PRESIDENCY COLLEGE
(AUTONOMOUS)
AFFILIATED TO BENGALURU CITY UNIVERSITY, APPROVED BY AICTE, DELHI & RECOGNISED BY THE GOVT. OF KARNATAKA
RE-ACCREDITED BY NAAC WITH 'A+' GRADE

GAIN MORE KNOWLEDGE REACH GREATER HEIGHTS

UNIT – I: Number Systems

Binary, Octal, Hexa-decimal numbers, base conversion, addition, subtraction of binary numbers, one's and two's complements, positive and negative numbers, Computer Arithmetic: Addition and Subtraction, Multiplication and Division, Integer Arithmetic Operations. Digital Logic Circuits: Logic gates, Boolean algebra, Map Simplification- **REFER CLASS NOTES for the above concepts.**

Combinational Circuits : Half Adder , Full Adder , flip flops.
Sequential circuits: Shift registers, Counters, Integrated Circuits, Multiplexers, Demultiplexers, Encoder, Decoder.

2 MARKS QUESTION:

1. Expand the following: BIT, ASCII, EBCDIC.

BIT: A bit (binary digit) is the smallest unit of data that a computer can process and store. A bit is always in one of two physical states, similar to an on/off light switch. The state is represented by a single binary value, usually a 0 or 1.

ASCII is the American Standard Code for Information Interchange, an American National Standard computer character code. ASCII is an 7-bit code with 128 characters. Extended (8-bit) ASCII: 256 code points. Many computers reserve 8 bits for ASCII.

EBCDIC, is **extended binary-coded decimal interchange code**, data-encoding system, developed by IBM and used mostly on its computers, that uses a unique eight-bit binary code for each number and alphabetic character as well as punctuation marks and accented letters and non-alphabetic characters.

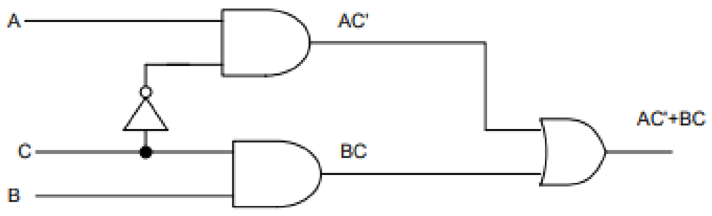
2. Define Computer Architecture

Computer Architecture is a structure of a digital computer, encompassing the design and layout of its instruction set and storage registers. **Instruction set architecture (ISA) acts as a bridge between computer's software and hardware**. A computer system is basically a machine that simplifies complicated tasks. The different components in the Computer System Architecture are **Input Unit, Output Unit, Storage Unit, Arithmetic Logic Unit, Control Unit** etc

3. Define Logic Gates

A logic gate is a device that acts as a building block for digital circuits. They perform basic logical functions that are fundamental to digital circuits. Most logic gates have two inputs and one output. Logic gates are based on Boolean algebra.

4. Draw the logic circuit for the following equation y= ac'+bc



5. What is Combinational Circuit? Give Example

- **Combinational circuit is a circuit in which we combine the different logic gates in the circuit.**
- **Combinational Logic circuit is implemented using Boolean circuits, where the output of logic circuit is a pure function of the present inputs only.**
- **Example: half- adder, full-adder, encoder, decoder.**

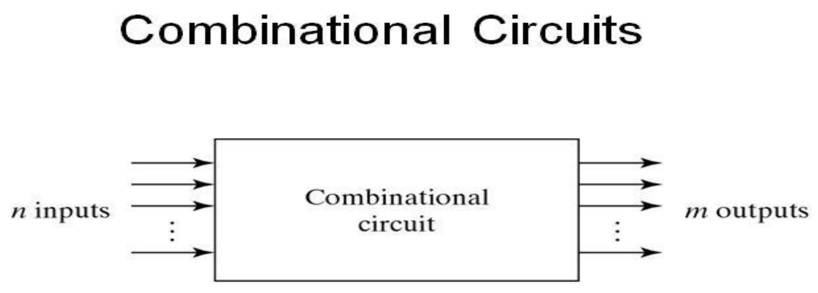
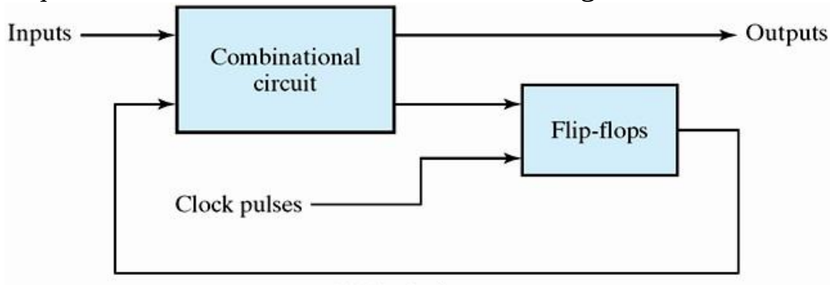


Fig. Block Diagram of Combinational Circuit

6. What is Sequential Circuit? Give Example

The sequential circuit is a special type of circuit that has a series of inputs and outputs.
sequential circuit consist of combinational logic circuit and memory element



(a) Block diagram

- The outputs of the sequential circuits depend on **both the combination of present inputs and previous outputs.**
- The previous output is treated as the present state. So, the sequential circuit contains the Combinational circuit and its memory storage elements.
- A sequential circuit doesn't need to always contain a combinational circuit. So, the sequential circuit can contain only the memory element.

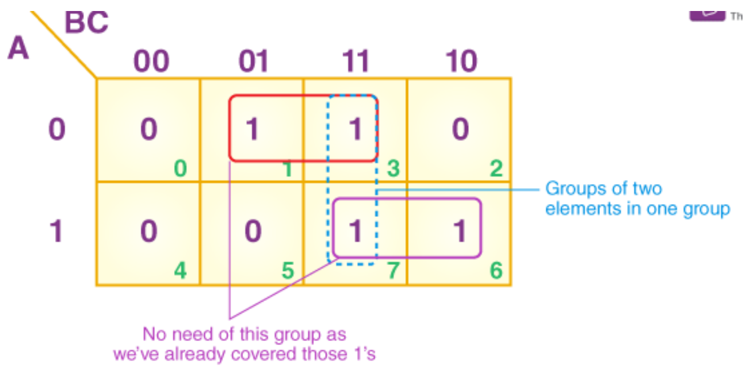
7. What is K-Map? Draw format for 3 variable input.

A Karnaugh map or a K-map refers to **a pictorial method that is utilised to minimise various Boolean expressions without using the Boolean algebra theorems**. We can easily minimise various expressions that have 2 to 4 variables using a K-map.

2 input variables =4
3 input variables= 8
4 input variables= 16

3 variables K-map:

Z = ∑A, B, C (1, 3, 6, 7)


A
BC
00
01
11
10
0
1
0
1
1
0
0
1
3
2
0
0
1
1
4
5
7
6
Groups of two elements in one group
No need of this group as we've already covered those 1's

8. Define minterm and maxterm

The product of all literals, either with complement or without complement, is known as **minterm**. Example: A.B , A'.B

The sum of all literals, either with complement or without complement, is known as **maxterm**. Example: A+B , A'+B, A+B'

9. What is racing around condition in JK Flip flop?

Race Around Condition In JK Flip-flop – For J-K flip-flop, if J=K=1, and if clk=1 for a long period of time, then Q output will toggle as long as CLK is high, which makes the output of the flip-flop unstable or uncertain. This problem is called race around condition in J-K flip-flop.

10. Define Flip flop. Give example

Definition of FLIPFLOP:

- **It is the basic memory element in a digital computer.**
- **It is used to store one bit of information with a '0' or '1' . It has only two states '0' OFF & '1' ON.**
- **TYPES OF FLIPFLOPS**

- **SR flip-flop (unclocked/ Latch)**
- **SR flip- flop(clocked)**
- **D flip-flop**
- **JK flip-flop**
- **T flip-flop**
- **MS Flip-flop**

11. What is an IC?

An **integrated circuit (IC)** is manufactured using silicon material and mounted in a ceramic or plastic **container (known as Chip).**

The basic components of an IC consist of electronic circuits for the digital gates. The various gates are interconnected inside an IC to form the required circuit.

The following categories can broadly classify an Integrated Circuit (IC):

- SSI (Small Scale Integration Devices) These type of devices contain several independent gates in a single package. The number of logic gates are usually less than 10 and are limited by the number of pins available in the IC.
- MSI (Medium Scale Integration Devices) These type of devices has a complexity of approximately 10 to 200 gates in a single package. The basic components include decoders, adders, and registers.
- LSI (Large Scale Integration Devices) LSI devices contain about 200 to a few thousand gates in a single package. The basic components of an LSI device include digital systems, such as processors, memory chips, and programmable modules.
- VLSI (Very Large Scale Integration Device) This type of devices contains thousands of gates within a single package. The most common example of a VLSI device is a complex microcomputer chip.

12. Define TTL, ECL

- TTL (Transistor-transistor Logic): The TTL technology was an upgraded version of a previous technology called as DTL (Diode-Transistor Logic).TTL came in existence when these diodes are replaced with transistors to improve the circuit operation.
- ECL (Emitter-coupled Logic): The ECL technology provides the highest-speed digital circuits in integrated form. An ECL circuit is used in supercomputers and signal processors where high speed is essential. The logic gates continuously draw current

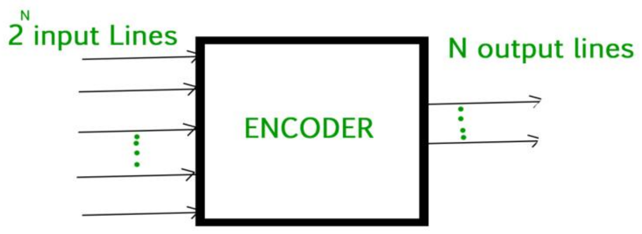
even in the inactive state. Hence power consumption is more as compared to other logic families.

- MOS (Metal-oxide semiconductor): The MOS (Metal-oxide semiconductor) is a unipolar transistor that depends on the flow of only one type of carrier, which may be electrons (n-channel) or holes (p-channel).MOS technology is generally categorized in two basic forms:
  - A p-channel MOS is referred to as PMOS
  - .An n-channel MOS is referred to as NMOS.
- CMOS (Complementary Metal-oxide semiconductor): A complementary metal-oxide semiconductor (CMOS) is the semiconductor technology used in most of today's integrated circuits (ICs), also known as chips or microchips.

13. What is encoder?

An encoder is a combinational circuit that converts binary information in the form of a 2^N input lines into N output lines, which represent N bit code for the input.

It has maximum of 2^n input lines and 'n' output lines. It will produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2^n input lines with 'n' bits.



Example: 8 to 3 line Encoder:
The 8 to 3 line Encoder is also known as **Octal to Binary Encoder**. In 8 to 3 line encoder, there is a total of eight inputs, i.e., Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7 and three outputs, i.e., A_0, A1, and A_2.

14. Define registers

Registers are a type of computer memory used to quickly accept, store, and transfer data and instructions that are being used immediately by the CPU. It is a sequence of flipflops.

15. Define Binary counters

A special type of sequential circuit used to count the pulse is known as a counter, or a collection of flip flops where the clock signal is applied is known as counters. The counter is one of the widest applications of the flip flop. Based on the clock pulse, the output of the counter contains a predefined state. The number of the pulse can be counted using the output of the counter.

16. What are shift registers?

A shift register is **a digital memory circuit found in calculators, computers, and data-processing systems**. A register that is designed to allow the bits of its contents to be moved to left or right.

Types of Shift registers:

- Serial In − Serial Out shift register.
- Serial In − Parallel Out shift register.
- Parallel In − Serial Out shift register.
- Parallel In − Parallel Out shift register.

17. What is Number system?

The number system is simply a system to represent or express numbers. There are various types of number systems and the most commonly used ones are decimal number system, binary number system, octal number system, and hexadecimal number system.

6 MARKS QUESTIONS

18. State and prove De-morgans Theorem

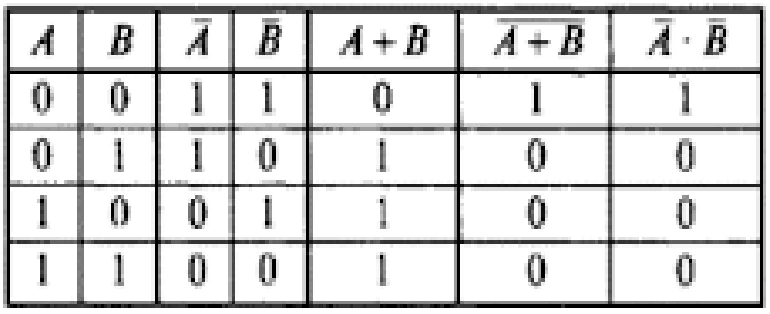
De Morgan´s Theorem***

1. **(A + B)' = A'B’ (OR LAW)→equ 1**
2. **(AB)' = A' + B‘ (AND LAW)→equ2**

1. Statement 1 (A + B)' = A'B’

This theorem states that the complement of the sum of Boolean expression is equal to product of the complement of the individual expression.

Proof using truth table

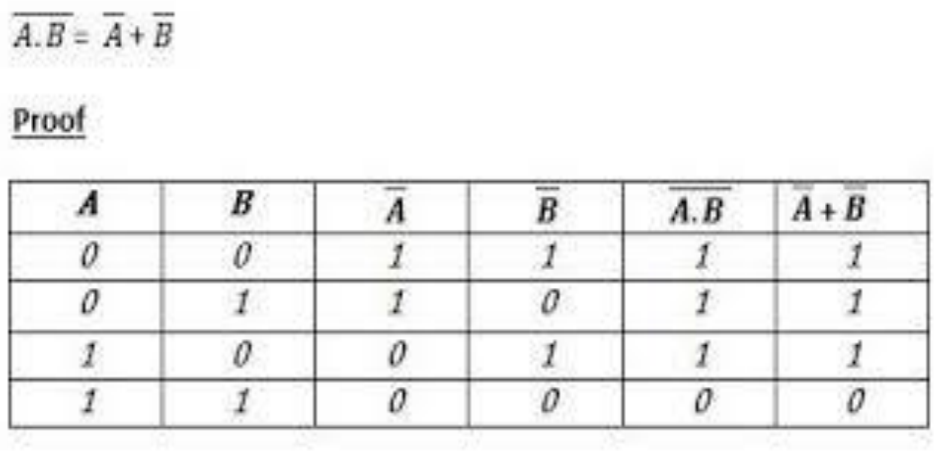


A	B	$\overline{A}$	$\overline{B}$	$A+B$	$\overline{A+B}$	$\overline{A}\cdot\overline{B}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

2. **Statement 2: (AB)' = A' + B'**

   This theorem states that the complement of product of Boolean expression is equal to the sum of the complement of the individual expression.

   Proof using truth table



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

Proof

A	B	$\overline{A}$	$\overline{B}$	$\overline{A.B}$	$\overline{A}+\overline{B}$
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	0	0

Hence statement 1 and statement 2 is proved by using Truth table.

