPUT			OUTPUT	
X	Y	Z	SUM OF X+Y+Z	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

LOGIC DIAGRAM OF FULL - ADDER



BLOCK DIAGRAM OF FULL ADDER



K-MAP FOR FULL ADDER

X \ YZ	00	01	11	10
0		1		1
1	1		1	

WHAT IS THE DIFFERENCE BETWEEN HALF ADDER AND FULL ADDER?

Half adder	**Full adder**
The most basic digital arithmetic circuit.	A full-adder performs the addition of three binary digits.
Performs the addition of two binary digits.	It is used for multi bit additions.
Output is sum of two signals.	Output is sum of three signals.
There are two input and two output terminal.	There are three input and two output terminal.
From full adder half adder cant not be built	Two full adder makes one full adder
On EX-OR gate and one AND gate are used.	Two EX-OR, two AND and one OR gate is used.

WHAT IS THE DIFFERENCE BETWEEN COMBINATIONAL CIRCUIT AND SEQUENCIAL CIRCUIT?

COMBINATIONAL CIRCUIT	**SEQUENTIAL CIRCUIT**
It is a digital logic circuit whose output depends on the present inputs.	It is a digital logic circuit whose output depends on the present inputs as well as previous inputs.
It can describe by the output values.	It can describe by the output values as well as state values.
It contains no memory element.	It contains at least one memory element.
It is easy to design and understand.	It is difficult to design and understand.
It is faster in speed.	It is slower in speed.
It is expensive in cost.	It is less expensive in cost.
Examples of combinational circuit are half adder and full adder.	Examples of sequential circuit are flip-flops like RS, Clocked RS, D and JK.
A combinational circuit is a connected arrangement of logic gates with a set of inputs and outputs.	However, if a circuit uses both gates and flip-flops, it is called a sequential circuit.
At any given time, the binary values of the outputs are a function of the binary values of the inputs.	Hence, a sequential circuit is an interconnection of flip-flops and gates.
The design of a combinational circuit starts from a verbal outline of the problem and ends in a logic circuit diagram.	If we think of a sequential circuit as some black box that, when provided with some external input, produces some external output

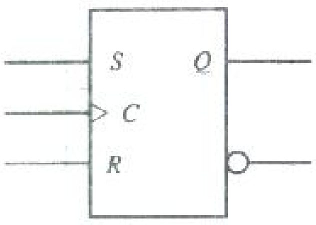
What is Flip-flops

- A Flip-flop is a binary cell capable of storing one bit of information.
- It has two outputs, one for the normal value and one for the complement value of the bit stored in it.
- Flip-flops are storage elements utilized in synchronous sequential circuits.
- Synchronous sequential circuits employ signals that effect storage elements only at discrete instances of time.
- A timing device called a clock pulse generator that produces a periodic train of clock pulses achieves synchronization.

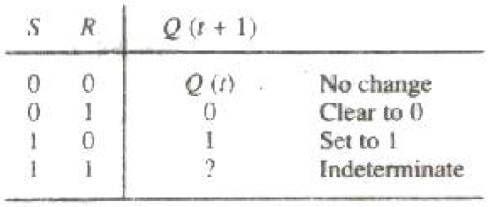
- Values maintained in the storage elements can only change when the clock pulses.
- Hence, a flip-flop maintains a binary state until directed by a clock pulse to switch states.
- The difference in the types of flip flops is in the number of inputs and the manner in which the inputs affect the binary state.
- Flip-flops can be described by a *characteristic table* which permutates all possible inputs (just like a truth table).
- The characteristic table of a flip-flop describes all possible outputs (called the *next state*) at time *Q*(*t*+1) over all possible inputs and the *present state* at time *Q*(*t*).
- The most common types of flip flops are:
  - SR Flip-Flop
  - D Flip-Flop
  - JK Flip-Flop
  - T Flip-Flop

SR Flip-Flop

Figure SR Flip-Flop



(a) Graphic symbol



S	R	Q (t + 1)	
0	0	Q (t)	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	?	Indeterminate

(b) Characteristic table

Inputs:
- S (for set)
- R (for reset)
- C (for clock)

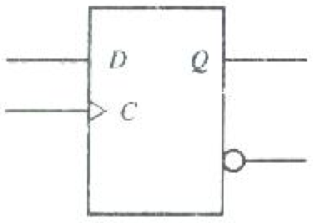
Outputs:
- Q
- Q'

The operation of the SR flip-flop is as follow.

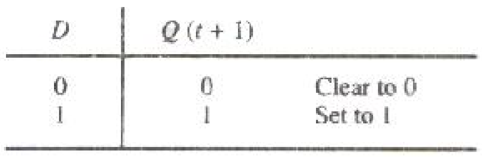
- If there is no signal at the clock input C, the output of the circuit cannot change irrespective of the values at inputs S and R.
- Only when the clock signals changes from 0 to 1 can the output be affected according to the values in inputs S and R
- If S =1 and R = 0 when C changes when C changes from 0 to 1 output Q is set to 1. If S = 0 and R =1 when C changes from 0 to 1.
- If both S and R are 0 during the clock transition, output does not change.
- When both S and R are equal to 1, the output is unpredictable and may go to either 0 or 1, depending on internal timing that occur within the circuit.

D Flip-Flop

D Flip-flop



(a) Graphic symbol



D	Q (t + 1)	
0	0	Clear to 0
1	1	Set to 1

(b) Characteristic table

Inputs:
- D (for data)
- C (for clock)

Outputs:
- Q
- Q'

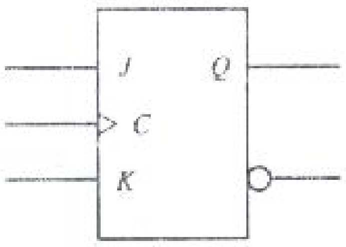
The operation of the D flip-flop is as follow.

- The D Flip-Flop can be converted from SR Flip-Flop by inserting an inverter between S and R and assigning the symbol D to the single input.
- The D input is sampled during the occurrence of a clock transition from 0 to 1.
- If D=1, the output of the flip-flop goes to the 1 state, but if D=0, the output of the flip-flop goes to the 0 state.
- The next state Q(t+1) is determined from the D input. The relationship can be expressed by a characteristic equation:

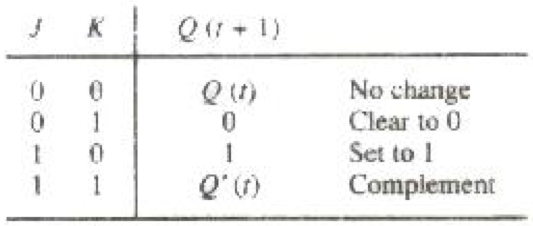
  $$Q(t+1) = D$$
- D Flip-Flop has the advantage of having only one input (excluding), but the disadvantage that its characteristic table does not have a "no change" condition Q(t+1) = Q(t).

JK Flip-Flop

Jk Flip-Flop



(a) Graphic symbol



J	K	Q (t + 1)	
0	0	Q (t)	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	Q' (t)	Complement

(b) Characteristic table

Inputs:
- J
- K
- C (for clock)

Outputs:

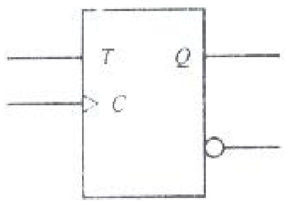
- Q
- Q'

The operation of the JK flip-flop is as follow.

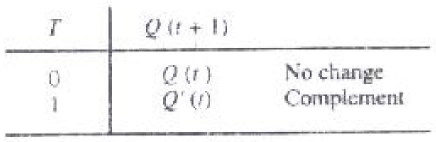
- A JK Flip-Flop is a refinement of the SR flip-flop in that the indeterminate condition of the SR type is defined in the JK type.
- Inputs J and K behave like inputs S and R to set and clear the flip-flop, respectively.
- When inputs J and K are both equal to 1, a clock transition switches the outputs of the flip-flop to their complement state.
- Instead of the indeterminate condition of the SR flip-flop, the JK flip-flop has a complement condition Q(t+1) = Q'(t) when both J and K are equal to 1.

T Flip-Flop

T Flip-Flop



(a) Graphic symbol



T	$Q(t+1)$	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

(b) Characteristic table

Inputs:

- T (for toggle)
- C (for clock)

Outputs:

- Q
- Q'

The operation of the T flip-flop is as follow.

- Most flip-flops are edge-triggered flip-flops, which means that the transition occurs at a specific level of the clock pulse.
- A positive-edge transition occurs on the rising edge of the clock signal.
- A negative-edge transition occurs on the falling edge of the clock signal.
- Another type of flip-flop is called a master-slave flip-flop that is basically two flip-flops in series.
- Flip-flops can also include special input terminals for setting or clearing the flip-flop *asynchronously*. These inputs are usually called *preset* and *clear* and are useful for initialing the flip-flops before *clocked* operations are initiated.

Flip-Flop Excitation Tables

- During the design of sequential circuits, the required transition from present state to next state is known.
- What the designer needs to know is what input conditions must exist to implement the required transition.
- This requires the use of flip-flop excitation tables.

Excitation Tables

SR Flip-Flop Excitation Table			
Q(t)	Q(t+1)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

JK Flip-Flop Excitation Table			
Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

T Flip-Flop Excitation Table		
Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Sequential Circuits

- When a circuit contains just gates, it is called a *combinational* circuit. However, if a circuit uses both gates and flip-flops, it is called a *sequential* circuit. Hence, a sequential circuit is an interconnection of flip-flops and gates.
- If we think of a sequential circuit as some *black box* that, when provided with some *external* input, produces some *external* output, a typical sequential circuit would function as follows:
- The *external* inputs constitute *some* of the inputs to the combinational circuit. The *internal* outputs of the combinational circuit are the *internal* inputs to the flip-flops.
- The *internal* outputs of the flip-flops constitute the remaining inputs to the combinational circuit. The *external* outputs are some combination of the outputs from the combinational circuit and flip-flops. The behavior of a sequential circuit is determined from the inputs, the outputs, and the state of the flip-flops. Both the outputs and the next state are determined by the inputs and the present state.
- A state diagram can represent the information in a state table graphically, where states are represented by circles (vertices) and transitions on specific input is represented by the *labels* on the directed lines (edges) connecting the circles.

Design Procedure

- Formulate behavior of circuit using a state diagram.
- Determine # of flip-flops needed (equal to # bits in circles).
- Determine # inputs (specified on edges of diagram).
- Create state table, assigning letters to flip-flips, input, and output variables.*
- For each row, list the next state as specified by the state diagram.
- Select flip-flop type to be used in circuit.
- Extend state table into an excitation table by including columns for each input of each flip-flop.
- Using excitation table and present state-to-next state transitions, formulate input conditions for flip-flops.
- Construct truth table for combinational circuit using present-state and input columns of excitation table (for inputs) and flip-flop inputs (for outputs).
- Use map simplification of truth table to obtain flip-flop input equations.**
- Determine *external* outputs of sequential circuit (flip-flop outputs and potentially combinational circuit outputs).
- Draw logic diagram as follows:

- Draw flip-flops and label all their inputs and outputs.
- Draw combinational circuit from the Boolean expressions given by the flip-flop input equations.
- Connect outputs of flip-flops to inputs in the combinational circuit.
- Connect outputs of combinational circuit to flip-flop inputs.

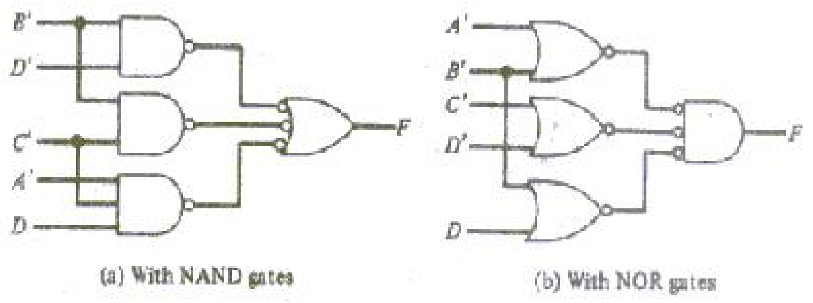
For *m* flip-flops and *n* inputs, the state table will consist of *m* columns for the present state, *n* columns for the inputs, and *m* columns for the next state. The number of rows in the table will be up to 2*m*+*n*, one row for each binary combination of present state and inputs.

** Each flip-flop input equation specifies a logic diagram whose output must be connected to one of the flip-flop inputs.

NAND and NOR Implementation

A sum-of-products expression can be implemented with NAND and NOR gates as shown in the figure 2.9

Figure 2.9 Logic Diagram with NAND and NOR gates

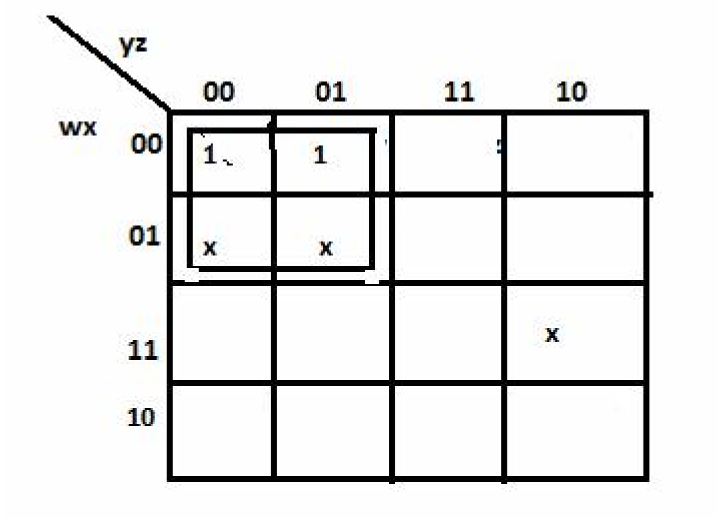


Don't Care Conditions

- In k-map each cell represents a minterm or maxterm and the 0's and 1's in k map represents the minterm that make the function equal to either 0 or 1.

- But in some occasion, it doesn't matter whether a function produces a 0 or 1 for a given minterm.
- When this condition occurs, an X is used in the map to represent the *don't care* condition.
- The minterm that may produce either 0 or 1 for function are said to be Don't Care and marked as x in map.
- This don't care condition are used to further simplify the Boolean expression.
- Don't care condition is the condition where any single square or map will appear as x n it is not necessary to write into Boolean expression.

Example
F(w,x,y,z)= ∑(0,1) + d(4,5,14)

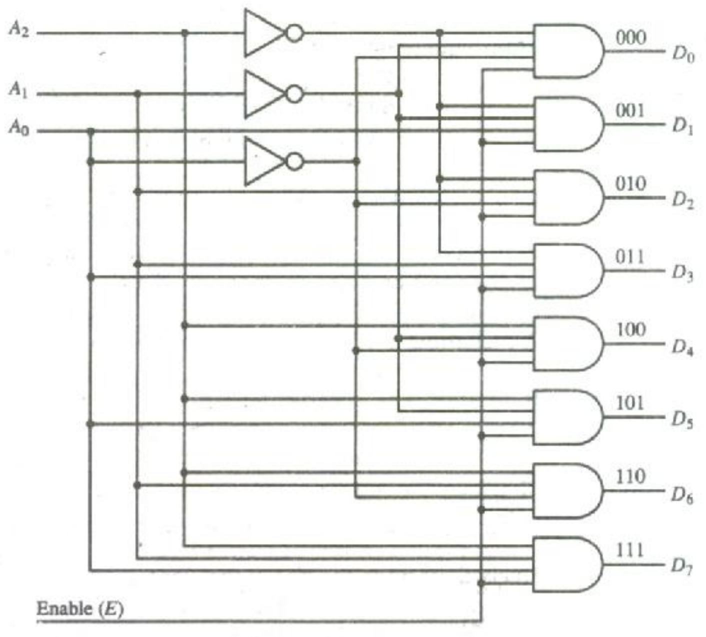


- So the ans is **W'Y'**

UNIT 2: DIGITAL COMPONENTS

What is Decoder?

- Discrete quantities of information are represented in digital computers with binary codes.
- A binary code of n bits is capable of representing up to 2n distinct elements of the coded information.
- A decoder is a combinational circuit that converts binary information from the n coded inputs to a maximum of 2n unique outputs.
- If the n-bit coded information has unused bit combinations, the decoder may have less than 2n outputs.
- The decoders presented in this section are called n-to-m-line decoders, where m <= 2n. Their purpose is to generate the 2n (or fewer) binary combinations of the n input variables. A decoder has n inputs and m outputs and is also referred to as an n x m decoder.
- The logic diagram of a 3-to-8-line decoder is shown bellow.



- The three data inputs. A0, A1, and A2, are decoded into eight outputs, each output representing one of the combinations of the three binary input variables.
- The three inverters provide the complement of the inputs, and each of the eight AND gates generates one of the binary combination.
- A particular application of this decoder is a binary-to-octal conversion. The input variables represent a binary number and the outputs represent the eight digits of the octal number system.
- However, a 3-to-8-line decoder can be used for decoding any 3-bit code to provide eight outputs, one for each combination of the binary code.
- Commercial decoders include one or more enable inputs to control the operation of the circuit. The decoder of the Figure has one enable input, E.
- The decoder is enabled when E is equal to 1 and disabled when E is equal to 0. The operation of the decoder can be clarified using the truth table listed in Table.
- When the enable input E is equal to 0, all the outputs are equal to 0 regardless of the values of the other three data inputs.

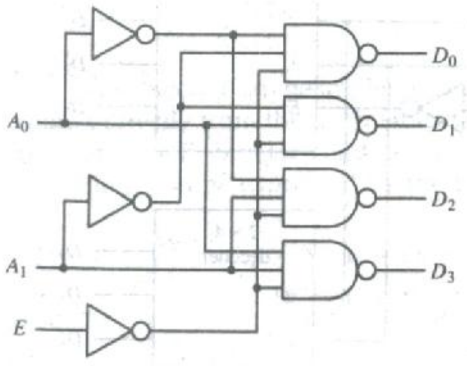
- The three x's in the table designate don't-care conditions. When the enable input is equal to 1, the decoder operates in a normal fashion.
- For each possible input combination, there are seven outputs that are equal to 0 and only one that is equal to 1.
- The output variable whose value is equal to 1 represents the octal number equivalent of the binary number that is available in the input data lines.

Truth Table for 3-to-8-line Decoder

Enable	Inputs			Outputs							
E	A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	×	×	×	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

- **NAND Gate Decoder**

- Some decoders are constructed with NAND instead of AND gates. Since a NAND gate produces the AND operation with an inverted output, it becomes more economical to generate the decoder outputs in their complement form.
- A 2-to-4-line decoder with an enable input constructed with NAND gates is shown in Figure.
- The circuit operates with complemented outputs and a complemented enable input E. The decoder is enabled when E is equal to 0. As indicated by the truth table, only one output is equal to 0 at any given time; the other three outputs are equal to 1.
- The output whose value is equal to 0 represents the equivalent binary number in inputs Ai and Ao.
- The circuit is disabled when E is equal to 1, regardless of the values of the other two inputs.



(a) Logic diagram

E	A_1	A_0	D_0	D_1	D_2	D_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	×	×	1	1	1	1

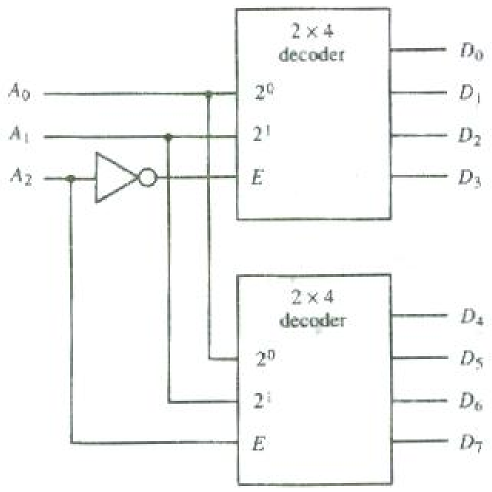
(b) Truth table

- **Decoder Expansion**

- A technique called decoder expansion can be utilized to construct larger decoders out of smaller ones.