19. Define Combinational Circuit and explain Half Adder

- **Combinational circuit is a circuit in which we combine the different logic gates in the circuit.**
- **Combinational Logic circuit is implemented using Boolean circuits, where the output of logic circuit is a pure function of the present inputs only.**

Combinational Circuits

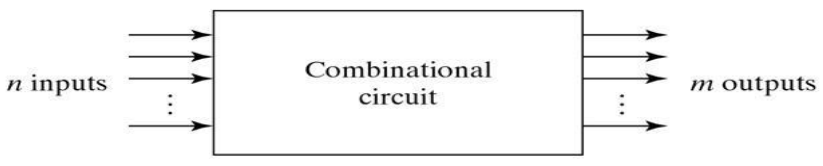
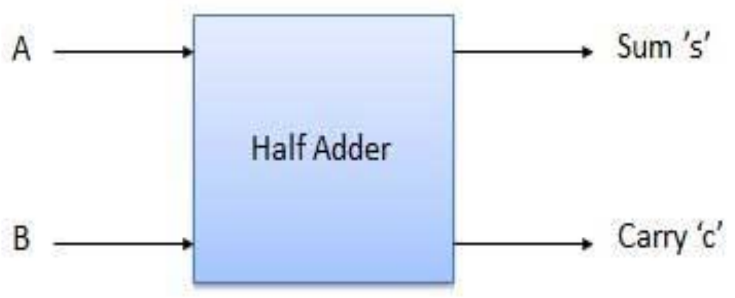


Fig. Block Diagram of Combinational Circuit

- **HALF ADDER (DEFINITION):**

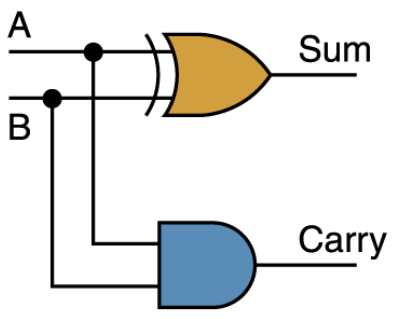
- **Half adder is a combinational logic circuit with two inputs and two outputs. It is used to add two binary bits.**
- **A circuit that computes the addition of two bits and produces the Sum and Carry bit is called a half adder**
- **This circuit has two inputs A & B and two outputs namely SUM and CARRY.**
- **HALF ADDER (BLOCK DIAGRAM)**



HALF ADDER (TRUTH TABLE)

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

HALF ADDER (LOGIC CIRCUIT)



- **HALF ADDER (Expressions):**

SUM = A ⊕ B

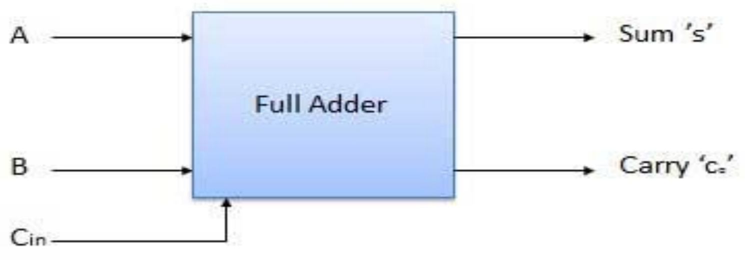
CARRY = AB

20. Explain Full Adder circuit with neat diagram

FULL Adders (Definition & Block Diagram)

- **FULL Adders Definition: A full - adder is a combinational circuit that performs the addition of three binary bits. It consists of three inputs A, B, C & two outputs SUM and CARRY.**

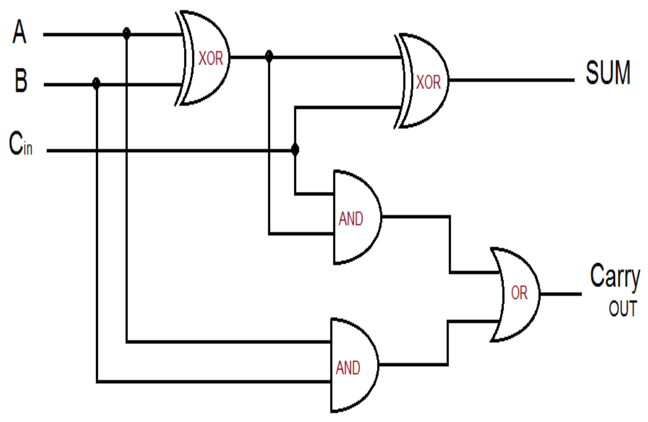
- **FULL Adders BLOCK DIAGRAM**



FULL ADDER (TRUTH TABLE)

Inputs			Outputs	
A	B	C_{in}	Sum	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

FULL ADDER (LOGIC CIRCUIT)



FULL ADDER(EXPRESSION)

SUM = (A XOR B) XOR Cin = (A ⊕ B) ⊕ Cin

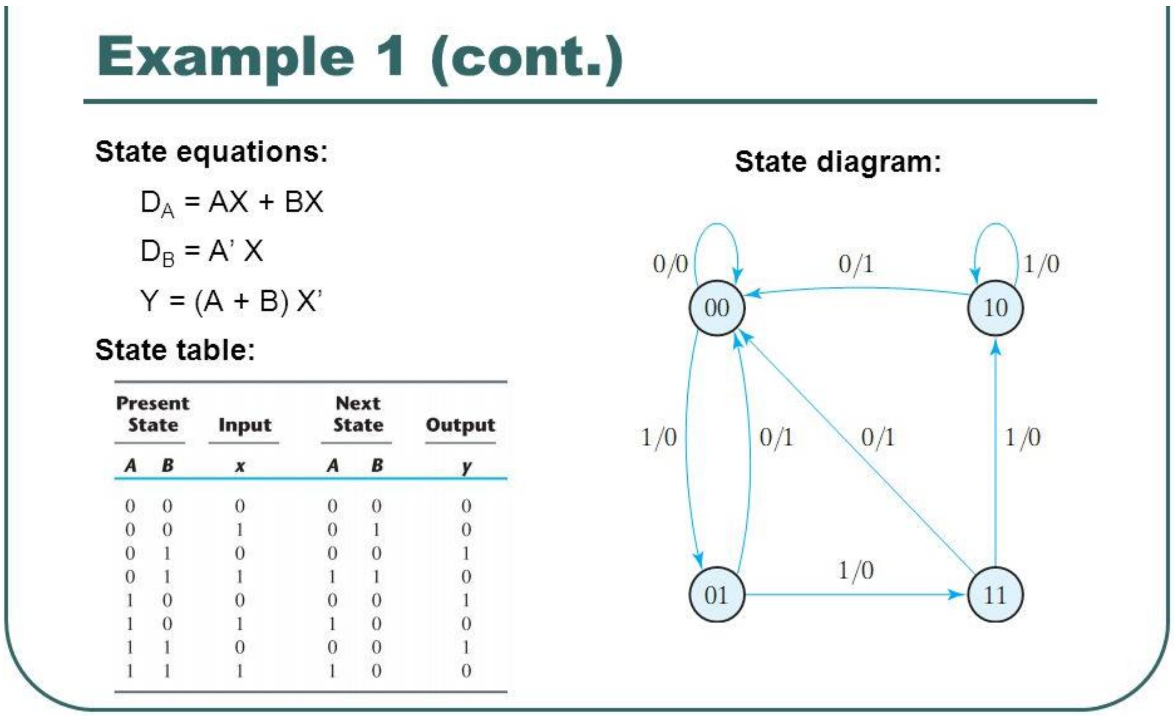
For the CARRY-OUT (Cout) bit:

Cout = Cin(A XOR B) OR A AND B = Cin(A ⊕ B) + A.B

21. Define state diagram and state table for y= AX' + BX'

The state diagram is a graphical representation of a sequential circuit in which the state are represented by circles and transitions between states shown by arrows. In another words, a state diagram consists of nodes interconnecting directed line segments corresponding to each state of the circuit.

Representation of state machine using relationship between input(s) present state, next state, and the output(s) in the tabular format is known as the State Table.



Example 1 (cont.)

State equations:

$D_A = AX + BX$

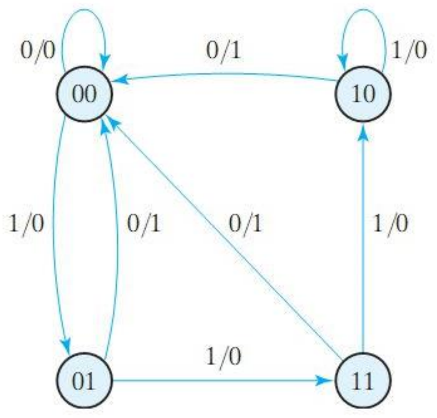
$D_B = A'X$

$Y = (A + B)X'$

State table:

Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

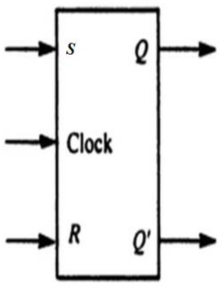
State diagram:



22. Explain SR flip flop.

- **SR FLIPFLOP or SET/RESET FLIP FLOP (DEFINITION):**

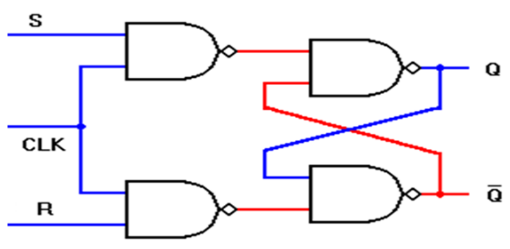
- The SR Flip Flop has 3 inputs, SET (S),RESET (R) and CLK.
- The SR Flip Flop has two outputs, Q and Q¯
- The Q output is considered the normal output and is the one most used.
- The other output Q¯ is simply the compliment of output Q.
- **SR FLIPFLOP (BLOCK DIAGRAM)**



SR FLIPFLOP (TRUTH TABLE)

INPUTS			OUTPUTS		
CLK	S	R	Q	Q'	STATUS
1	0	0	No change		Hold
1	0	1	0	1	RESET(0)
1	1	0	1	0	SET(1)
1	1	1	INVALID		PROHIBITED

SR FLIPFLOP (LOGIC CIRCUIT)



- **WORKING OF SR FLIPFLOP:**

- **Case 1:With the inputs the S=0 and R=0, then the output remains in previous state i.e. it holds the previous data.(No change/ Hold state)**

- **Case 2:With the inputs set to S=0and R=1, when the clock pulse is applied, the active high signal on R resets the filp flop to 0 ,then the flip flop said to be in the RESET state.**

- **Case 3:With the inputs set to S=1and R=0, when the clock pulse is applied, the active high signal on S sets the flip flop to 1 ,then the flip flop said to be in the SET state.**

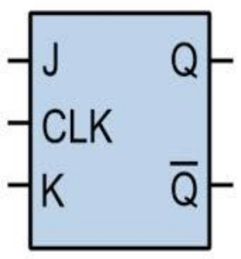
- **Case 4: When both the S=1 and R=1, then the flip flop will be in undefined state(INVALID/PROHIBITED).**

23. What is JK Flip flop explain in detail.

- **JK FLIPFLOP or JACK/KILBY FLIP FLOP (DEFINITION):**

- A JK flip-flop has two inputs similar to that of RS flip-flop. We can say JK flip-flop is a refinement of RS flip-flop.
- JK means Jack Kilby, a Texas instrument engineer who invented IC.
- The three inputs of JK Flip-flop is J (set) , K (reset) ,clk and two outputs Q and Q’

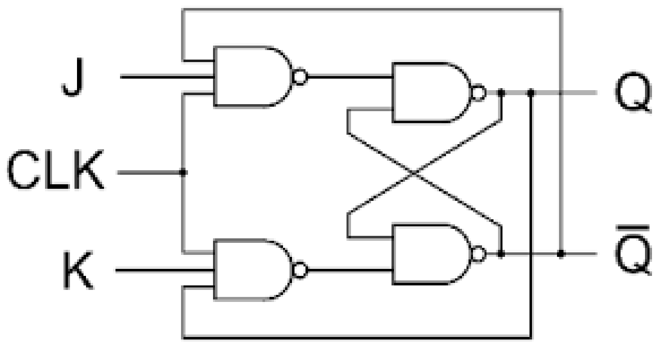
- **JK FLIPFLOP (BLOCK DIAGRAM)**



JK FLIPFLOP (TRUTH TABLE)

INPUTS			OUTPUTS		MODE
CLK	J	K	Q	Q'	
1	0	0	No change		Hold
1	0	1	0	1	RESET
1	1	0	1	0	SET
1	1	1	TOGGLE		TOOGLE

JK FLIPFLOP (LOGIC CIRCUIT)



- **WORKING OF JK FLIPFLOP:**

- **Case 1:With the inputs the J=0 and K=0, then the output remains in previous state i.e. it holds the previous data.(No change)**

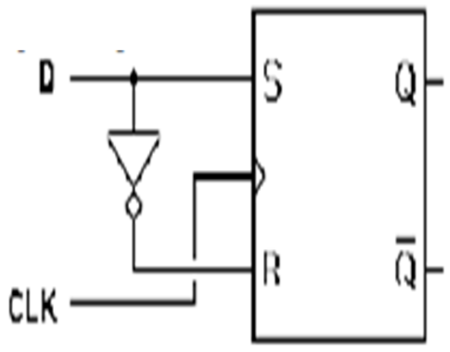
- **Case 2:With the inputs set to J=0and K=1, when the clock pulse is applied, the active high signal on R resets the flip flop to 0 ,then the flip flop said to be in the RESET state.**

- **Case 3:With the inputs set to J=1and K=0, when the clock pulse is applied, the active high signal on S sets the flip flop to 1 ,then the flip flop said to be in the SET state.**
- **Case 4: When both the J=1 and K=1, then the flip flop will be in TOGGLE state(TOGGLE).**

24. Explain D and T Flip flop in detail

- **D FLIPFLOP or DELAY/DATA FLIP FLOP (DEFINITION):**

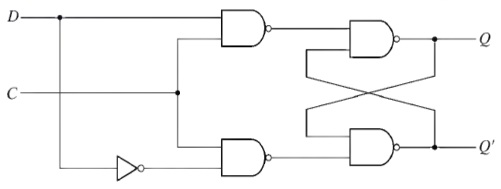
- It is a flip-flop with single input data D and a clock
- It is called as **data flip flop and also Delay Flip flop**
- The data at D input is delayed by one clock pulse before it gets to output.
- **D FLIPFLOP (BLOCK DIAGRAM)**



D FLIPFLOP (TRUTH TABLE)

INPUTS		OUTPUTS		
CLK	D	Q	Q'	STATUS
1	0	0	1	RESET
1	1	1	0	SET

D FLIPFLOP (LOGIC CIRCUIT)

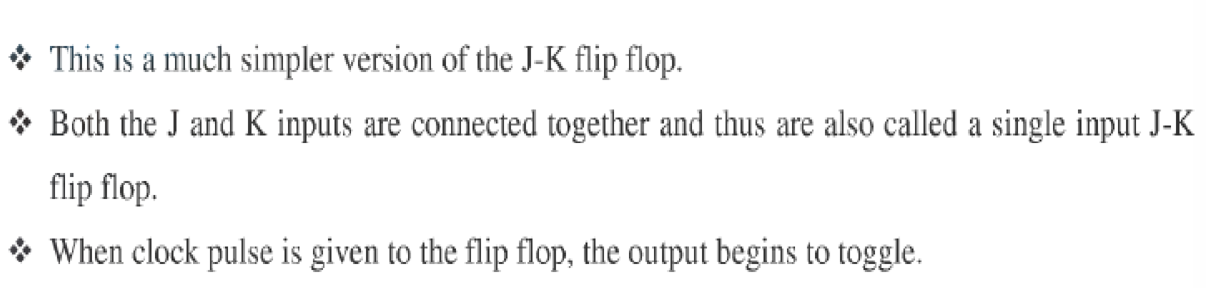


(a) Logic diagram

- **WORKING OF D FLIPFLOP:**

- **Case 1 :With the inputs set to D=0, when the clock pulse is applied, the active high signal resets the flip flop to Q= 0 ,then the flip flop said to be in the RESET state.**
- **Case 2:With the inputs set to D=1, when the clock pulse is applied, the active high signal SETS the flip flop to Q=1 ,then the flip flop said to be in the SET state.**

- **T FLIPFLOP or TOGGLE FLIP FLOP (DEFINITION):**

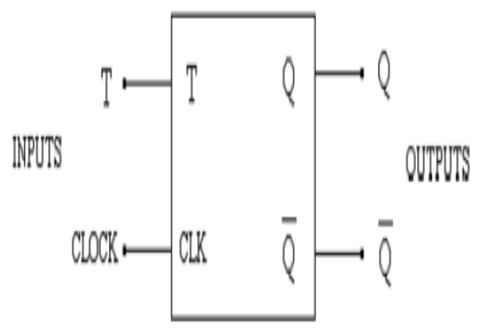


❖ This is a much simpler version of the J-K flip flop.

❖ Both the J and K inputs are connected together and thus are also called a single input J-K flip flop.

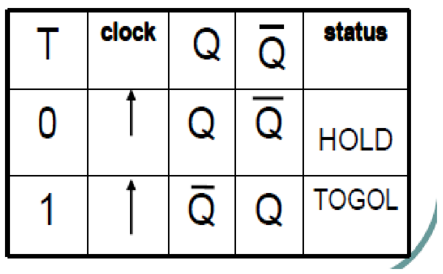
❖ When clock pulse is given to the flip flop, the output begins to toggle.

- **T FLIPFLOP (BLOCK DIAGRAM)**



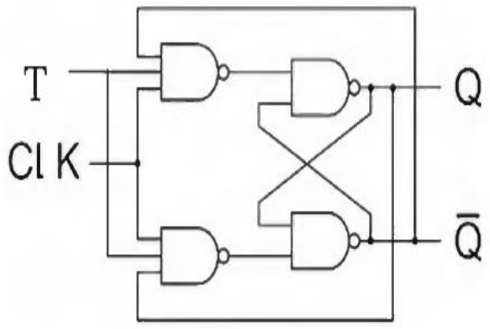
T FLIPFLOP (TRUTH TABLE)

Figure 4.5.2 :Truth Table for T Flip Flop



T	clock	Q	$\bar{Q}$	status
0	↑	Q	$\bar{Q}$	HOLD
1	↑	$\bar{Q}$	Q	TOGOL

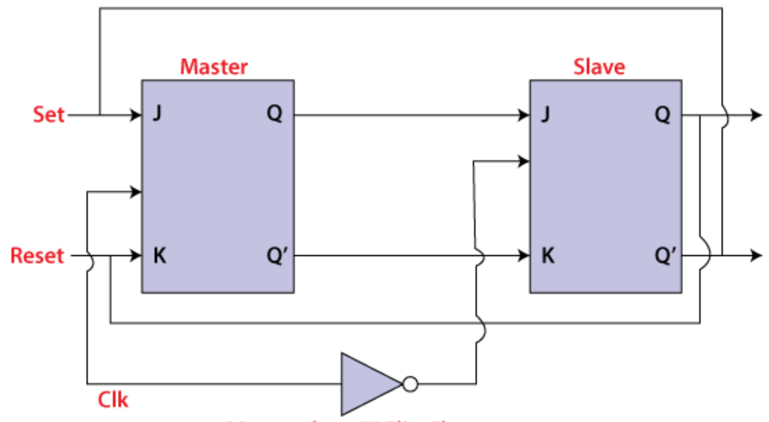
T FLIPFLOP (LOGIC CIRCUIT)



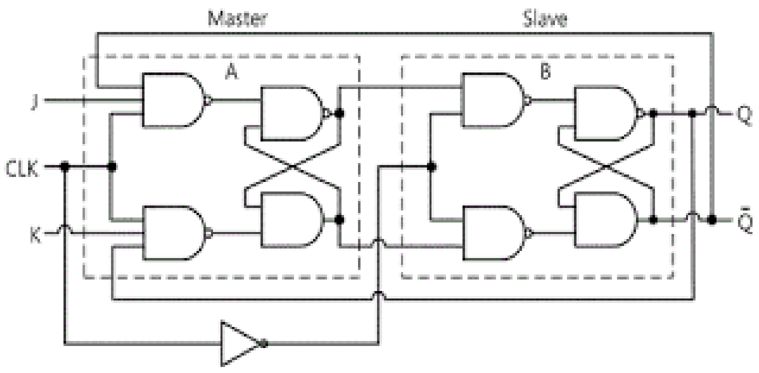
- **WORKING OF T FLIPFLOP:**
- **CASE1: When T=0 and clock=1 then Q and Q' Holds Previous States. So no change in outputs. This state is referred as HOLD STATE**
- **CASE 2: When T=1 and clock=1 then Q and Q' will change their outputs from 0 to 1 and 1 to 0 for every clock period, so toggling of outputs. This state is referred as TOGGLE STATE**

25. Explain Master- slave flipflop.

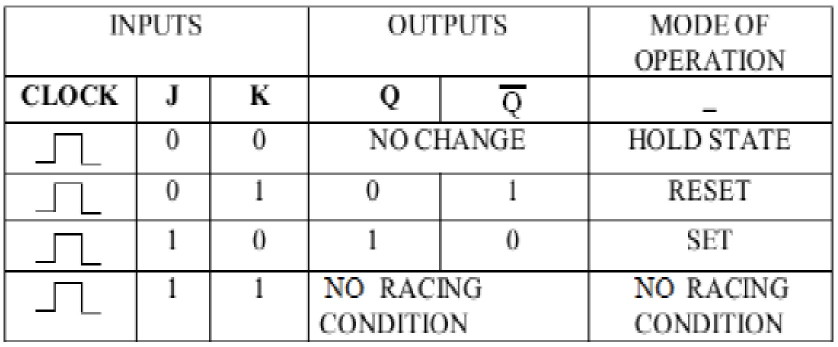
Master slave Flip flop: The Master-Slave Flip-Flop is basically a combination of two JK flip-flops connected together in a series configuration. Out of these, one acts as the **"master"** and the other as a **"slave"**. The output from the master flip flop is connected to the two inputs of the slave flip flop whose output is fed back to inputs of the master flip flop. In addition to these two flip-flops, the circuit also includes an **inverter**.



Master-slave JK Flip-Flop



b) Truth table



INPUTS			OUTPUTS		MODE OF OPERATION
CLOCK	**J**	**K**	**Q**	**$\bar{Q}$**	–
⎍	0	0	NO CHANGE		HOLD STATE
⎍	0	1	0	1	RESET
⎍	1	0	1	0	SET
⎍	1	1	NO RACING CONDITION		NO RACING CONDITION

- **WORKING OF JK FLIPFLOP:**

- **Case 1:With the inputs the J=0 and K=0, then the output remains in previous state i.e. it holds the previous data.(No change)**

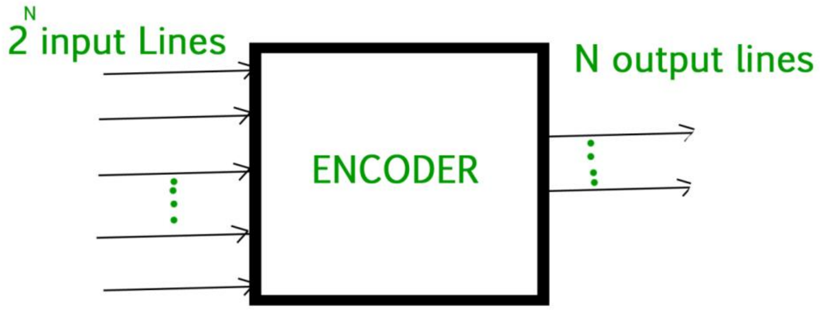
- **Case 2:With the inputs set to J=0and K=1, when the clock pulse is applied, the active high signal on R resets the flip flop to 0 ,then the flip flop said to be in the RESET state.**

- **Case 3:With the inputs set to J=1and K=0, when the clock pulse is applied, the active high signal on S sets the flip flop to 1 ,then the flip flop said to be in the SET state.**

- **Case 4: When both the J=1 and K=1, then the flip flop will NOT BE IN (NO RACING CONDITION).**

26. What is an Encoder? Explain 3x8 encoder.

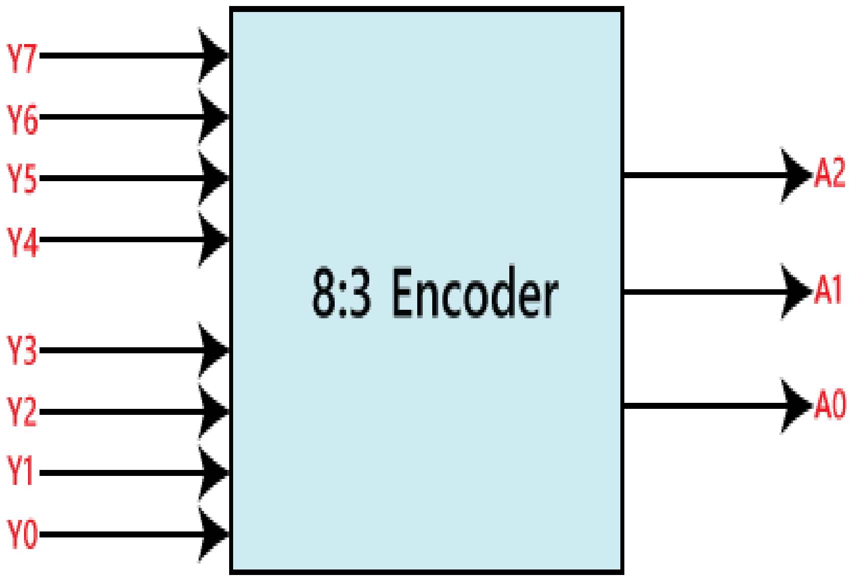
An encoder is a combinational circuit that converts binary information in the form of a 2^N input lines into N output lines, which represent N bit code for the input.



8 to 3 line Encoder: **The 8 to 3 line Encoder is also known as** Octal to Binary Encoder**. In 8 to 3 line encoder, there is a total of eight inputs, i.e., Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7 and three outputs, i.e., A_0, A1, and A_2. In 8-input lines, one input-line is set to true at a time to get the respective binary code in the output side.**

Below are the block diagram and the truth table of the 8 to 3 line encoder.

Block Diagram:



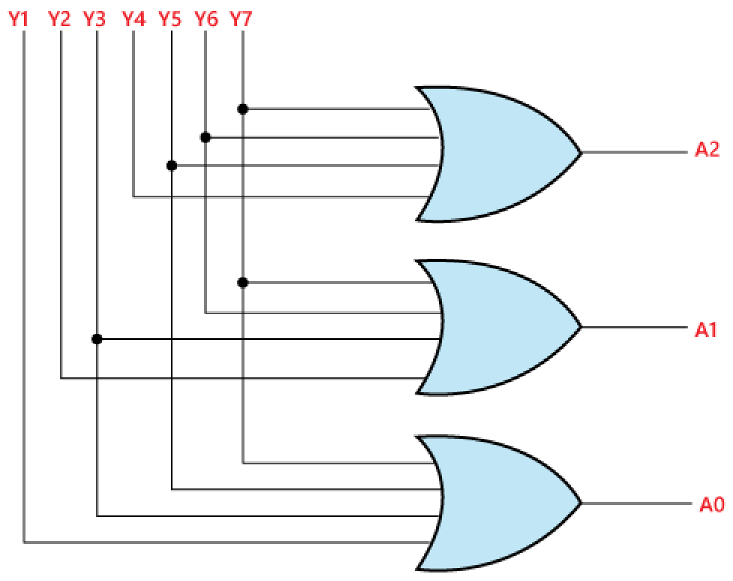
Truth Table:

INPUTS								OUTPUTS		
Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_2	A_1	A_0
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 logical expression of the term A0, A1, and A2 are as follows:

$A_2=Y_4+Y_5+Y_6+Y_7$
$A_1=Y_2+Y_3+Y_6+Y_7$
$A_0=Y_7+Y_5+Y_3+Y_1$

Logical circuit of the above expressions is given below:



Uses of Encoders:

1. These systems are very easy to use in all digital systems.
2. Encoders are used to convert a decimal number into the binary number. The objective is to perform a binary operation such as addition, subtraction, multiplication, etc.

27. Explain 4x1 multiplexer in detail.

4×1 Multiplexer:

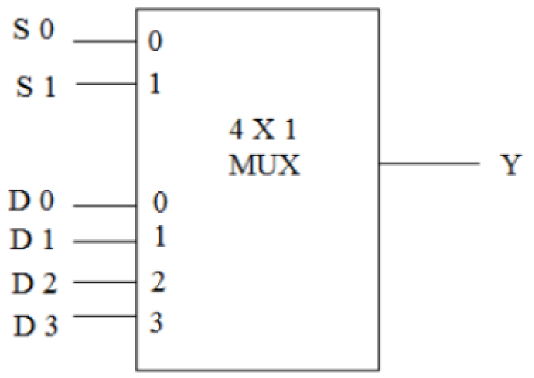
- In the 4×1 multiplexer, there is a total of four inputs, i.e., d_0, d_1, d_2, and d_3, 2 selection lines, i.e., S_0 and S_1 and single output, i.e., Y.
- On the basis of the combination of inputs that are present at the selection lines S^0 and S_1, one of these 4 inputs are connected to the output.