- For example, two 2-to-4-line decoders can be combined to construct a 3-to-8-line decoder. Figure below shows 3-8-line decoder constructed with two 2x4 decoders.

3X8 decoder constructed with two 2X4 decoders



- The above given Figure shows how the decoders with enable inputs can be connected to form a larger decoder.
- As you can see that there are two 2-to-4-line decoders are combined to achieve a 3-to-8-line decoder.
- The two least significant bits of the input are connected to both decoders.
- The most significant bit is connected to the enable input of one decoder and through an inverter to the enable input of the other decoder.
- It is assumed that each decoder is enabled when its E input is equal to 1. When E is equal to 0, the decoder is disabled and all its outputs are in the 0 level. When A2 = 0, the upper decoder is enabled and the lower is disabled.
- The lower decoder outputs become inactive with all outputs at 0. The outputs of the upper decoder generate outputs D0 through D3, depending on the values of A1 and A0(while A2 = 0).
- When A2= 1, the lower decoder is enabled and the upper is disabled. The lower decoder output generates the binary equivalent D4, through D7 since these binary numbers have a 1 in the A2 position.

What is Encoder?

- An encoder is a digital circuit that performs the inverse operation of a decoder. An encoder has 2^n (or less) input lines and n output lines.
- The output lines 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 below.

Inputs								outputs		
D7	D6	D5	D4	D3	D2	D1	D0	A2	A1	A0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

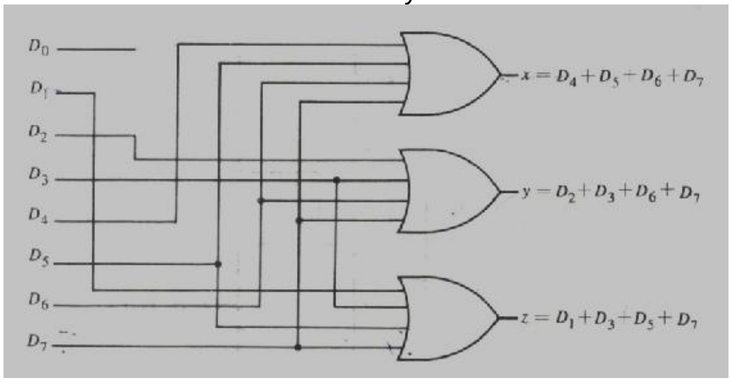
- The encoder can be implemented with OR gates whose inputs are determined directly from the truth table.
- Output A0 =1 if the input octal digit is 1 or 3 or 5 or 7. Similar conditions apply for other two outputs.

These conditions can be expressed by the following Boolean functions :

$$A0 = D1 + D3 + D5 + D7$$
$$A1 = D2 + D4 + D6 + D7$$
$$A2 = D4 + D5 + D6 + D7$$

The encoder can be implemented with three OR gates.

Octal-to-binary Encoder


D_0
D_1
D_2
D_3
D_4
D_5
D_6
D_7
$x = D_4 + D_5 + D_6 + D_7$
$y = D_2 + D_3 + D_6 + D_7$
$z = D_1 + D_3 + D_5 + D_7$

Write a note on Multiplexer

- A multiplexer is a combinational circuit that receives binary information from one of 2n input data lines and directs it to a single output line.
- The selection of a particular input data line for the output is determined by a set of selection inputs. A 2n-to-1 multiplexer has 2n input data lines and n input selection lines whose bit combinations determine which input data are selected for the output.
- The 4-to-1 line multiplexer has six inputs and one output. A truth table describing the circuit needs 64 rows since six input variables can have 26 binary combinations. This is

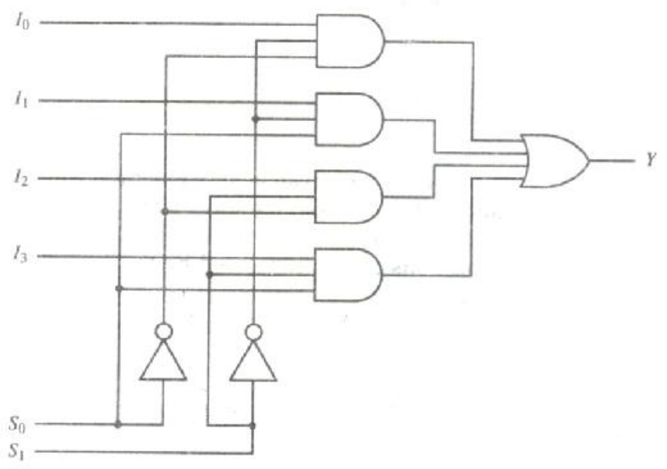
an extremely long table and will not be shown here. A more convenient way to describe the operation of multiplexers is by means of a function table.

- The function table for the multiplexer is shown in table.
- The table demonstrates the relationship between the four data inputs and the single output as a function of the selection inputs S1 and S0.

Function table for 4-to-1 line multiplexer

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

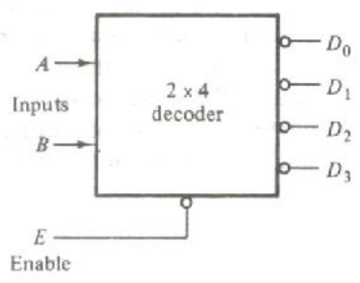
4-to-1 line Multiplexer



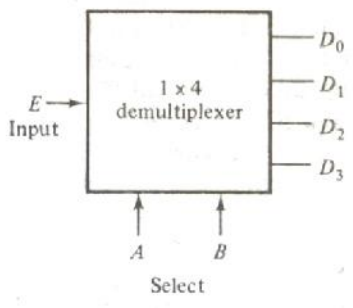
Write a note on DeMultiplexer

- A decoder with an enable input can function as a demultiplexer.
- A demultiplexer is a circuit that receives information on a single line and transmits this information on one of 2n possible output lines.
- The selection of a specific output line is controlled by the bit values of n selection lines. The decoder of figure a can function as a demultiplexer if the E line is taken as a data input line and lines A and B are taken as the selection lines.
- The single input variable E has a path to all four outputs, but the input information is directed to only one of the output lines, as specified by the binary value of the two selection lines A and B.
- For example, if the selection lines AB = 10; output D2 will be the same as the input value E, while all other outputs are maintained at 1.
- Because decoder and demultiplexer operations are obtained from the same circuit, a decoder with an enable input is referred to as a decoder / demultiplexer.
- It is the enable input that makes the circuit a demultiplexer.

Block Diagram for DeMultiplexer



(a) Decoder with enable



(b) Demultiplexer

Truth Table Of Demultiplexer

Input					Output	
D	**S0**	**S1**	**F0**	**F1**	**F2**	**F3**
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Explain MUX in detail.

- MUX is combinational circuit that is used to direct one out of 2n input data Lines to A single output line.
- It is also known as data selector because it selects one of many inputs And directs it to the output.
- The selection of particular input data line is controlled by as Set of selection inputs.
- Normally there are 2n input data lines and n input selection lines.

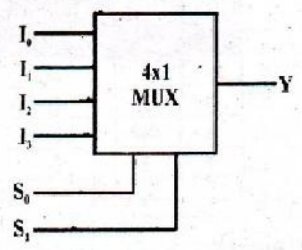
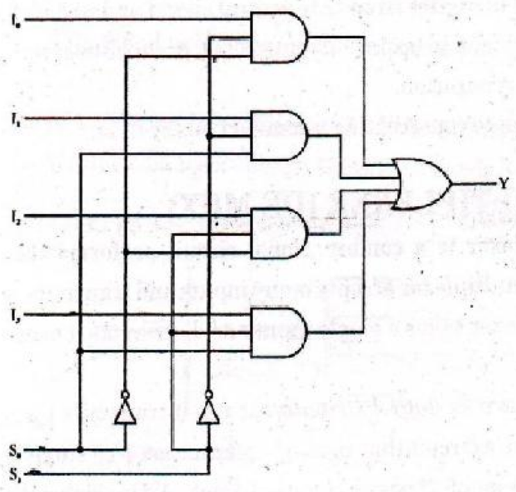


Fig. 3.4 : Block Diagram of 4 x 1 MUX



- The Block diagram of 4 to 1 line MUX is shown in fig. and logic circuit is displayed in fig.
- Each of the 4 data input I0 through I3 is applied to one input of AND gate.
- The two selection inputs s1 and s0 are decoded To select a particular AND gate.
- The output of the AND gates are applied to a single OR gate to provide the single output.

simplification of Multiplexer.

- It is used to connecting two or more sources to a single destination among computer units.
- It is used in digital circuits to control signal and in data routing.
- It is also useful in operation sequencing.
- It is useful to constructing a common bus system.

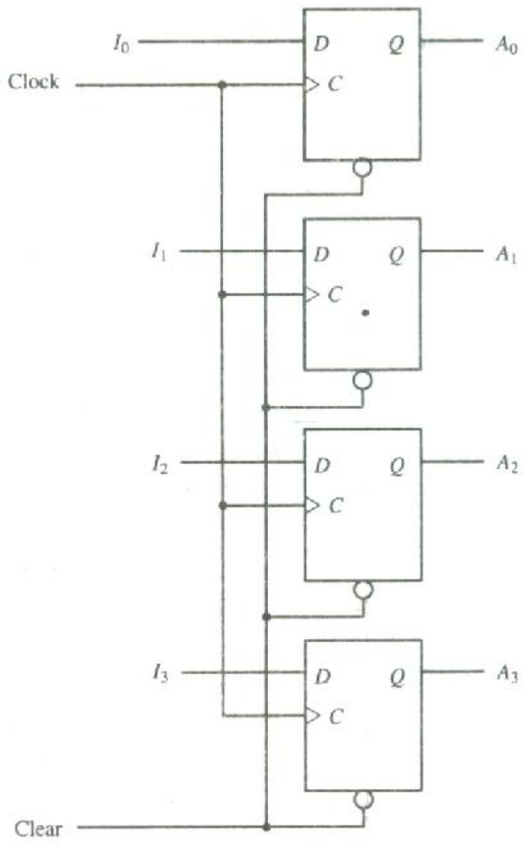
What are Registers? Explain the types of registers.