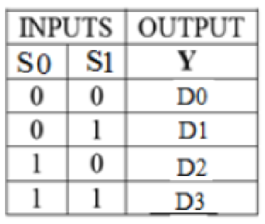
4 X 1 MUX (4-bit MUX / 4 to 1 line MUX)

2^n –to – 1 line

2^2 –to – 1 line ; 4 inputs ; 2 select input ; 1 output

Logic Symbol **Function Table**



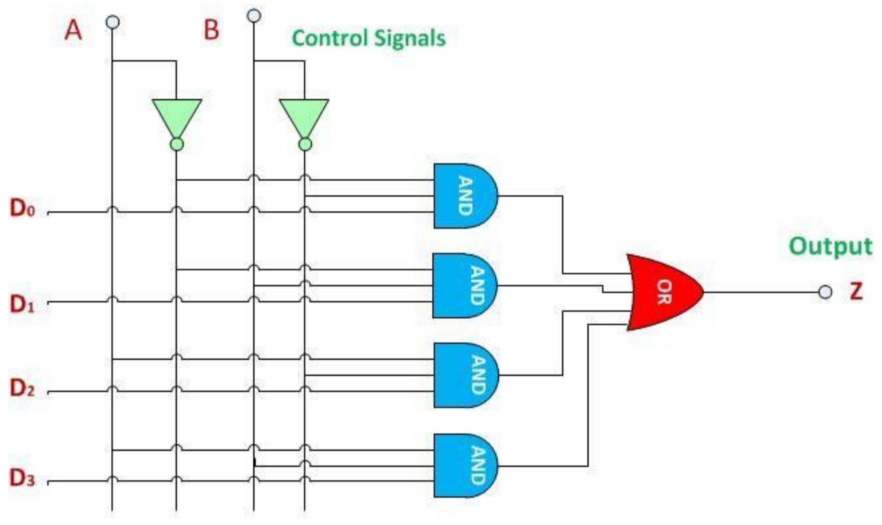


INPUTS		OUTPUT
S0	S1	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

- The logical expression of the term Y is as follows:

  $Y=S_1' S_0' D_0+S_1' S_0 D_1+S_1 S_0' D_2+S_1 S_0 D_3$

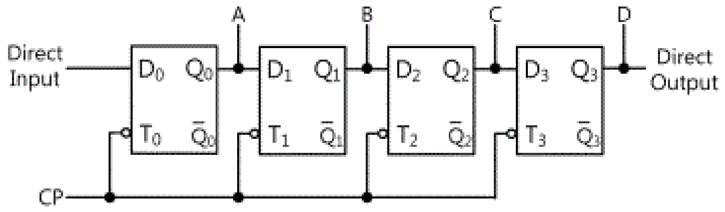
Logical circuit of the above expression is given below:



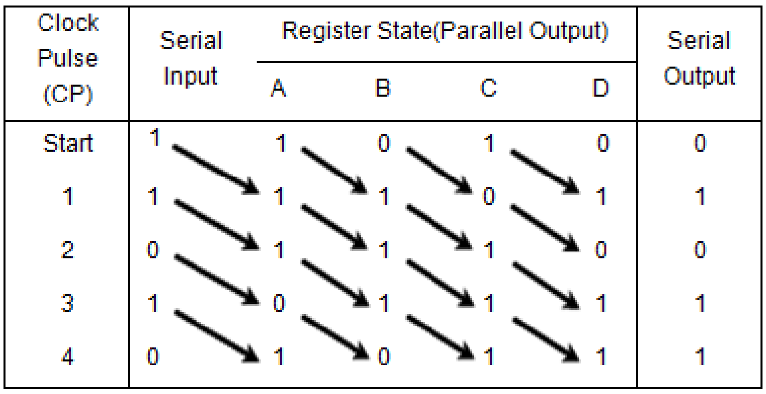
Four – To – One Multiplexer

28. Explain the operation of 4 bit shift register

A *shift register* basically consists of several single bit "D-Type FF", one for each data bit, either a logic "0" or a "1", connected together in a serial type daisy-chain arrangement so that the output from one data latch becomes the input of the next latch and so on.


Direct Input
A
B
C
D
D_0 Q_0 T_0 $\bar{Q}_0$
D_1 Q_1 T_1 $\bar{Q}_1$
D_2 Q_2 T_2 $\bar{Q}_2$
D_3 Q_3 T_3 $\bar{Q}_3$
Direct Output
CP

(a) Shift Register using D Flip-flop



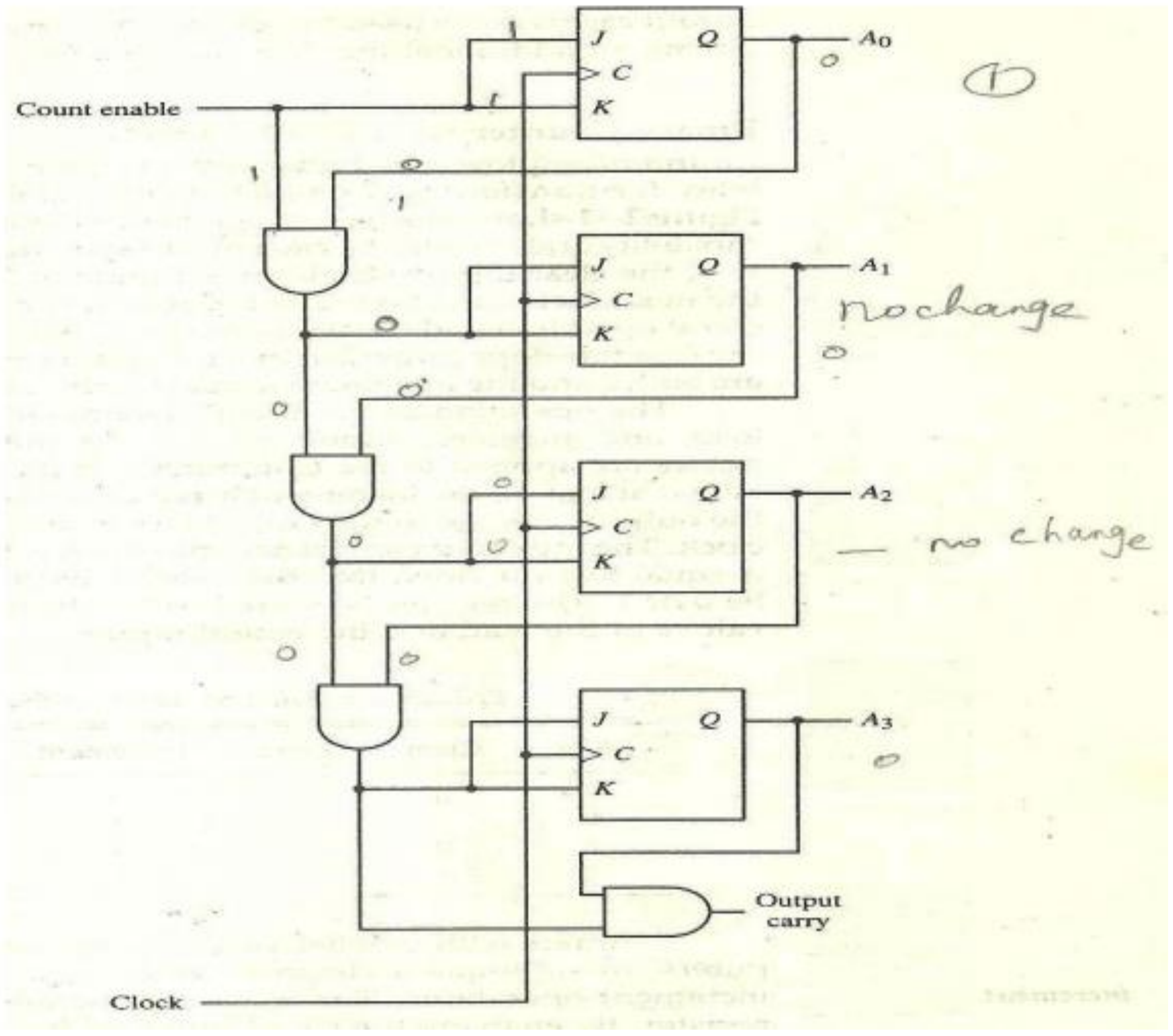
Clock Pulse (CP)	Serial Input	Register State(Parallel Output)				Serial Output
		A	B	C	D	
Start	1	1	0	1	0	0
1	1	1	1	0	1	1
2	0	1	1	1	0	0
3	1	0	1	1	1	1
4	0	1	0	1	1	1

(B) State Table of 4 bit Right Shift Register

29. write a note on 4-BIT binary Counter

4 bit synchronous binary counter:

- The C inputs of all flip flops receive the common clock
- If the count enabled is 0, all J and K inputs are maintained at 0 and the output of counter does not change
- A_0 is complemented when counter is enabled and clock goes to positive transition
- Each of the other three flip flops are complemented when all previous least significant flip flop are equal to 1 and count is enabled



Present State				Next State			
A_3	A_2	A_1	A_0	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0

30. What is an IC? Classify them

- An **integrated circuit (IC)** is manufactured using silicon material and mounted in a ceramic or plastic **container (known as Chip).**
- The basic components of an IC consist of electronic circuits for the digital gates. The various gates are interconnected inside an IC to form the required circuit.

The following categories can broadly classify an Integrated Circuit (IC):

- SSI (Small Scale Integration Devices) **These type of devices contain several independent gates in a single package. The number of logic gates are usually less than 10 and are limited by the number of pins available in the IC.**
- MSI (Medium Scale Integration Devices) **These type of devices has a complexity of approximately 10 to 200 gates in a single package. The basic components include decoders, adders, and registers.**
- LSI (Large Scale Integration Devices) **LSI devices contain about 200 to a few thousand gates in a single package. The basic components of an LSI device include digital systems, such as processors, memory chips, and programmable modules.**
- VLSI (Very Large Scale Integration Device) **This type of devices contains thousands of gates within a single package. The most common example of a VLSI device is a complex microcomputer chip.**

10 MARKS QUESTIONS.

31. Perform 1's and 2's Complement Subtraction.

Binary subtraction using 1’s complement method

Steps

1. **Identify the smaller number**
2. **Determine the 1’s complement of the smaller number**
3. **Add 1’s complement of the number to larger number**
4. **Remove the carry and add it to the result**
5. **If second number is larger then add (–)sign before the result**

Q1 . Subtract using 1’s complement 0011 and 0101

STEP1: Identify the smaller number

Identify smaller number➔ 0011

STEP2: Determine the 1’s complement of the smaller number

1’s complement number of smaller➔ 1100

STEP3: Add 1’s complement of the number to larger number

Add 1’complent to larger number

```
  1100
+ 0101
------
 1000 1
```

STEP4: Remove the carry and add it to the result

Remove the carry and add it to the result

```
0001
 +   1
------
 0010
```

STEP5: If second number is larger then add (–)sign before the result

Result is -0010

2’s complement subtraction

1. **Identify smaller number**
2. **Determine the 2's complement of a smaller number**
3. **Add the 2's complement number to larger number**
4. **Discard the final carry**
5. **If second number is larger than first number then add (-) before the result**

Perform 1001- 1100

STEP1: Identify smaller number

1001 – smaller number

STEP2: Determine the 2's complement of a smaller number

2'S COMPLEMENT = 1'S COMPLEMENT +1

= 0110+1

= 0111 (2'S COMPLEMENT)

0110+1 = 0111(2's complement of smaller number)

STEP3: Add the 2's complement number to larger number

0111 → 2's comp of smaller number

+1100→ larger number

10011→ Discard carry and consider only result

STEP4: Discard the final carry

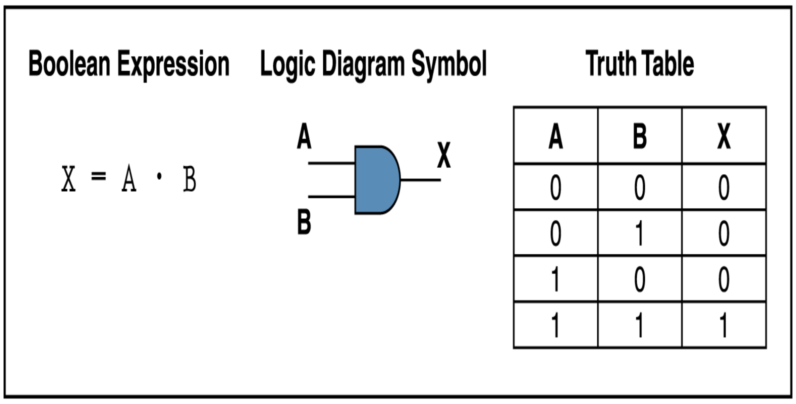
10011→ Discard carry and consider only result

STEP5: If second number is larger than first number then add (-) before the result. So result is (- 0011)

32. Explain Basic logic gates in details

AND GATE DEFINITION: The AND gate is an electronic circuit that gives a high output (1) only if all its inputs are high.

- An AND gate accepts **two input** signals and gives only **one output**
- A dot (.) is used to show the AND operation i.e. A.B.

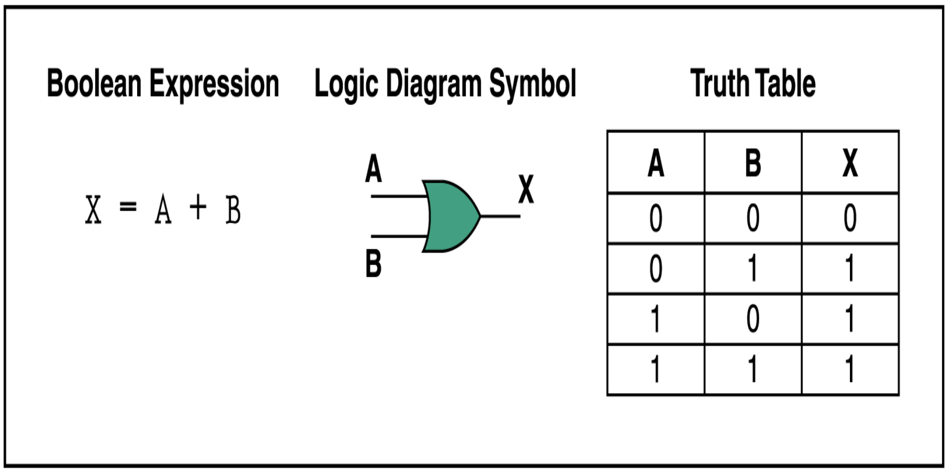


$X = A \cdot B$

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

OR GATE DEFINITION: The OR gate is an electronic circuit that gives a high output (1) if one or more of its inputs are high.

- **An OR gate accepts two input signals and gives only one output**
- **A plus (+) is used to show the OR operation.**



$X = A + B$

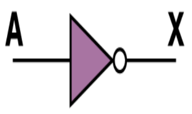
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

- **NOT GATE DEFINITION**
- **The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an inverter**
- **A NOT gate accepts one input value and produces one output value**

Boolean Expression

X = A'

Logic Diagram Symbol



Truth Table

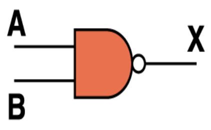
A	X
0	1
1	0

- NAND GATE Definition: The outputs of all NAND gates are high if any of the inputs are low.
- The NAND gate is the complement of AND gate
- It has two inputs and only one output
- The symbol is an AND gate with a small circle on the output. The small circle represents inversion.

Boolean Expression

X = (A · B)'

Logic Diagram Symbol



Truth Table

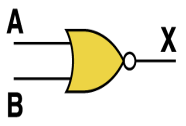
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

- NOR GATE Definition: The output of NOR gate is high if both of the inputs are low.
- The symbol is an OR gate with a small circle on the output. The small circle represents inversion.
- The NOR gate is the complement of OR gate
- It has two inputs and only one output

Boolean Expression

X = (A + B)'

Logic Diagram Symbol



Truth Table

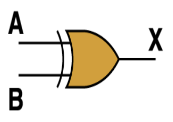
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

- **XOR GATE Definition: The 'Exclusive-OR' gate is a circuit which will give a high output ,When both the inputs are different. An encircled plus sign (+) is used to show the XOR operation.**
  - **An XOR gate produces**
  - **0 or LOW if its two inputs are the same**
  - **1 or HIGH if its inputs are different**

Boolean Expression

$X = A \oplus B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

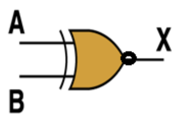
An XNOR gate produces

- **1 or HIGH if its two inputs are the same**
- **0 or LOW if its inputs are different**

Boolean Expression

$X = \overline{A \oplus B}$

Logic Diagram Symbol



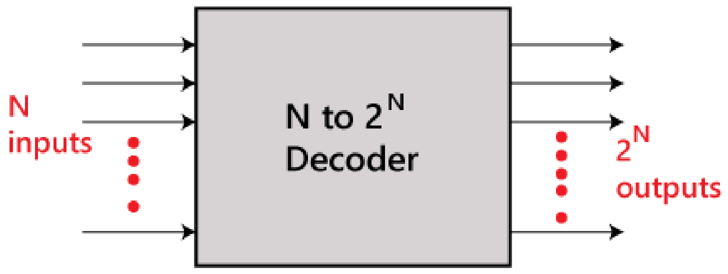
Truth Table

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

33. Explain 3x8 Decoder with circuit.

A decoder is also a combinational circuit as encoder but its operation is exactly reverse as that of the encoder.