- A *register* is a group of flip-flops capable of storing one bit of information.
- An *n*-bit register has a group of *n* flip-flops and is capable of storing any binary information of *n* bits.
- In addition to flip-flops, registers can have combinational gates that perform certain data-processing tasks. The gates control how and when new information is transferred into the registers.
- The transfer of new information into a register is referred to as a *register load*. If the loading occurs simultaneously at a common clock pulse transition, we say that the load is done in *parallel*.
- The *load input* in a register determines the action to be taken with each clock pulse.

- When the load input is 1, the data from the input lines is transferred into the register's flip-flops. When the load input is 0, the data inputs are *inhibited* and the flip-flop maintains its present state.

  A 4-bit register is shown in the figure below. A clock transition applied to the C inputs of the register will load all four inputs I_0 through I_3 in parallel.

Figure 5.1 4-bit register



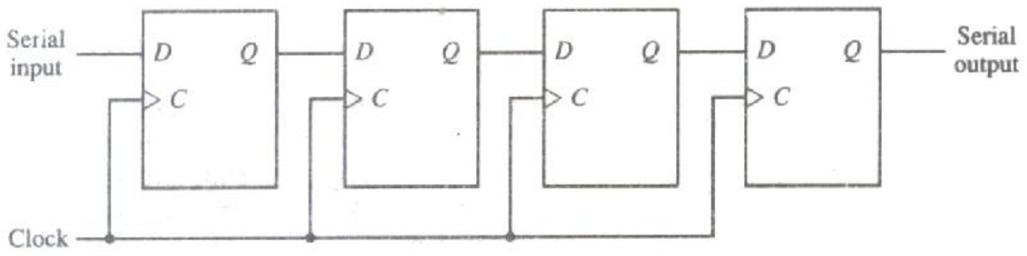
Shift Registers

- A *register* capable of shifting its binary information in one or both directions is called a *shift register*.
- Shift registers are constructed by connecting flip-flops in *cascade*, where the output of one flip-flop is connected to the input of the next flip-flop.
- All flip-flops receive common clock pulses that initiate the shift from one stage to the next.
- A *serial input* shift register has a single external input (called the serial input) entering an outermost flip-flop. Each remaining flip-flop uses the output of the previous flip-flop as its input, with the last flip-flop producing the external output (called the serial output).
- A register capable of shifting in one direction is called a *unidirectional* shift register.
- A register that can shift in both directions is called a *bi-directional* shift register.
- The most general shift register has the following capabilities:
  - An input for clock pulses to synchronize all operations.
  - A shift-right operation and a serial input line associated with the shift-right.
  - A shift-left operation and a serial input line associated with the shift-left.
  - A parallel load operation and *n* input lines associated with the parallel transfer.

- N parallel output lines.
- A control state that leaves the information in the register unchanged even though clock pulses are applied continuously.
- A mode control to determine which type of register operation to perform.

The simplest possible shift register is one that uses only flip-flops, as shown in the figure below.

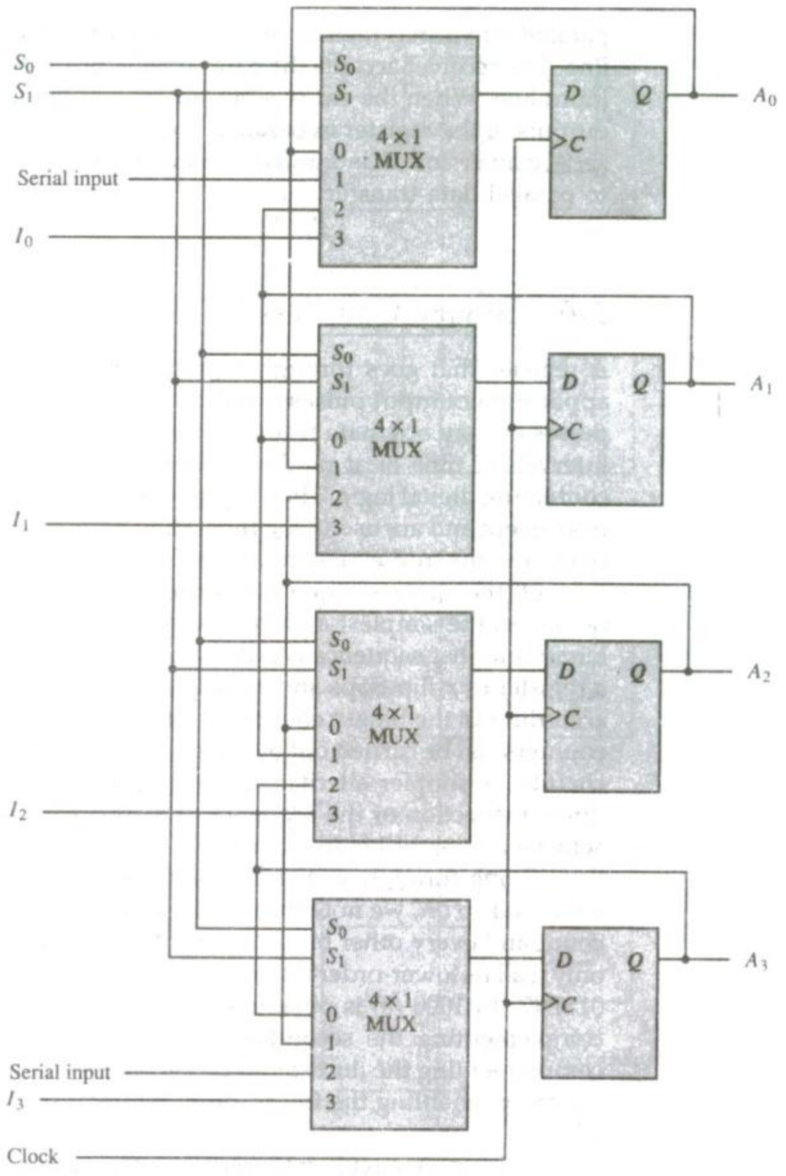
4-bit shift register



Bidirectional Shift Register with parallel load

- A register that can shift in both directions is called a bi-directional shift register.
- A 4-bit bidirectional shift register with parallel load is shown in figure below. Each stage consists of a D flip-flop and a 4X1 MUX.
- The two selection inputs S1 and S0 select one of the MUX data inputs for the D flip-flop. The selection lines control the mode of operation of the register according to the function table shown in table below.
- When the mode control S1S0 = 00, data input 0 of each MUX is selected.
- This condition forms a path from the output of each flip-flop into the input of the same flip-flop.
- The next clock transition refers into each flip-flop the binary value it held previously, and no change of state occurs. When S1S0 = 01, the terminal marked 1 in each MUX has a path to the D input of the corresponding flip-flop.
- This causes a shift-right operation, with the serial input data transferred into flip-flop A0 and the content of each flip-flop Ai-1 transferred into flip-flop Ai for i=1,2,3. When S1S0 = 10 a shift-left operations results, with the other serial input data going into flip-flop A3 and the content of flip-flop Ai+1 transferred into flip-flop Ai for I=0,1,2. When S1S0 = 11, the binary information from each input I0 through I3 is transferred into the corresponding flip-flop, resulting in a parallel load operation.
- In the diagram, the shift-right operation shifts the contents of the register in the down direction while the shift left operation causes the contents of the register to shift in the upward direction.

Bidirectional Shift register with parallel load



Application of Shift Registers

- Shift registers are often used to interface digital systems situated remotely from each other. For example, suppose that it is necessary to transmit an n-bit quantity between two points.
- If the distance between the source and the destination is too far, it will be expensive to use n lines to transmit the n bits in parallel.
- It may be more economical to use a single line and transmit the information serially one bit at a time.
- The transmitter loads the n-bit data in parallel into a shift register and then transmits the data from the serial output line.
- The receiver accepts the data serially into a shift register through its serial input line. When the entire n bits are accumulated they can be taken from the outputs of the register in parallel.
- Thus the transmitter performs a parallel-to-serial conversion of data and the receiver converts the incoming serial data back to parallel data transfer.

UNIT 3: DATA REPRESENTATION

What is Number System?

- A number system is a set of numbers, together with one or more operations such as addition or multiplication.
- There are mainly two types of number system
  - Positional number system
  - Non positional number system
- **NON POSITIONAL NUMBER SYSTEM**
  - In this system each symbol represents the same value so it is difficult to perform arithmetic operation with such numbers.
- **POSITIONAL NUMBER SYSTEM**
  - In this system each number is identified by the position where it is placed. It means that the value of number is depended on the position. In this kind of system it is possible to perform the arithmetic operation very easily.
  - There are mainly 4 types of positional number systems are commonly used.
    - Binary
    - Octal
    - Decimal
    - HexaDecimal
- **Binary Number System**
  - This number system is used in computers or digital systems
  - 1.
  - Each digit enter in the system is represented in the form of 0 and 1s
- **Octal Number Systems**
  - in this system there are 8 different characters are used to represent the numbers.
  - We can use 0 to 7 different symbols are used.
  - In octal number system base is 8
  - In this number system values increase from right to left as 1, 8, 64, 512, 4096..
- **Decimal Number System**
  - in this system there are 10 different characters are used to represent the numbers.
  - We can use 0,1,2,3,4,5,6,7,8,9 different symbols are used.
  - In octal number system base is 10.
  - In this number system each positions of number is given depending on weight.
  - For example: 4123

    $=4 * 10^3 + 1 * 10^2 + 2 * 10^1 + 3 * 10^0$
    $=4 * 1000 + 1 * 100 + 2 * 10 + 3 * 1$
    $=4000+100+20+3$
    $=4123$
- **HexaDecimal Number System**
  - This system is very useful in microprocessor
  - There are 16 different characters are used to represent the numbers.
  - We can use 0,1,2,3,4,5,6,7,8,9 and A,B,C,D,E,F different symbols are used.
  - The values are increases from right to left of hexadecimal points as 1, 16, 256, 65536 and so on.