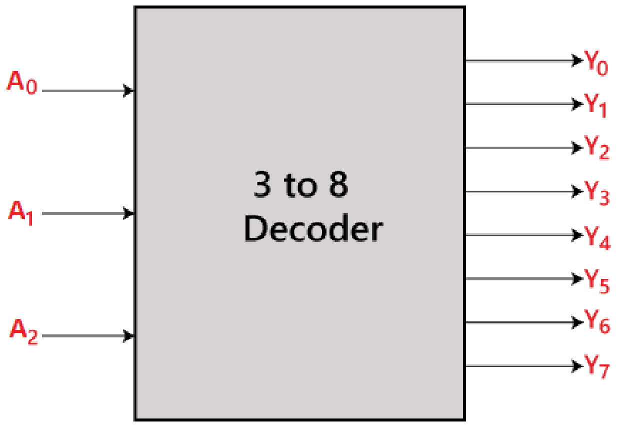
- A decoder is a device that generates the original signal as output from the coded input signal and converts n lines of input into 2n lines of output. An AND gate can be used as the basic decoding element because it produces a high output only when all inputs are high.



3 to 8 line decoder:

- The 3 to 8 line decoder is also known as **Binary to Octal Decoder**.
- In a 3 to 8 line decoder, there is a total of eight outputs, i.e., Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7 and three outputs, i.e., A_0, A1, and A_2.
- This circuit has an enable input 'E'. Just like 2 to 4 line decoder, when enable 'E' is set to 1, one of these four outputs will be 1.

The block diagram and the truth table of the 3 to 8 line encoder are given below.



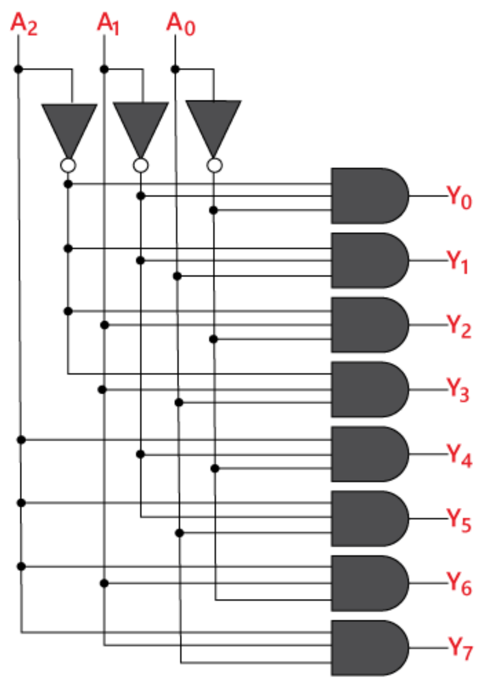
Truth table:

Inputs			Outputs							
A	B	C	D0	D1	D2	D3	D4	D5	D6	D7
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 logical expression of the term Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7 is as follows:

$Y_0=A_0'.A_1'.A_2'$
$Y_1=A_0.A_1'.A_2'$
$Y_2=A_0'.A_1.A_2'$
$Y_3=A_0.A_1.A_2'$
$Y_4=A_0'.A_1'.A_2$
$Y_5=A_0.A_1'.A_2$
$Y_6=A_0'.A_1.A_2$
$Y_7=A_0.A_1.A_2$

Logical circuit of the above expressions is given below:



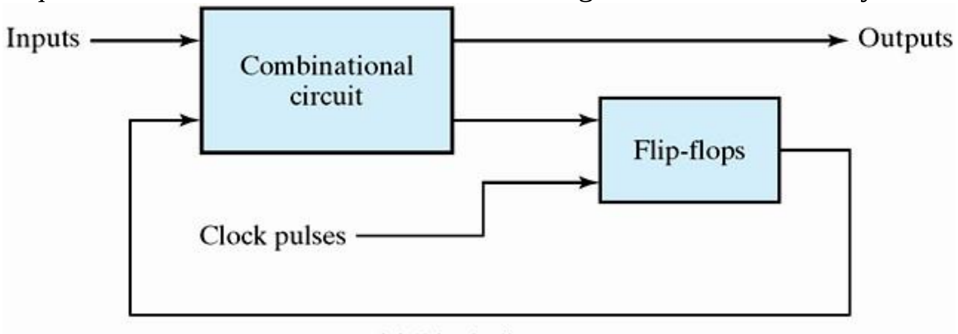
34. Explain combinational Circuit with example (REFER BOTH ADDERS)

35. Draw the logic circuit for the following equation and simplify using K MAP F(X,Y,Z) = X'Y'Z+X'YZ+XY'Z'+XY'Z

36. Explain about Sequential circuit with state table and state diagram for the following equation.

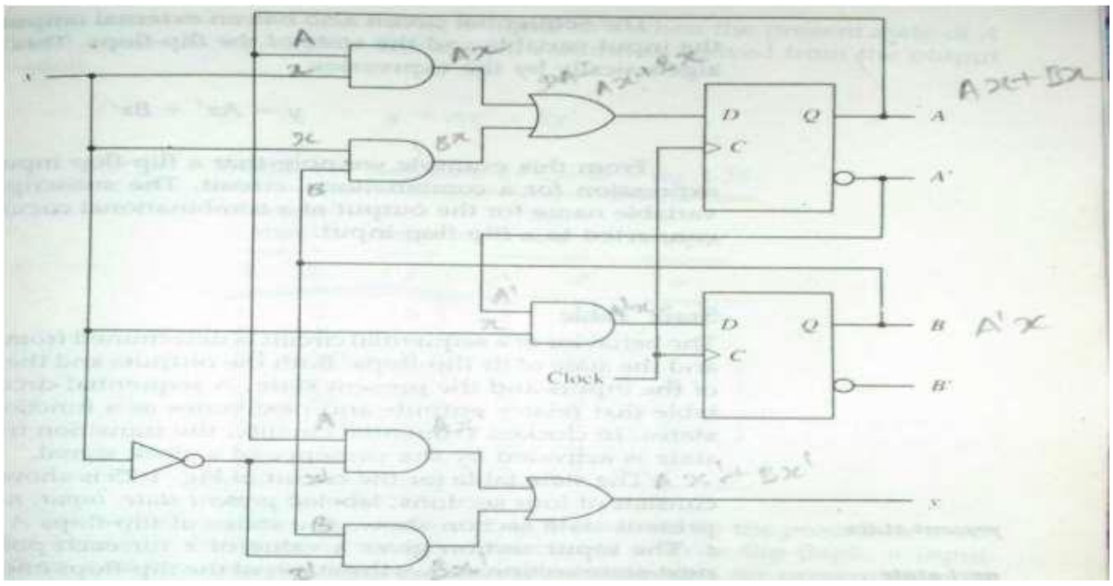
Input Equation:
DA=Ax + Bx
DB=A’x
• Output Equation:
Y=Ax’+Bx’

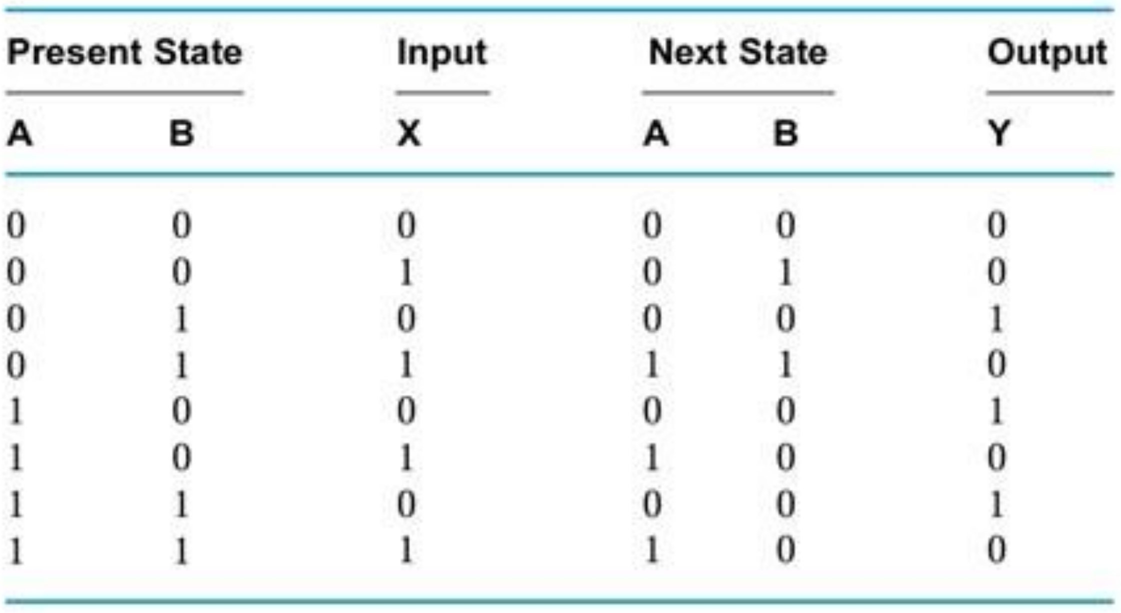
The sequential circuit is a special type of circuit that has a series of inputs and outputs. sequential circuit consist of combinational logic circuit and memory element



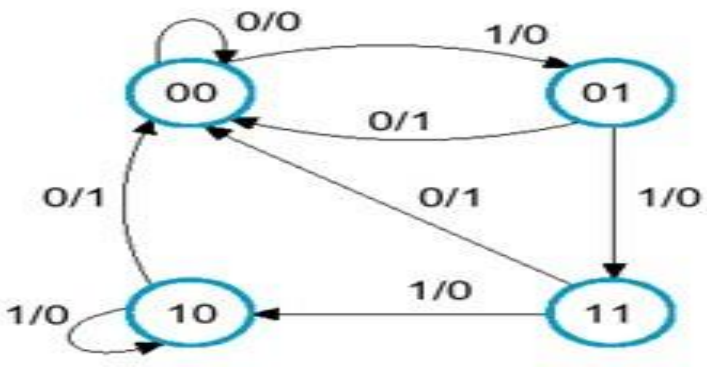
(a) Block diagram

- The outputs of the sequential circuits depend on **both the combination of present inputs and previous outputs.**
- It consists of 2 flip flops **DA and DB.**
- The external source x is applied to the logic gate, then the output of the logic gates is given as input to the flip flop, the output of flip flop is again given as an input to the logic gate.
- We can frame input output equation with help of circuit diagram
- **(KINDLY REFER CLASS NOTES)**





Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0



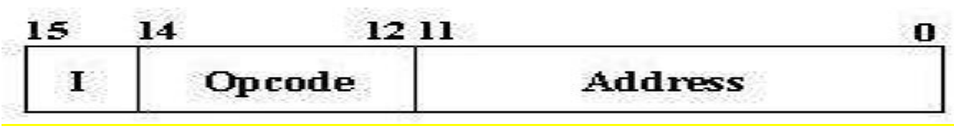
$$\begin{tikzcd}
00 \arrow[loop above, "0/0"] \arrow[r, bend left, "1/0"] & 01 \arrow[l, bend left, "0/1"] \arrow[d, "1/0"] \\
10 \arrow[u, "0/1"] \arrow[loop left, "1/0"] & 11 \arrow[l, "1/0"] \arrow[ul, "0/1"']
\end{tikzcd}$$

UNIT – II: Basic Computer Organization and Design

Instruction codes, Computer Registers, Computer Instructions and Instruction cycle. Timing and Control, Memory-Reference Instructions, Input-Output and interrupt. Central processing unit: Stack organization, Instruction Formats, Addressing Modes, Data Transfer and Manipulation, Complex Instruction Set Computer (CISC), Reduced Instruction Set Computer (RISC).

2 MARKS QUESTIONS.

1. **What is meant by Instruction code/ what is General Instruction format of a computer?/ What is basic computer Instruction format.**

- **Instruction Code : A group of bits that instruct the computer to perform a specific operation.**
- 

- **It is usually divided into parts.**

  Mode: Left most bit specify the addressing mode I

  Opcode: Next 3 bits specify operation code (opcode).

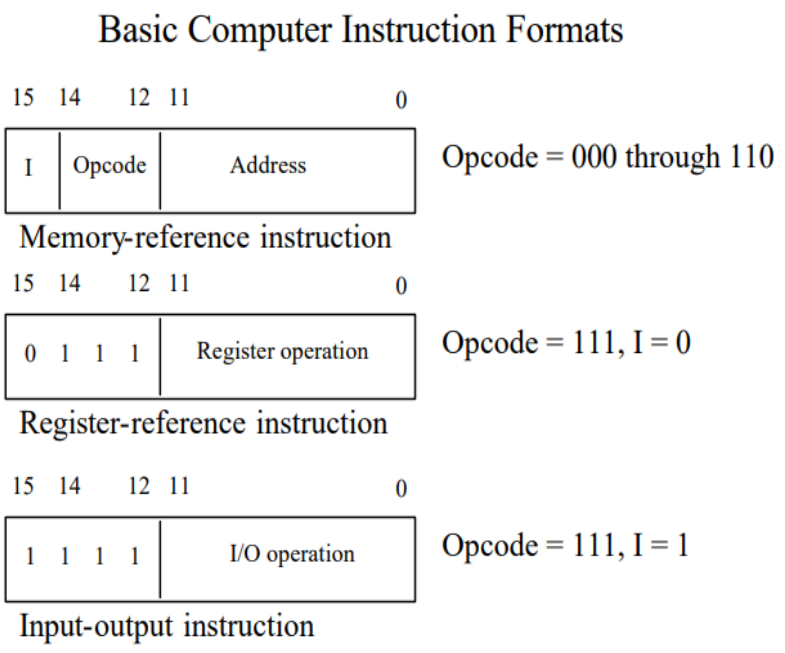
  Address field: First 12 bits (0-11) specify an address OR operation.

  – Mode: Mode field which specifies how operand is to be founded.
  – Operation Code (opcode): This specifies the operation to be performed like addition.
  – Address field which contain the location of operand. (register or memory location).
  –

2. **Explain 3 basic computer instruction code with format.**

   Types of basic computer instruction Format.

- **Memory - reference instruction.**
- **Register - reference instruction.**
- **Input-Output instruction**



- **Memory Reference – These instructions refer to memory address as an operand. The other operand is always accumulator.**
- **Register Reference – These instructions perform operations on registers rather than memory addresses.**
- **Input / Output – These instructions are for communication between computer and outside environment (IO devices).**

3. Define Interrupt and its types

The process of altering the flow of execution of a program is called interrupt.

Program interrupt refers to the transfer of program control from a currently running program to another service program as a result of an external or internal generated request.

Control returns to the original program after the service program is executed.

Types of interrupts

- **Internal Interrupts**
- **Software Interrupts**
- **External Interrupts**

Internal Interrupts

- Internal interrupts arise from illegal or erroneous use of an instruction or data.
- They are also called trap.
- Examples of interrupts caused by internal error conditions are register overflow, attempt to divide by zero, stack overflow, etc.

External Interrupts

- They come from I/O Devices, or from circuit monitoring or from any other external source.
- Examples for causing external interrupts are I/O device requesting transfer of data, I/O device finished transfer of data, elapsed time of an event.