Conversions

Converting Decimal To Binary

- In conversion from decimal to any other number system, the steps to be followed are:
- Divide the decimal number by the base of 2.
- Note the remainder in one column and divide the quotient again with the base. Repeat this process until the quotient is reduced to a zero.
- Example:
- **The decimal number is 65**

2	**65**	**1**
2	**32**	**0**
2	**16**	**0**
2	**8**	**0**
2	**4**	**0**
2	**2**	**0**
	1	

- **The binary number of 65 is 1000001**

Converting Binary To Decimal

- The decimal number of 100001 is
- = (1*26)+(0*25)+(0*24)+(0*23)+(0*22)+(0*21)+(1*20)
- = (1*64) + (0*32) + (0*16) + (0*8) + (0*4) + (0*2) + (1 * 1)
- = 64 + 0 + 0 + 0 + 0 + 0 + 1
- = 65
- The decimal number of 1000001 is 65

Octal Number System (Base 8)

- A commonly used positional system is the Octal System. The octal system has a base of 8.
- The values increase from left to right as 1, 8, 64, 512, 4096,… .

Converting Decimal To Octal

- In conversion from decimal to any other number system, the steps to be followed are:
- Divide the decimal number by the base of the 8.

Example:

- **The decimal number is 224**

8	**224**	**0**
8	**28**	**4**
8	**3**	**3**

- **The octal number of 224 is 340**

Converting Octal To Decimal

- The octal number is 340
- = (3*82)+(4*81)+(0*80)
- = (3*64) + (4*8) + (0*1)
- = 192 + 32 + 0
- = 224
- The decimal number of 340 is 224

Converting Binary Octal

000	**0**
001	**1**
010	**2**
011	3
100	4
101	5
110	6
111	7

Converting from Binary to Octal

- The binary number must be divided into groups of three from the octal point – to the right in case of the fractional portion and to the left in case of the integer portion. Each group can then be replaced with their octal equivalent.
- Example
- Binary 101010101010100
- 101 010 101 010 100
- 5 2 5 2 4
- So, 52524 is the Octal equivalent of binary 101010101010100

Converting Octal to Binary

- Each octal digit is replaced with the appropriate 'triple' of binary digits.
- e.g. 65
- 5
- 110 101
- The binary equivalent of the Octal number 65 is 110101

Hexadecimal Number System (Base 16)

- There is another commonly used positional system, hexadecimal system.
- The hexadecimal system has a base of 16, so the value increases from left to right as 1, 16, 256, 65536,. . . .
- We need to keep a simple table in mind before we attempt any conversion from hexadecimal or vice-versa.

Converting Decimal To HexaDecimal

- In conversion from decimal to any other number system, the steps to be followed are:
- Divide the decimal number by the base of 16.

Example:

- **The decimal number is 370**

16	**370**	**2**
16	**23**	**7**
16	**1**	**1**
	0	

- **The hexadecimal number of 370 is 172**

Converting Hexadecimal To Decimal

- The hexadecimal number 172
- = (1*162) + (7*161) + (2*160)
- = (1*256) + (7*16) + (2*1)
- = 256+112+2
- = 370

- The decimal number of 172 is 370

Converting Binary to Hexadecimal

- Each hexadecimal digit is represented by 4 binary digits.

  Binary Hexadecimal

0000	**0**
0001	**1**
0010	**2**
0011	**3**
0100	**4**
0101	**5**
0110	**6**
0111	**7**
1000	**8**
1001	**9**
1010	**A**
1011	**B**
1100	**C**
1101	**D**
1110	**E**
1111	**F**

- To convert a binary number to its hexadecimal equivalent we split the quantity into groups of four onwards, as before.
- Each of this group of four is directly converted into their hexadecimal equivalent.
- We may add zeros to the left of the number if necessary.
- example
- Binary 10101011000010
- 0010 1010 1100 0010
- 2 A C 2
- So, the hexadecimal equivalent of binary 10101011000010 will be 2AC2

Converting Hexadecimal to Binary

- The conversion from hexadecimal to binary consists of writing off the binary
- Equivalent of each hexadecimal digit in groups of four.
- e.g.
- Hexadecimal 1901A0412C
- 0001 1001 0000 0001 1010 0000 0100 0001 0010 1100
- 1 9 0 1 A 0 4 1 2 C
- Thus the required binary number can be written as:
- 1100100000001101000000100000100101100

Converting Hexadecimal to Octal

- Write 4 digit binary numbers for each hexadecimal.
- Arrange the entire number sequence into group of 3 bit section.
- If any bit is missing add 0 on leftmost section.
- Now write down octal equivalent of each 3 bit section.
- Example:
- The hexadecimal number is 2A35.
- 2 A 3 5
- 0010 1010 0011 0101

  000 010 101 000 110 101

- 0 2 5 0 6 5
- The octal number is 25065

Converting Octal to Hexadecimal

- Write 3 digit binary number for each octal.
- Arrange the entire number sequence into group of 4 bit section.
- If any bit is missing add 0 on leftmost section.
- Now write down hexadecimal equivalent of each 4 bit section.
- Example:
- The octal number is 25065.
- 2 5 0 6 5
- 010 101 000 110 101
- 0010 1010 0011 0101
- 2 A 3 5
- The hexadecimal number is 2A35

Binary Arithmetic

- All the arithmetic operations are possible in binary numbering system like addition, subtraction, multiplication and division All the Arithmetic operations are done in binary number system are explained as under:

Addition

- For binary addition the following rules of binary addition are to be considered:

input		Sum of A + B	Carry
A	B		
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- e.g. 1 Add two binary numbers 11011 and 111
- **Carry** **1 1 1 1**

```
  1 0 1 1 1
    + 1 1 1
-----------
1 0 0 0 1 0 (Answer)
```

Subtraction

- Binary substraction is just similar as we perform substraction on decimal numbers.
- Though there are other methods of performing subtraction, we will consider the method of subtraction know as complementary subtraction.
- To perform substraction we must have to follow the following rules:

input		**Subtraction of A B**	**borrow**
A	**B**		
0	**0**	**0**	**0**
0	**1**	**1**	**1**
1	**0**	**1**	**0**
1	**1**	**0**	**0**

Multiplication

- Multiplication in binary follows the same rules that are followed in the decimal system. The table to be remembered is:

input		Multiplication of A * B
A	B	
0	0	0
0	1	0
1	0	0
1	1	1

- **e.g. : 1010 * 1001**

```
     1010
   x 1001
   ------
     1010
    0000
   0000
  1010
  -------
 101101 0
```

- **The answer is (1011010)**

Division

Table for binary division is given as under:

input		Division of A / B
A	B	
0	0	Undefined
0	1	0
1	0	Undefined
1	1	1

- The steps for binary division are:
- Start from the left of the dividend.
- Perform subtraction in which the divisor is subtracted from the dividend
- If subtraction is possible put a 1 in the quotient and subtract the divisor from the corresponding digits of the dividend else put a 0 in the quotient
- Bring down the next digit to the right of the remainder.
- Execute step 2 till there are no more digits left to strating down from the dividend.

- e.g.
- 100001 / 110

```
        0101        (quotient)
110   | 100001
      |   110
      |  ------
          1000
           110
         ------
           100
           110
         -------
            1001
             110
         ---------
              11      (reminder)
```

Write a note on Floating Point Representation.

- A number which has both an integer part as well as fractional part is called Real or Floating point Number.
- For e.g. 2.365, 78.738789, 6.5643 etc are the examples of floating point numbers.
- The floating point representation of numbers has two pars.

sign	Fixed point Mantissa	Exponent

- The first part represents a sign fixed point number called **Mantissa.**
- The second part designated the position of decimal or binary point; this part is known as **exponent.**
- The fixed point mantissa may be integer or fractional.
- For example : the decimal no +6132.789 is represent in floating point with exponent and fraction as follows:

sign	Fixed point Mantissa	Exponent
+	0.6132789	04

- The value of exponent indicates that the actual position of decimal point is 4 positions to the right of the indicated decimal point in the fraction.
- We can also represent this number as +0.6132789 × 10^4 like scientific notation.
- In general floating point is represented in the form of **$m \times r^e$.**
- where
  m = mantissa
  r = radix
  e = exponent.

- In above example 6132789 is mantissa, 4 is exponent and 10 is radix.

Write a note on Error Detection Code. OR what do you mean by parity? Explain it with error detection code.

- An error detection code is a binary code that detects error during information transmission.
- The error detection code can not be correct error but gives just indication that the error is presents.
- There are various techniques to detect error but the most common is parity bit.

Parity Bit

- A parity bit is an extra bit included with a binary message to make the total number of 1's either odd or even.
- Generally there are 2 techniques even and odd parity is used.
- In even parity bit the sum of all 1's is even while in odd parity the sum of all 1's is odd.
- For e.g. for three bit message has two possible parity shown in bellow table.

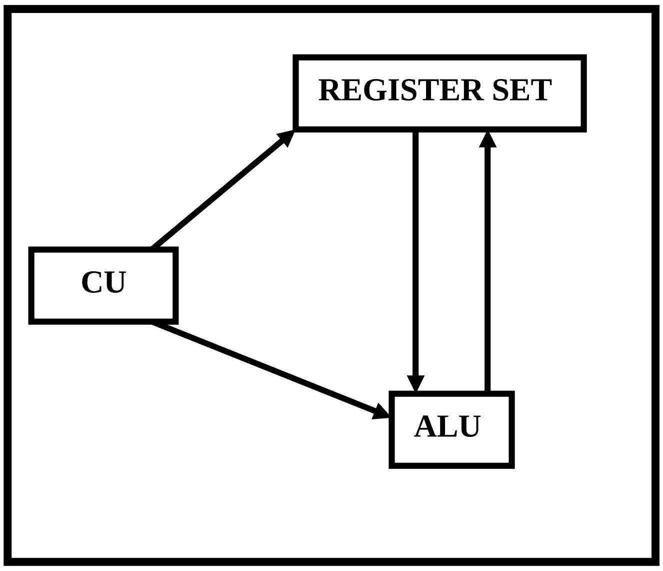
Message Xyx	P(odd)	P(even)
000	1	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	0	1

- During transformation of information from one location to another parity bit is handle as follows:
    - At sending end, the message transferred to parity generator.
    - The parity generator generates required parity.
    - Then the message transferred to destination.
    - At destination the parity checker checks the proper parity adopted.
    - An error is detected if the checked parity doesn't confirm to the adopted parity.
- The parity generator & checker network are logic circuits constructs with Exclusive or gate.

UNIT 4: CENTRAL PROCESSING UNIT

Write a note on Central Processing Unit.

- The central processing unit is the brain of digital computer.
- It performs various types of function depending on the instruction that incorporated in the computer.
- The main function of the central processing unit is to execute the programs and generates the results, except that it also controls the internal and external parts of the computers.



- The control unit, register sets and ALU are the major three components of central processing unit.
- The main memory is also a part of central processing unit.
- The CPU of small computer contains a single microprocessor while the CPU of large computer may contain multiple microprocessors.
- The microprocessor contains two major parts that is CU – Control Unit and ALU – Arithmetic and Logical Unit.
- **Control Unit:** it controls the entire operations of the system. It also controls the input and output devices.
- **Arithmetic & Logical Unit:** it performs the mathematical as well as logical operations on the instruction. This unit is responsible to generate the output on inputted instructions.
- **Register sets:** the register sets contain more than one registers. The registers are basically used for storing immediate results during the process of executing instruction. Various registers performs various task.

Explain General Register Organization.

- The set of registers in a computer are connected to the ALU using busses and Multiplexers.
- A 14-bit control word specifies two source registers (SELA & SELB), a destination register (SELD), and an operation (OPR).
- The registers can be specified using three bits each as follows:

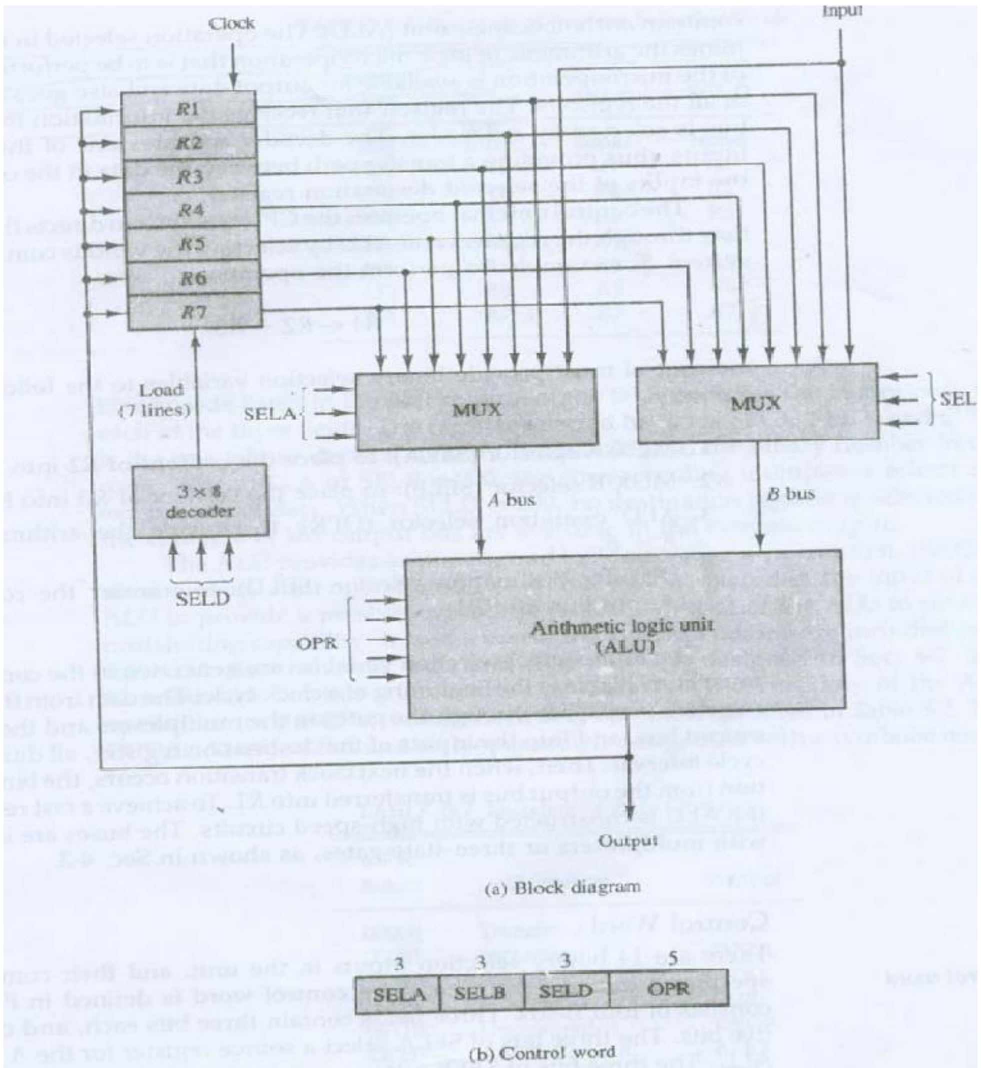


Figure 8-2 Register set with common ALU.

- A bus organization for seven CPU registers is shown in fig.
- The output of each Register Connected to two multiplexers to perform the two buses A and B.
- The selection lines in each multiplexer select one register of the input data for the particular logic unit (ALU).
- The operation selected in the ALU determines the arithmetic or logic micro operations that are to be performed.
- The result of the micro-operation is available for output data and also goes into the inputs of all the registers.
- The register that receives the information from the output bus is selected the decoder.

- The decoder activates one of the register load inputs, thus providing a transfer Path between the data in the output bus and the inputs of the selected destination register.
- The control unit that operates the CPU bus system directs the information flow through the registers and ALU selecting the various components in the system.

SHORT NOTE: CONTROL WORD

- There are 14 – bit binary selection input in the unit and their combine value that specify a control word.
- The figure of 14 – bit combine word is as given below:

SELECTION A	SELECTION B	SELECTION D	OPR
3 bit	3 bit	3 Bit	5 bit

- In this 14 – bit combine word, first 3 columns contains 3 bits and last column contain 5 bits.
- The first field is a SELECTION A that contains 3 bits
- The second field is a SELECTION B that contains 3 bits
- The third field is a SELECTION D that contains 3 bits
- The last field is a OPR that contains 5 bits
- The 3 bits of SELECTION – A, SELECTION – B and SELECTION – D that selects a source register for the input of the ALU
- The 5 bits of OPR that selects a destination register for output of the ALU

WHAT IS INTERRUPT? EXPLAIN IN DETAIL.

- The concept of program interrupt is use to handle variety of problems that arrives out of normal program sequence.
- The interrupt facility is useful in multi programming Environment when two or more program resided in the memory at same time.
- The function of interrupt facility is to take care of data transfer of one or more program while another program is currently being executed.

TYPES OF INTERRUPT :

- There are three major type of interrupt.
  - **External interrupt :**
  - **Internal interrupt:**
  - **Software interrupt**

- **External interrupt :**

  - The external interrupt caused by external events, they comes from Input output devices, timing devices, timing devices of power supply or any other External sources.
  - The external interrupt are asynchronous and they are independent of The program being executed at the time.

- **Internal interrupt:**

   - Internal interrupt is generated by some exceptional condition that cause by the program itself.
   - This type of interrupt arises from illegal or erroneous use of an Instruction or data. It is known as traps.
   - Examples that cause the internal interrupts are register overflow, stack overflow, etc.
   - This interrupt are synchronous with program. If the program is rerun the internal interrupts will occur in the same place each time.

- **Software interrupt**

   - Software interrupts is initiated by executing an instruction.
   - Software Interrupt is a special call instruction that behave like an interrupt rather then subroutine Call.
   - The most common use of this interrupt is associated with supervisor switching from CPU. User mode to the supervisor mode.
   - When input-output transfer is required the supervisor mode is requested by means of supervisor call instruction.

WHAT IS STACK ORGANIZATION? EXPLAIN THE TYPES OF STACK ORGANIZATION.