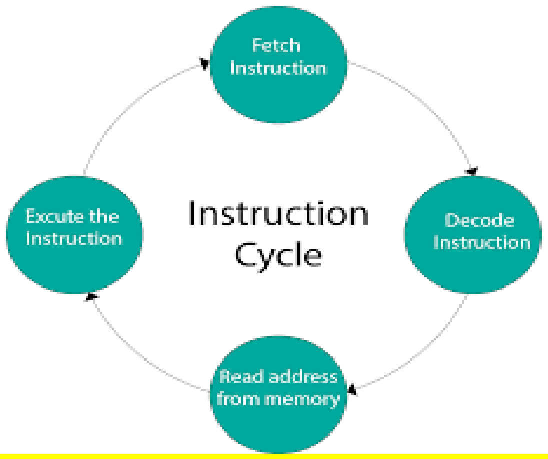
Software Interrupts

- Software interrupt is a special call instruction that behaves like an interrupt rather than a subroutine call. It can be used by the programmer to initiate an interrupt procedure at any desired point in the program.
- Example:
- Windows to linux/linux to windows
- User mode to admin mode/admin mode to user mode.

4. **What are the features of RISC AND CISC (Refer 6 Marks question)**
5. **What is Instruction cycle?**

Instruction Cycle: Processing required for complete execution of an instruction is called instruction cycle.

In Basic Computer, a machine instruction is executed in the following cycle:
1. Fetch an instruction from memory
2. Decode the instruction
3. Read the effective address from memory if the instruction has an indirect address
4. Execute the instruction
Upon the completion of step 4, control goes back to step 1 to fetch, decode and execute the next instruction.
This process is continued indefinitely until HALT instruction is encountered.

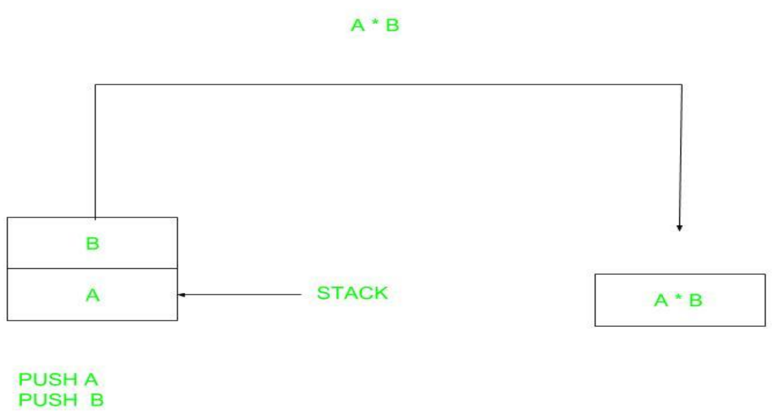


6. What is PC and AC? Explain the content it holds?

The program counter (PC) also has 12 bits and it holds the address of the next instruction to be read from memory after the current instruction is executed.
The accumulator (AC) register is a general purpose processing register. It is also called as Processor Register. The result of the operation is stored in Accumulator.

7. Explain stack organization.

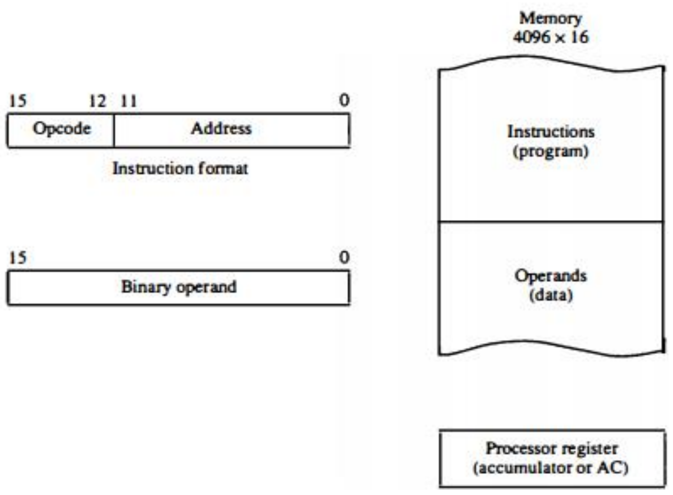
Stack is also known as the Last In First Out (LIFO) list. It is the most important feature in the CPU. It saves data such that the element stored last is retrieved first. The insertion operation is known as push operation and the deletion operation is known as pop operation. In a computer stack, these operations are simulated by incrementing or decrementing the TOS (TOP OF STACK) register. It is used in zero address Instruction format.



8. What is stored program organization?

A process of storing and retrieving data either from memory or register is called Stored program organization

Figure 5-1 Stored program organization.



9. Differentiate between direct and indirect addressing.

- **Direct Addressing: In this type mode(I=0) , the second part of the instruction specifies the address of the operand.**
  - **Example : 0 ADD 457**
- **Indirect addressing: In this type mode(I=1) , the second part of the instruction specifies the reference of the operand.**
  - **Example : 1 ADD 300**

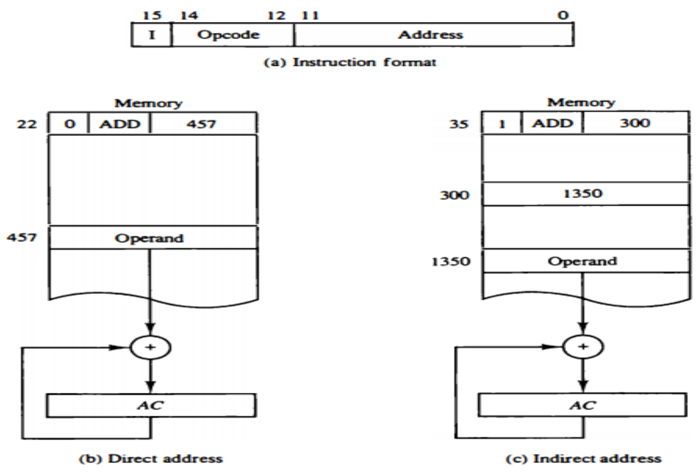


Figure Demonstration of direct and indirect address.

10. What is CPU organization?

Central Processing Unit (CPU) is the brain of a computer .Performs all the calculations & controls all the components of a computer. Carries out the instructions of a computer program, performs the basic arithmetical, logical, and input and output operations of the system.

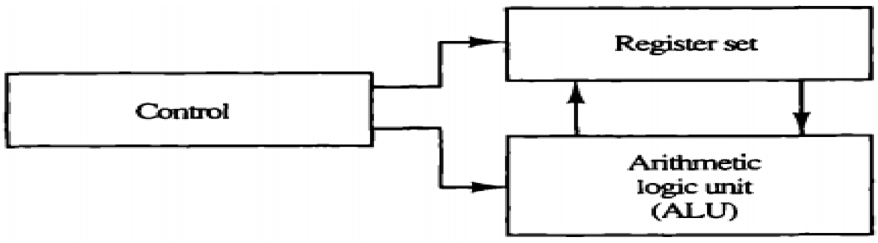


Figure 1 Major components of CPU.

The three components of the CPU are following,
1. Arithmetic Logic Unit
2. Control Unit
3. Registers

11.What is Arithmetic logic Unit?

Arithmetic Logic Unit (ALU) There is an electronic circuit in arithmetic logic unit which executes all arithmetic and logical operations.

It performs arithmetic calculations like as addition, subtraction, multiplication and division as well as comparisons. The ALU unit can compare numbers, letters, or special characters.

12.Define registers.

Registers/ Memory Unit

- Registers are temporary storage areas which are responsible for holding the data that is to be processed.
- They store the instructions and data in a processor.

There are some registers that are used for specific tasks.

1. Program Control Register (PC)
   - It holds the address of next instruction to be executed.
2. Memory Address Register (AR/MAR)
   - It holds the address of the active memory location.
3. Instruction Register(IR)
   - It holds the instruction being executed.
4. Accumulator
   - It holds the data under execution, intermediate results & result of operations

6 MARKS QUESTIONS.

13. Write a note on Registers

- **The program counter (PC) also has 12 bits and it holds the address of the next instruction to be read from memory after the current instruction is executed.**
- **The memory address register (AR) has 12 bits.**
- **The accumulator (AC) register is a general purpose processing register.**
- **The data register (DR) holds the operand read from memory.**
- **The instruction CODE read from memory is placed in the instruction register (IR).**
- **The temporary register (TR) is used for holding temporary data during the processing.**
- **Two registers are used for input and output. The input register (INPR) receives an 8-bit character from an input device.**
- **The output register (OUTR) holds an 8-bit character for an output device.**

SL NO	REGISTER SYMBOL	BITS	REGISTER NAME	FUNCTION
1	PC	12	PROGRAM COUNTER	HOLDS ADDRESS OF INSTRUCTION
2	AR	12	ADDRESS REGISTER	HOLDS address for memory
3	AC	16	ACCUMULATOR	Processor register
4	DR	16	DATA REGISTER	Holds memory operand
5	IR	16	INSTRUCTION RESGISTER	Holds Instruction code
6	TR	16	TEMPORARY REGISTER	Temporary data
7	INPR	8	INPUT REGISTER	Input character
8	OUTR	8	OUTPUT REGISTER	Output character

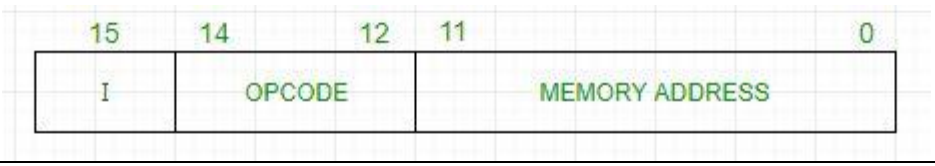
14. Explain about Basic Computer Instructions.

Types of basic computer Instructions:

- Memory - reference instruction.
- Register - reference instruction.
- Input-Output instruction

Memory Reference – These instructions refer to memory address as an operand. The other operand is always accumulator.

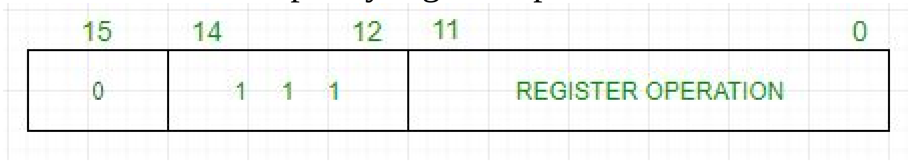
- Specifies 12-bit address,
- 3-bit opcode (other than 111) and 1-bit addressing mode for direct and indirect addressing.



Memory Reference Instruction	
AND	Performs logical AND operation with AC content
LDA	Load Memory Word to AC
ADD	ADD memory word to AC
STA	Store the content of AC to memory
BUN	Branched Unconditionally
BSA	Branch and save return address
ISZ	Increment and skip if zero

Register Reference – These instructions perform operations on registers rather than memory addresses.

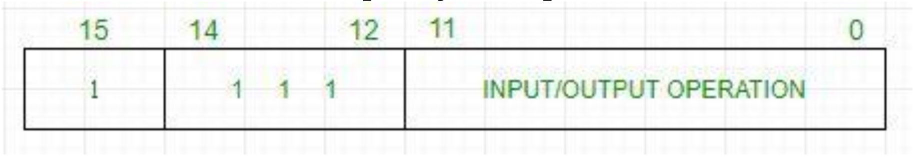
- The IR(14 – 12) is 111 (differentiates it from memory reference) and
- IR(15) is 0 (differentiates it from input/output instructions).
- The rest 12 bits specify register operation.



REGISTER REFERENCE INSTRUCTIONS	MEANING
CLA	Clear accumulator content
CLE	Clear E bit of Accumulator
CMA	Complement accumulator content
CME	Complement E bit of Accumulator
SNA	Skip to next instruction if AC is negitive value
SPA	Skip to next instruction if AC is positive value
HLT	Halt the computer
SZE	Skip next instruction if E is 0

Input/ Output Reference – These instructions are for communication between computer and outside environment (IO devices).

- The IR(14 – 12) is 111 (differentiates it from memory reference) and
- IR(15) is 1 (differentiates it from register reference instructions).
- The rest 12 bits specify I/O operation.



INPUTOUTPUT INSTRUCTION

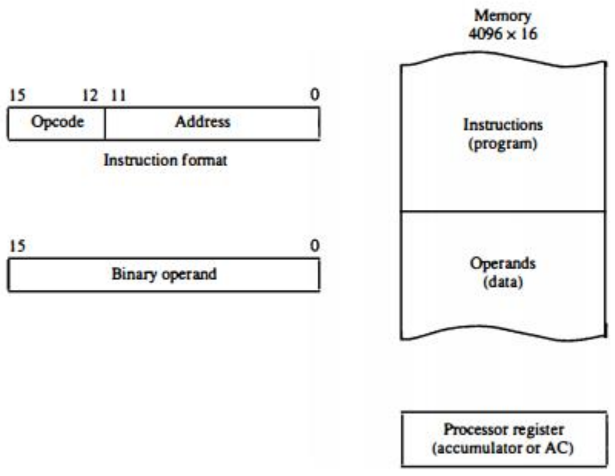
IO reference instructions	
INP	Input character to AC
OUT	Output character from AC
SKI	Skip on input flag
SKO	Skip on output flag
ION	Interrupt on
IOF	Interrupt off

15. Explain any 6 types of addressing modes.(refer 10 Marks question)

16. Explain Stored program organization.

- **A process of storing and retrieving data either from memory or register is called Stored program organization**

Figure 5-1 Stored program organization.



3 types of stored program organization

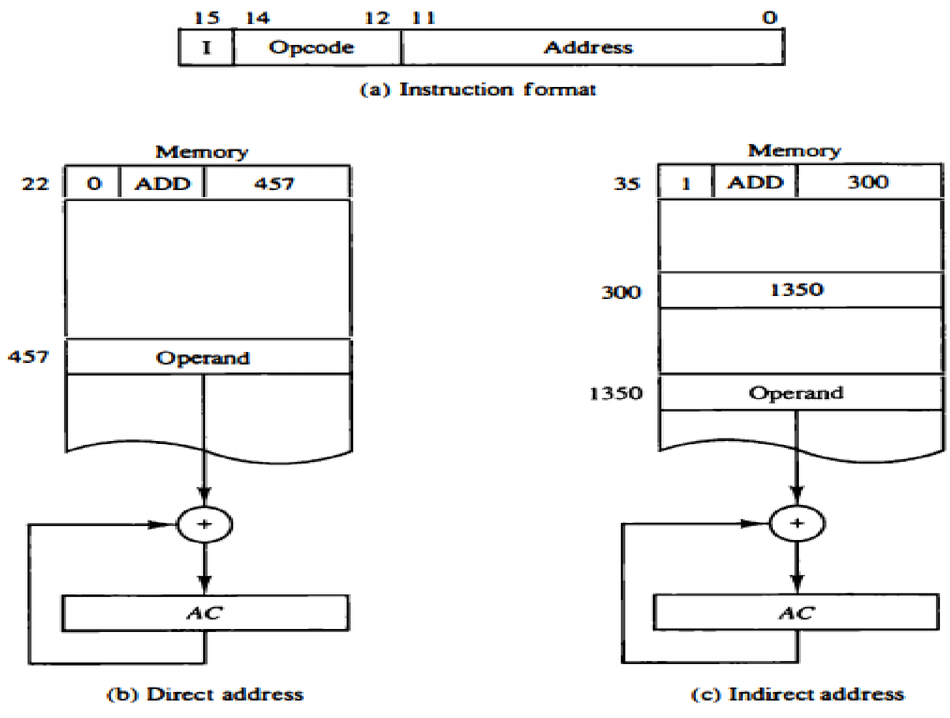


Figure Demonstration of direct and indirect address.