- A stack is an ordered set of elements only one of which can be used at a time
- The point of access is called top of the stack
- The number of elements in the stack or length of the stack is a variable
- The items may be added to or deleted from the top of the stack as Push Down list or Last In First Out (LIFO) List
- A useful feature that is included in the CPU of most computer is stack or Last In First Out (LIFO) List
- A stack is a storage device that stores information in such a manner that the item stored last is the first item retrieved.
- The register that holds the address for the stack is called a Stack Pointer (SP) because its value always points at the top item in the stack
- The two operation of stack are PUSH and POP
- The operation of insertion is called PUSH
- The operation of deletion is called POP

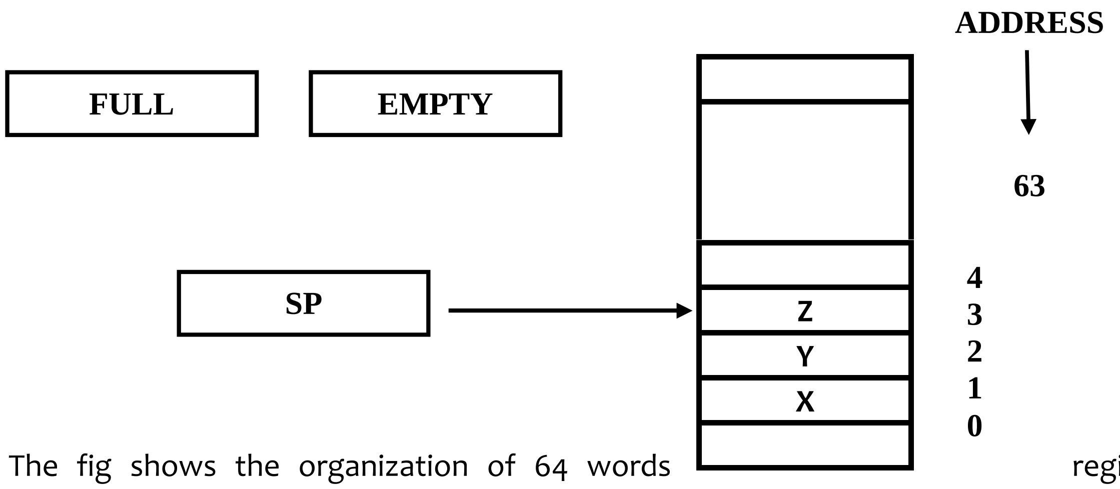
TYPES OF STACK OPERATION

- There are two types of stack operation like REGISTER STACK & MEMORY STACK

REGISTER STACK

- A stack can be placed in a portion of a large memory or it can be organized as a collection of a finite number of memory words or registers
- Register stacks organized the operation from each and every word and its operation into the memory in register stack operation

- The stack pointer (SP) register contains 4 binary numbers whose value is equal to the address of the word that is currently on top of the stack



- The fig shows the organization of 64 words register stack. As shown in figure
- Three items are place in stack that is x, y and z item is top of the stack. So that SP is now 3.
- SP means Stack pointer.
- DR means Data Register
- There are two types of operations are used in this type of Register Stack Operation like PUSH and POP

PUSH

- If the stack is not full then a new item is inserted with a push operation
- If the stack is full and user tries to insert a new value in the stack then it is called Stack Overflow error.

POP

- A new item is deleted from the stack if the stack is not empty.
- If the stack is empty and user tries to delete an item from the stack then it is called Stack Underflow error.

MEMORY STACK

- A stack exist as a stand alone unit or it can be implemented in RAM that attached to CPU
- There are three types of segments are used in the memory stack like 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
- This type of three segments are connected with the common address bus
- There are two types of operations are used in this type of Memory Stack Operation like PUSH and POP

PUSH

- If the memory is not full then a new item is inserted with a push operation
- If the memory is full and user tries to insert a new item in the memory then it is called Memory Overflow error or Full Memory error.

POP

- A new item is deleted from the memory if the memory is not empty.
- If the memory is empty and user tries to delete an item from the memory then it is called Memory Underflow error or Empty Memory error.

SHORT NOTE: REVERSE POLISH NOTATION

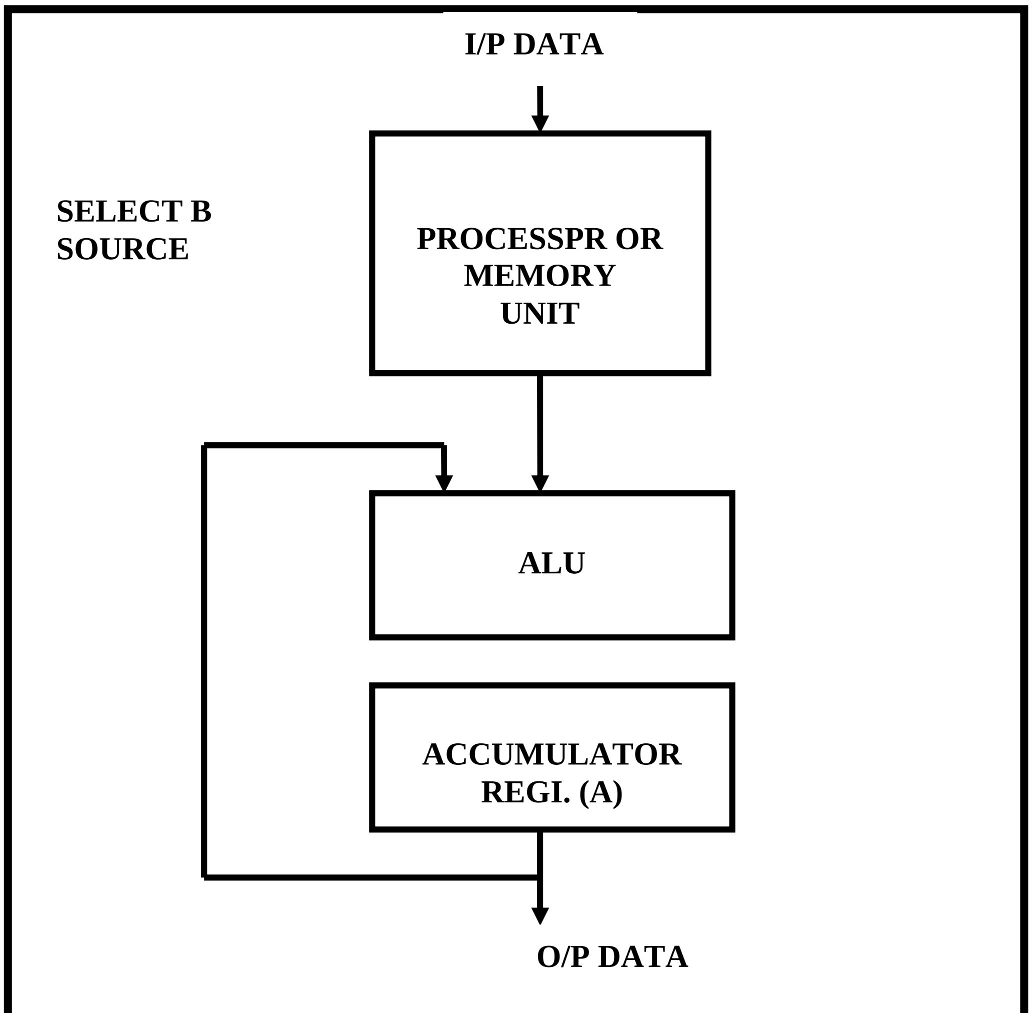
- Evaluating ordinary arithmetic expressions using a computer is difficult, particularly when an Expression consists of parenthesis and brackets.
- In this case an expression has to be scanned from Left as well from the right.
- These problems arise because in an ordinary arithmetic expression operator is placed in between the two operands. Such types of ordinary expressions are called Infix expression.
- To overcome this problem the polish mathematician Lukasi showed that a Arithmetic Expression can be represented in prefix or postfix Notation.
- In prefix notation (also known as Polish notation) the operator is placed before the operands while in postfix notation (also known as Reverse polish notation, RPN) the operator is placed after the operands.
- In short arithmetic expression can be represented in three different way as follows.

A+B	:	INFIX NOTATION
+AB	:	PREFIX OR POLISH NOTATION
AB+	:	POSTFIX OR REVERSE POLISH NOTATION (RPN)

- RPN has number of advantages over infix notation for expressing algebraic formulas.
- Any formula can be expressed without parenthesis.
- It is convenient for evaluating formulas on computer with stack.
- The infix operator has precedence which is undesirable.
- This disadvantages is eliminated By reverse polish notation.
- The main advantage of polish notation is that any formula can be expressed without Parenthesis.
- This is possible by combining reverse polish notation with a stack registers.
- Hence stack is very useful for handling long and complex problems involving arithmetic Expressions.
- This procedures is employed in some electronic calculators and computers.
- In this method The arithmetic expressions first converted reverse polish notation and then operands are pushed Into the stack in the order in which they appear.
- Let us discuss one example for more clarification of this procedure. Consider arithmetic
- Expression.
- (6*3) + (5*9)
- In reverse polish notation it is expressed as
- 36*95*+

SHORT NOTE: ACCUMULATOR REGISTER

- Some processor unit separate one register from all other so this type of register is called an Accumulator Register
- This register is also called AC or A register.
- The name of this register is derived from Arithmetic addition process.



PROCESSOR WITH ACCUMULATOR REGISTER

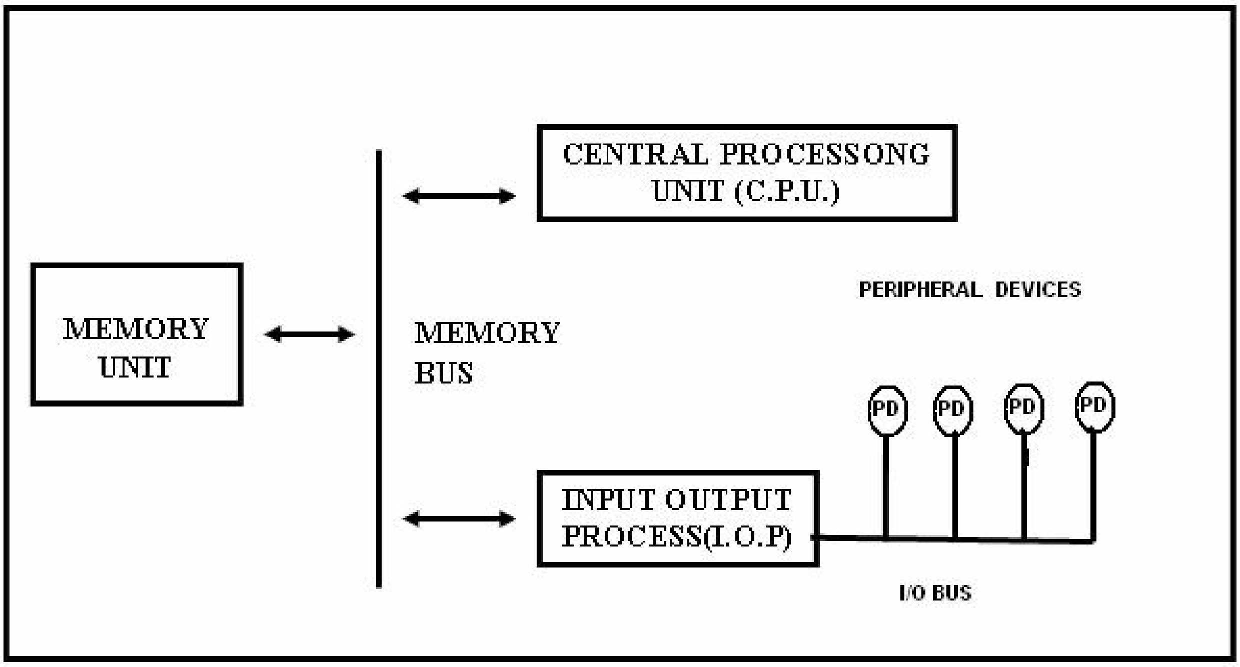
- The accumulator register is a multipurpose register capable of performing not only add micro operation but many other micro operation as well.
- Input B supplies one external source information & this information may come from other processors or directly from the main memory of the computer.
- The A register supplies the other source information to the ALU.
- The result of operation is transfer back to the A register and replace its previous contain.
- The output from the A register may go to an external destination or into the input terminals of other processor register.
- A register using the sequence of micro operations.
-

T1	$A \leftarrow 0$	CLEAR A
T2	$A \leftarrow A+R1$	TRANSFER R1 TO A

- T3 $A \leftarrow A+R2$ ADD R2 TO A.
-
- Register A is first cleared then first number.
- In R1 is transferred into the accumulator register (A) by adding it to the present zero content to A.
- The second number in R2 is then added to the present value of A.
- The sum formed in A may be used for other computations or may be transferred to a required destination.

SHORT NOTE: INPUT – OUTPUT PROCESSOR (IOP)

- An Input – Output Processor (IOP) is a software program that is used to control input – output operations
- An Input – Output Processor (IOP) controls sending and receiving data between Input – Output Process
- An Input – Output Processor (IOP) finds and corrects sending and receiving errors between Input – Output Process
- An Input – Output Processor (IOP) assembles and disassembles messages between Input – Output Process
- An Input – Output operation is performed by the CPU



- In figure, we can show that, Memory bus contains CPU and IOP for communication
- Input / Output bus (I / O) Bus is connected with the Input – Output Processor (IOP) with the help of Peripheral devices (PD).
- An Input – Output Processor (IOP) communicates with the peripheral device through the I/O bus and with the memory through the memory bus.
- When IOP needs to transfer data to or from memory, it starts a memory cycle from the CPU and then it transfers the data to or from memory.
- In computer, CPU is the master processor and IOP is slave processor.
- CPU sends information by using IOP path.

- IOP contains status word into that memory location.
- CPU checks the status word if the status word is correct then CPU inform to the IOP about sending information.
- After that, IOP reads and executes commands from the memory that located at the specified location.
- When the IOP completes the Input Output data transfer then it informs to the CPU for data transfer is completed.

UNIT 5: INPUT OUTPUT ORGANIZATION

WRITE A NOTE ON MEMORY BUSES.

- A collection of cables through which is transmitted from one peripherals devices into another peripherals devices that is known as Bus
- Buses can send the data information, address information, special instruction from one device into another device.
- There are many types of buses are used like
  - Internal bus
  - External bus
  - System bus
  - Instruction bus
  - Memory bus
  - Data bus
  - Address bus
  - Control bus
  - Input output bus

Internal Bus

- Internal bus is placed inside the processor.
- Internal bus is used to send information between processor register and internal components of processor

External Bus

- External bus is placed outside the processor.
- External bus is used to send information between one parts of computer into another part of computer

System Bus

- System bus is used to connect the processor with main memory
- Instruction Bus
- Instruction bus is used to fetch the instruction from main memory

Memory Bus

- Memory bus is used to carry memory location.
- Memory bus is used to connect system memory with the processor

Data Bus

- Data bus is used to send data from one parts of computer into another part of computer
- Data bus is used to send data to the processor, memory and other parts of computers
- Data bus is used to connect the processor, memory and other parts of computers for communicate with each other for transferring the data

Address Bus

- Address bus is used to send address from one parts of computer into another part of computer
- Address bus is used to send address to the processor, memory and other parts of computers

Control Bus

- Control bus is used to control the data and information of the data bus and address bus

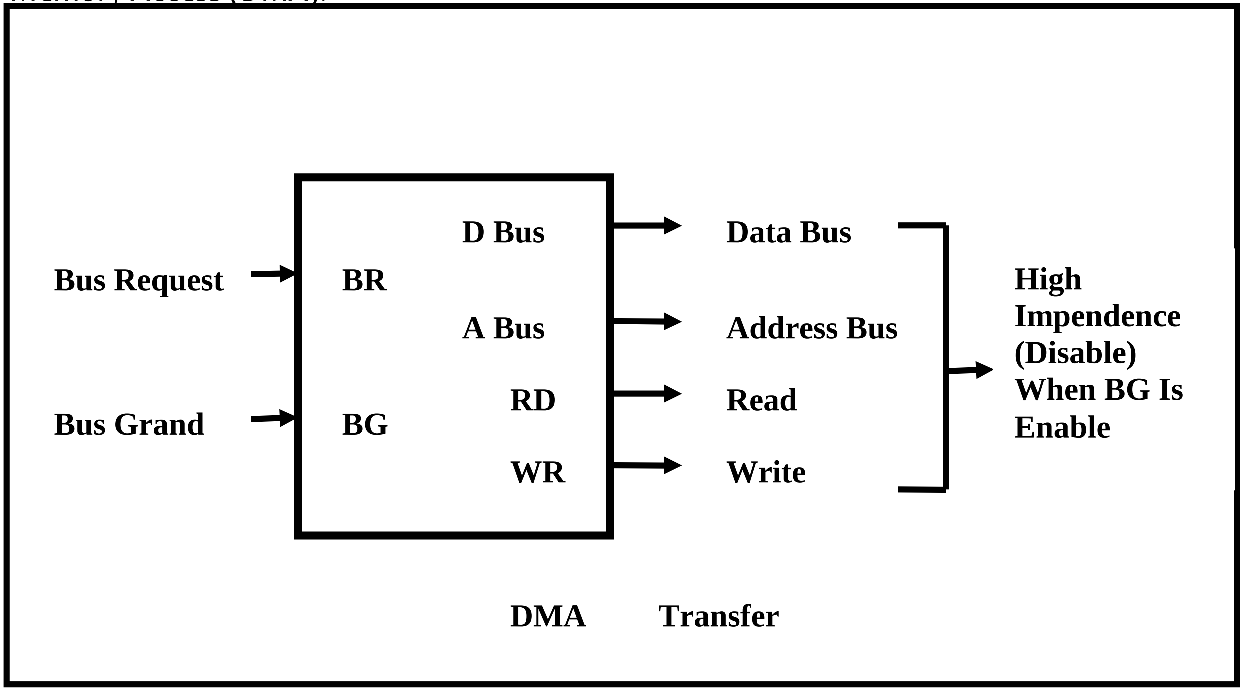
Input – Output Bus

- Input – Output bus is used to connect between processor and input – output parts for data transmission.

- Input – Output bus contains data line, address line and control line for data transmission

SHORT NOTE: DMA OR DMA CONTROLLER

- DMA means Direct Memory Access.
- DMA is a process of communication or data transfer that controlled by an external peripheral.
- When the DMA is used in the memory, the microprocessor controlled the data transfer.
- For 8085 microprocessor, HOLD and HLDAC signal use for this type of data transfer.
- Direct memory access is an input output technique for fast data transfers.
- Direct memory access (DMA) allows hardware system for reading or writing process of the central processing unit.
- Many hardware systems use DMA including Disk drive controllers, graphics cards, network cards and sound cards.
- The data transfer speed of Central processing unit is limited between secondary memory and Main memory.
- This problem can be resolved by removing CPU from the path.
- After Removing CPU, different peripheral manages the memory buses directly for the improvement of data transfer rate. This special kind of method is known as Direct Memory Access (DMA).



- In this type of DMA process, When Input device needs to transfer data to or from memory, it request to the DMA controller by setting DMA Bus Request Input to 1 and DMA Bus Grant Input to 1.
- Here 1 means Enable and 0 means Disable
- Bus Request (BR) is used to pass the request to the CPU.
- After that, CPU accept the request from the Bus Request (BS) signal when the Bus Grant (BG) is Enable.
- After that, CPU can read and write the request by using Read (RD) and Write (WR) signal.

- After that CPU gives address into that request by using Address Bus (A BUS) and also give data into that request by using Data Bus (D BUS).
- After that CPU can transfer to that request from one place into another by using DMA controller.
- When data transfer is completed then Bus Request and Bus Grant signal becomes disable.
- DMA method allows the Input / Output devices to directly communicate with the main memory of a computer.
- DMA method directly reads and writes data from the memory by using memory bus
- DMA is a technique for moving data directly between main memory and peripheral of computer without need of CPU
- DMA Controller contains 3 types of register like Address Register, Word Count Register and Control Register.

ADDRESS REGISTER

- Address register contains the address of memory where the data from the input device needs to be stored or from where the data for the output device needs to be fetched
- CPU gives this address to the data bus.
- The address is transferred to the Address Register from the Data Bus through the Internal Bus.
- Address in this register is incremented after transferring each word to or from memory

WORD COUNT REGISTER

- Word Count Register stores the number of words the need to be read from or written into memory
- It is decremented after each word transfer.
- When the value in this register becomes zero, it is assumed that data transfer has completed

CONTROL REGISTER

- The control register specifies the mode of data transfer like it is a read operation or it is a write operation
- CPU reads the data through the data bus after setting to Read Input to 1
- CPU writes the data through the data bus after setting to Write Input to 1