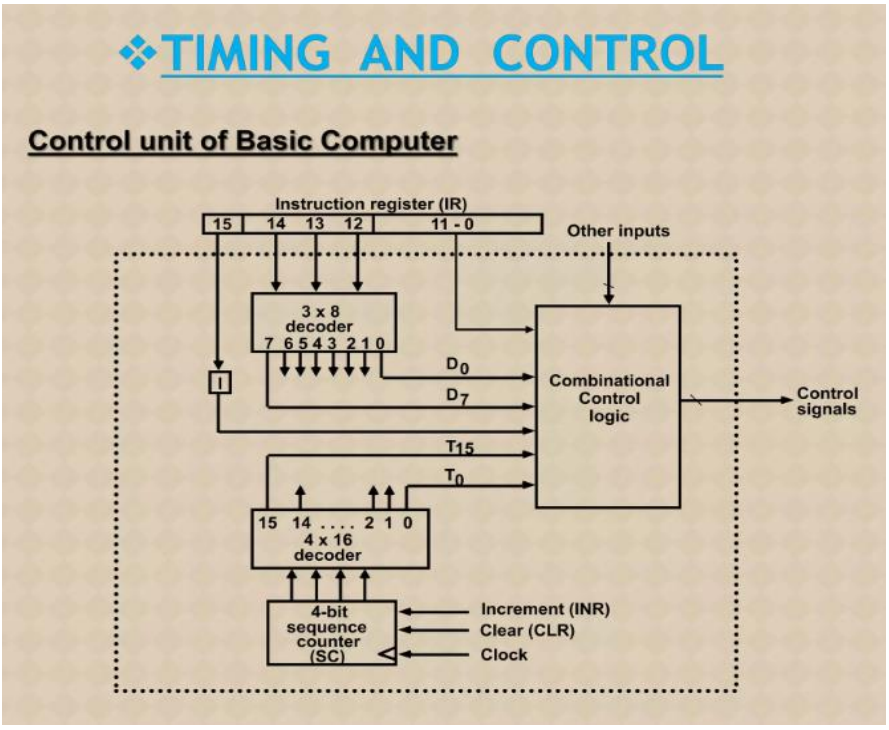
–

- **Immediate Addressing: In this second part of the instruction holds data rather than address to perform operation specified in the instruction**

- **Example: ADI 5**
  - **AC← AC+5**
- **Direct Addressing: In this type mode(I=0) , the second part of the instruction specifies the address of the operand.**
  - **Example : 0 ADD 457**
- **Indirect addressing: In this type mode(I=1) , the second part of the instruction specifies the reference of the operand.**
  - **Example : 1 ADD 300**

17. Write a note on Timing and control unit.

- **Block Diagram consists of**
- **2 Decoders- 3x8 decoder, 4x16 decoder**
- **4 bit Sequence counter, control logic gates**
  - **An instruction read from memory is placed in the instruction register (IR).**
  - **Instruction register is divided into three parts:**
    - **i) the I bit**
    - **ii) the operation code**
    - **iii) bits 0 through 11**



- **Opcode in bits 12 - 14 are decoded with a 3X8 decoder. Eight outputs of the decoder are designated by the symbols D0 through D7 .**

- **Bit 15 of the instruction is transferred to a flip flop designated by symbol I.**
- **Bits 0 - 11 are applied to the control logic gates.**
- **The 4 bit sequence counter can count in binary from 0 through 15.**
- **The outputs of the counter are decoded into 16 timing signals T0 through T15.**

18. Explain Data transfer Instructions with examples.

- **Data Transfer Instructions:** Data transfer instructions is used to transfer of data from one location to another without changing the information content.
  The common transfers may be between memory and processor registers, between processor registers and input/output.

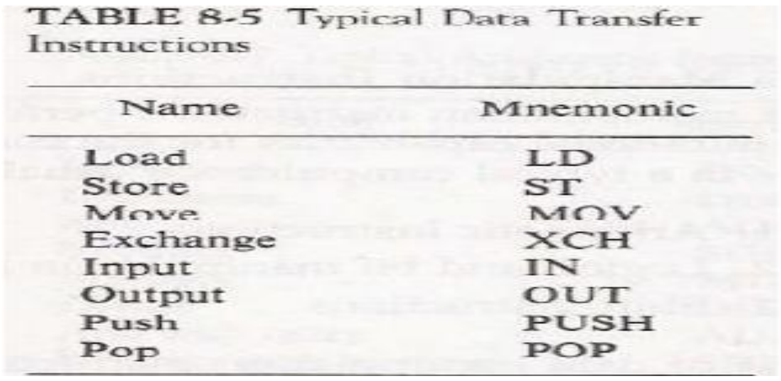


TABLE 8-5 Typical Data Transfer Instructions

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
Pop	POP

.
l. Load (LD): The *load* instruction is used to transfer data from memory in to a processor register, usually an accumulator.
AC← MEMORY

2. STORE (ST): The *store* instruction is used to transfer data from a processor register into memory.
AC→ MEMORY

3. MOVE (MOV): The *move* instruction has been used in computers with multiple CPU registers to transfer data from one register to another and also between CPU registers and memory or between two memory words.

4. EXCHANGE (XCHG): The *exchange* instruction swaps information between two registers or a register and a memory word.

5. INPUT (IN) & OUTPUT (OUT): The *input* and *output* instructions transfer data among processor registers and input or output terminals.
6. PUSH & POP: The *push* and *pop* instructions transfer data between processor registers and a memory stack.

19. Explain Arithmetic operation instructions with example

Arithmetic instructions

The four basic arithmetic operations are addition, subtraction, multiplication and division.

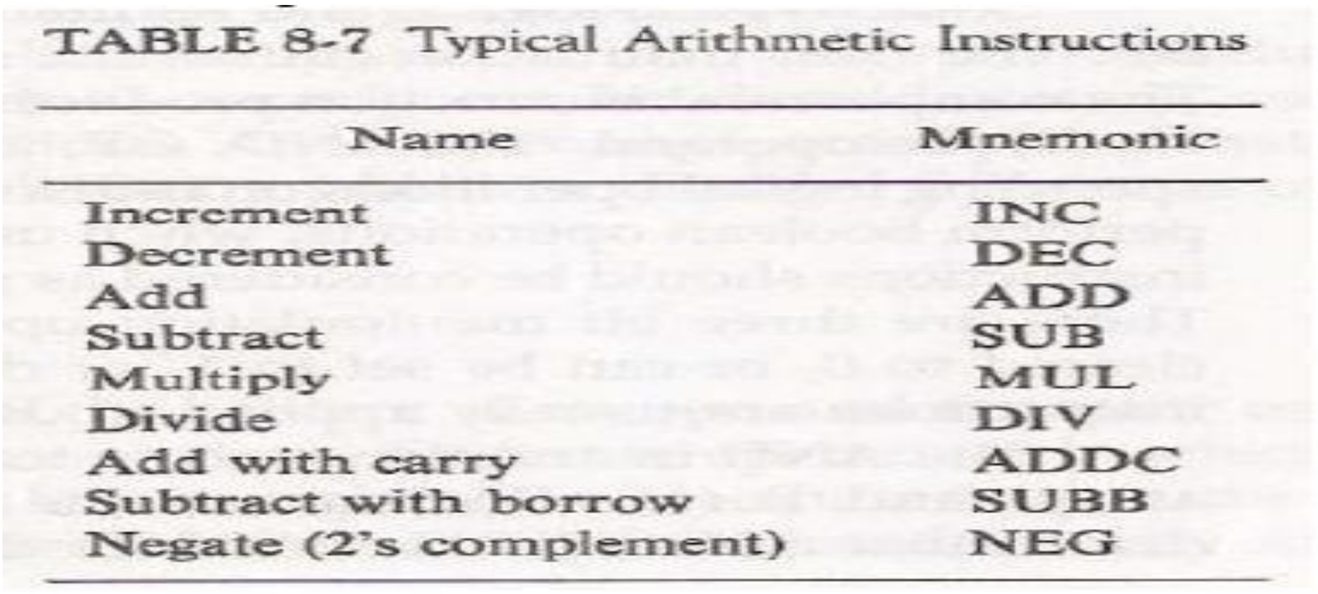


TABLE 8-7 Typical Arithmetic Instructions

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with carry	ADDC
Subtract with borrow	SUBB
Negate (2's complement)	NEG

1. The increment instruction adds 1 to the value stored in a register or memory word. **A number with all 1's, when incremented, produces a number with all 0's.**

2. The decrement instruction subtracts 1 from a value stored in a register or memory word. **A number with all 0's, when decremented, produces number with all 1's.**

Add, subtract, multiply, and divide instructions may use different types of data.

The data type assumed to be in processor register (AC) during the execution of these arithmetic operations is defined by an operation code.

– ADD: The data(register/ memory) is added with AC and result is stored AC

- SUB: The Register/memory content is SUBTRACTED FROM AC and result is stored AC
- Multiplicand will be in an accumulator, depending upon the size of the multiplicand and the multiplier and the generated product is stored in two registers depending upon the size of the operands.
- The division operation generates two elements - a **quotient** and a **remainder**. The dividend is in an accumulator.
- The negate instruction forms the 2's complement number(1'S Complement+1), effectively reversing the sign of an integer when represented it signed-2's complement form.

20. Explain Logic operation instructions with example

Logical instructions perform binary operations on bits.
•They are useful for manipulating individual bits or a group of that represent binary-coded information.
• By proper application of the logical instructions it is possible to change bit values, to clear a group of bits, or to insert new bit values into operands stored in register memory words.

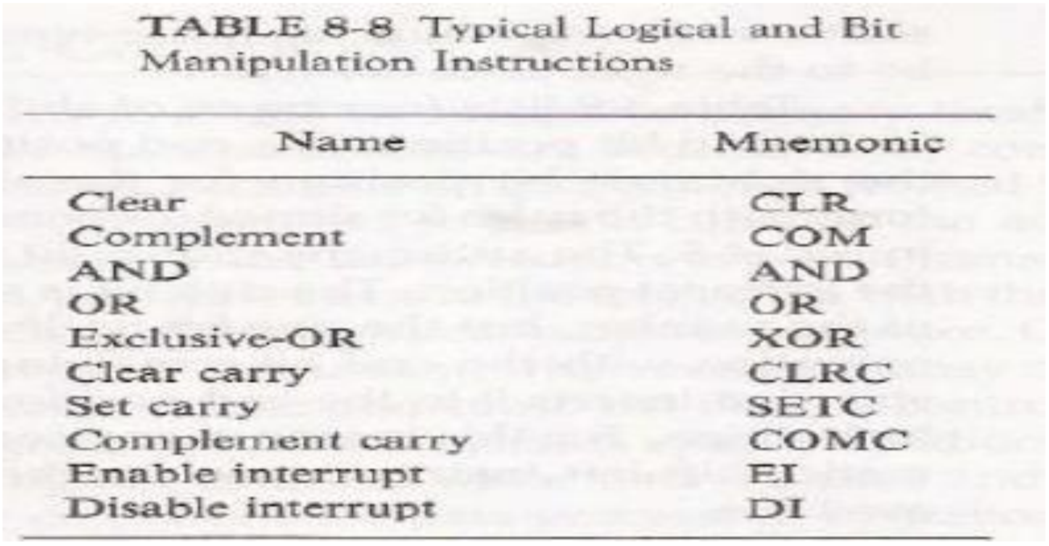


TABLE 8-8 Typical Logical and Bit Manipulation Instructions

Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-OR	XOR
Clear carry	CLRC
Set carry	SETC
Complement carry	COMC
Enable interrupt	EI
Disable interrupt	DI

LOGICAL

1. Clear (CLR): The clear instruction causes the specified operand to be replaced by 0's.
2. COMPLEMENT (COM): The complement instruction produces the 1's complement by inverting all bits of the operand.
3. The AND, OR, and XOR instructions produce the corresponding logical operations on individual bits of the operands•

21. Explain Bit Manipulation operation instructions with example

Bit Manipulation:

The logical instructions can also be used to performing bit manipulation operations.

•There are three bit manipulation operations possible:

- a selected bit can cleared to 0,(RESET)
- can be set to 1,(SET)
- Can be complemented. (COMPLEMENT)

- RESET: The AND instruction is used to clear a bit or a selected group of bits of an operand.
- SET: The OR instruction is used to set a bit or a selected group of bits of an operand.
- COMPLEMENT: Similarly, the XOR instruction is used to selectively complement bits of an operand.

Other bit manipulations instructions are included in above table perform the operations on individual bits such as a carry can be cleared, set, or complemented.

22. Write a note on CISC and RISC.

CISC complex instruction set computer.
A computer with a large number of instructions is classified as complex instruction set computer.

Characteristics

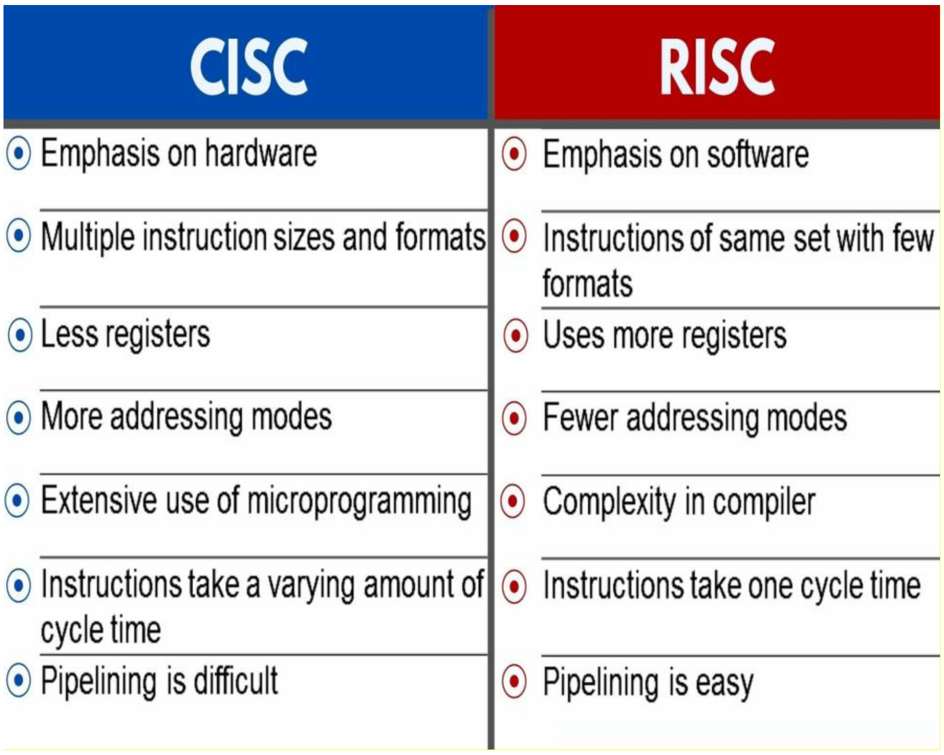
- 1. A large number of instructions- typically from 100 to 250 instructions.
- 2. Some instructions that perform specialized tasks and are used infrequently
- 3. A large variety of addressing modes- typically from 5 to 20 different modes.
- 4. Variable length instruction format
- 5. Instruction that manipulate operands in memory.

RISC involves an attempt to reduce execution time by simplifying the instruction set of the computer.

Characteristics

23. Relatively few instructions
24. Relatively few addressing modes
3. Memory access limited to load and store instructions
4. All operations done within the registers of the CPU

5. Fixed length, easily decoded instruction format
6. Single cycle instruction execution
7. Hardwired rather than micro programmed control



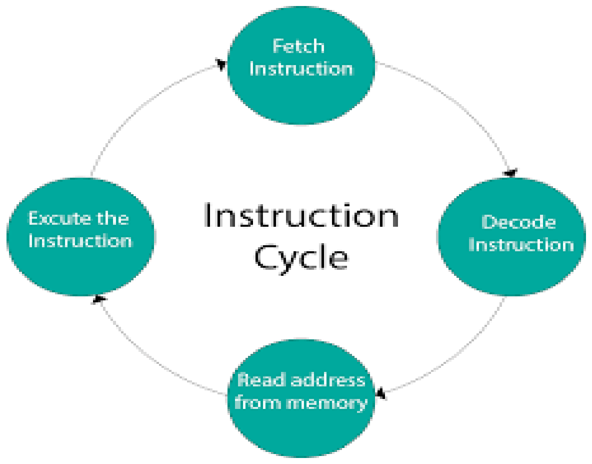
CISC	RISC
Emphasis on hardware	Emphasis on software
Multiple instruction sizes and formats	Instructions of same set with few formats
Less registers	Uses more registers
More addressing modes	Fewer addressing modes
Extensive use of microprogramming	Complexity in compiler
Instructions take a varying amount of cycle time	Instructions take one cycle time
Pipelining is difficult	Pipelining is easy

10 MARKS QUESTIONS.

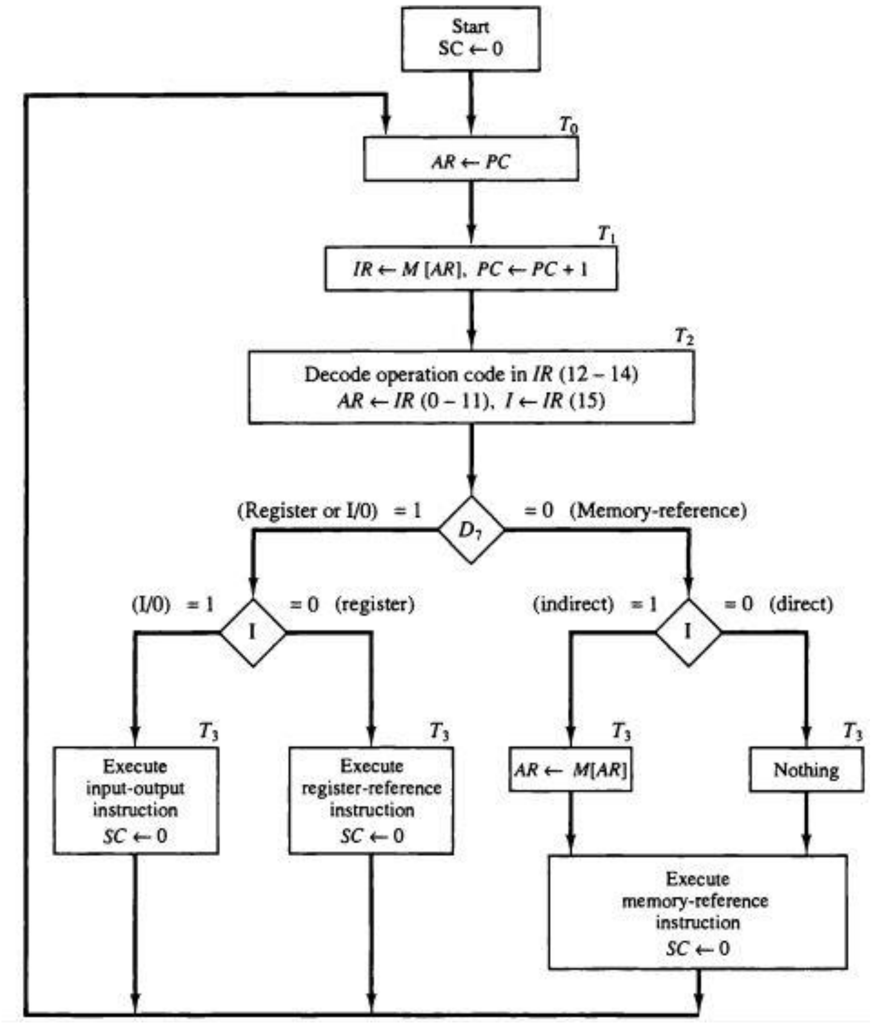
25. Explain Instruction cycle with flowchart.

Instruction Cycle: Processing required for complete execution of an instruction is called instruction cycle.

In Basic Computer, a machine instruction is executed in the following cycle:
1. Fetch an instruction from memory
2. Decode the instruction
3. Read the effective address from memory if the instruction has an indirect address
4. Execute the instruction
Upon the completion of step 4, control goes back to step 1 to fetch, decode and execute the next instruction.
This process is continued indefinitely until HALT instruction is encountered.



Instruction cycle flowchart



D’7I’T3: → NOTHING

D’7IT3: → AR←M[AR]

D7I’T3: EXECUTE register reference instruction

D7IT3: execute Input-out instruction

Type of instruction during instruction cycle can be determined by a special bit called D7 and the Mode value I are specify using following Notations
D’7 IT3: AR <- M [AR]
D’7 I’ T3: Nothing
D7 I’ T3: Execute a register-reference instruction
D7 I T3: Execute an input-output instruction

Then, among decoded, D7 determines which type of instruction.
i) If D7 = 1, it will be either register-reference or input-output instruction.
  a) If I = 1, input-output instruction is executed during T3.
  b) If I = 0, register-reference instruction is executed during T3.
ii) If D7 = 0, it will be memory-reference instruction.
  a) If I = 1, indirect addressing mode instruction during T3.
  b) If I = 0, direct addressing mode instruction during T3.
The SC is reset after executing each instruction.

26. Explain Instruction formats based on address by considering following example x=(a+b)*(c+d)

CLASSIFICATION OF INSTRUCTION BASED ON NUMBER OF ADDRESS

On the basis of number of address instruction are classified as:

- **Three address instructions**
- **Two address instruction**
- **one address instructions**
- **zero address instructions**

Three address instructions: Computers with three-address instruction formats can use each address field to specify either a processor register or a memory operand.

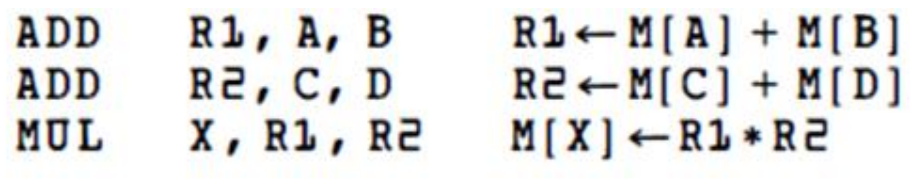
- **It is also called General register organization.**

  - **SYNTAX**

– OPCODE	– ADDRESS1	– ADDRESS2

  - **Example: ADD R1, R2, R3**
  - **Micro operation: R1 <- R2 + R3**
  - **Content of R3 is added to content of R2 and the sum is stored in R1**

  - **EVALUATE X=(A+B)*(C+D)**



```
ADD   R1, A, B     R1←M[A] + M[B]
ADD   R2, C, D     R2←M[C] + M[D]
MUL   X, R1, R2    M[X]←R1*R2
```

  - **The symbol M [A] denotes the operand at memory address A.**
  - **The value of A and B is added and the result is stored in R1.**
  - **The value of C and D is added and the result is stored in R2.**
  - **The value of R1 and R2 is multiplied and result is stored in memory address of X.**

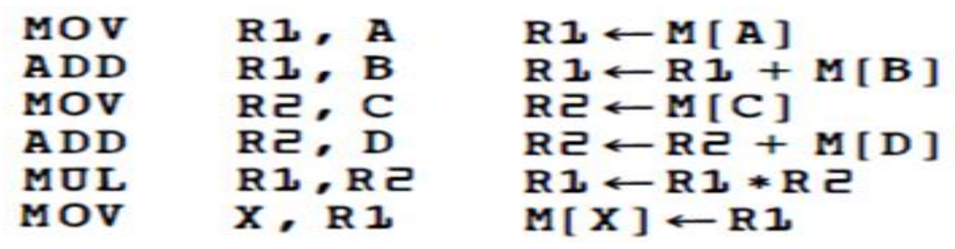
Two address instructions:

Each instruction consists of one opcode and two address field.

- Each address field can specify either a processor register or memory locations.
- Assumes that the destination address is the same as that of the first operand.

OPCODE	ADDRESS1	ADDRESS2

• Instruction: ADD R1, R2
Micro operation: R1 <-R1 + R2
.
EVALUATE X=(A+B)*(C+D)



```
MOV     R1, A     R1 ← M[A]
ADD     R1, B     R1 ← R1 + M[B]
MOV     R2, C     R2 ← M[C]
ADD     R2, D     R2 ← R2 + M[D]
MUL     R1,R2     R1 ← R1 * R2
MOV     X, R1     M[X] ← R1
```

The MOV instruction transfers or moves data to and from memory and processor register

Syntax MOV destination, source;

. From the memory location the value of A is transferred to register R1

- The value of B is added to content of R1 and result is stored in R1
- From the memory location the value of C is transferred to register R2
- The value of D is added to content of R2 and result is stored in R2
- The content of R2 is multiplied with content of R1 and result is stored in R1
- The content of R2 is transferred to X and it is stored in memory location

.One address instructions

Each instruction consists of one opcode and one address field. One address can be a register name or memory address.

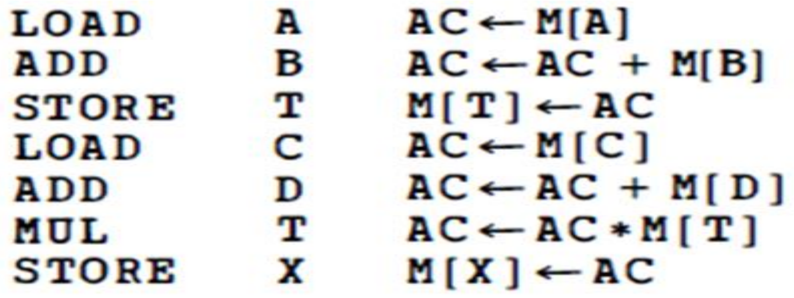
- It Also called SINGLE ACCUMULATOR ORGANIZATION
- It uses AC register for all data Manipulation and *AC* contains the result of all operations.

SYNTAX

OPCODE	**ADDRESS**

• Instruction: ADD X
Micro operation: AC ← AC + M[X]

.EVALUATE X = (A+B)*(C+D)



```
LOAD     A     AC ← M[A]
ADD      B     AC ← AC + M[B]
STORE    T     M[T] ← AC
LOAD     C     AC ← M[C]
ADD      D     AC ← AC + M[D]
MUL      T     AC ← AC * M[T]
STORE    X     M[X] ← AC
```

All operations are performed between the AC register and a memory operand.

- T is the address of a temporary memory location required for storing the intermediate result.

Zero address instruction

Each instruction consists of one opcode and no address field (zero address).

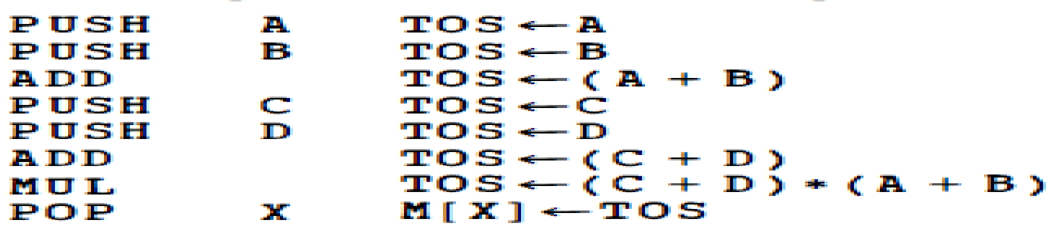
- A stack-organized computer does not use an address field for the instructions. Arithmetic operation pops two operands from the stack and pushes the result.

• Also called stack organization

Syntax

OPCODE

Evaluate (A+B)*(C+D)



```
PUSH    A     TOS ← A
PUSH    B     TOS ← B
ADD           TOS ← (A + B)
PUSH    C     TOS ← C
PUSH    D     TOS ← D
ADD           TOS ← (C + D)
MUL           TOS ← (C + D) * (A + B)
POP     X     M[X] ← TOS
```

- TOS indicates Top of the Stack.
- The name "zero-address” is given to this type of computer because of the absence of an address field in the computational instructions.

27. Explain about addressing modes and its types.

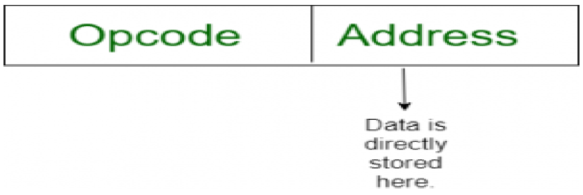
- **The addressing mode gives or indicates a rule to identify the operand location.**

- **Different addressing modes:**
  - **1. Implied mode**
  - **2. Immediate mode**
  - **3. Register mode**
  - **4. Register indirect mode**
  - **5. Auto increment or auto decrement mode**
  - **6. Direct address mode**
  - **7. Indirect address mode**
  - **8. Relative address mode**
  - **9. Indexed addressing mode**

- **10.Base register index mode**

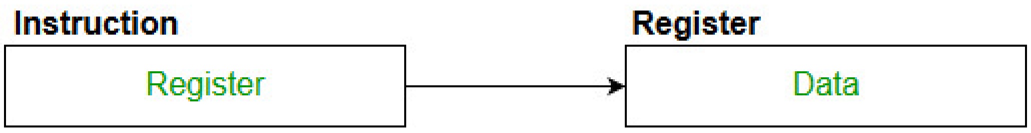
- **Implied mode:: In implied addressing the operand is specified in the instruction itself.**
  Example:
  - **CLC (used to reset Carry flag to 0)**
  - **CMA(Complement Accumulator)**

- **Immediate addressing mode: In this mode data is present in address field of instruction.**

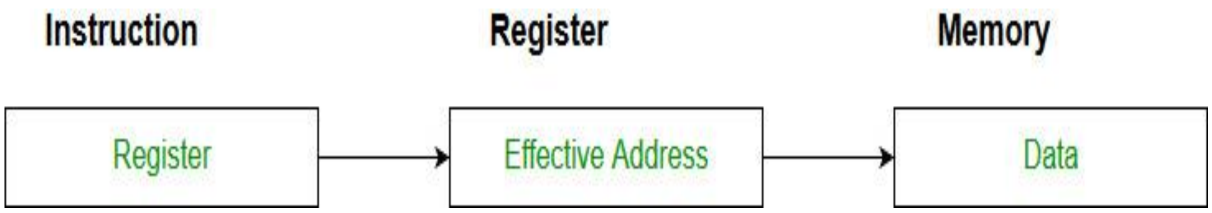


  - **MVI A, 35H (move the data 35H into register-A)**

- **Register mode: In register addressing the operand is placed in general purpose registers.**
  - **MOV A,C (move the contents of C register to A register)**



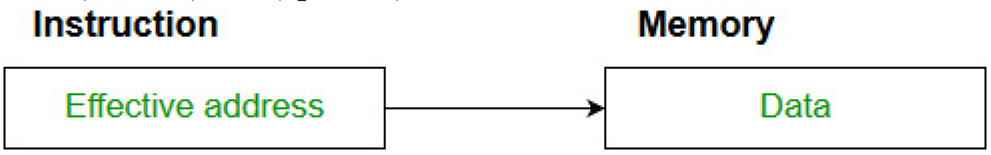
- **Register Indirect mode: In this addressing the operand's are stored Indirectly.** ***Here two register reference is required to access the data.***



- **Auto increment and decrement: The register is incremented or decremented after its value is used to access memory. When the address stored in the register refers to a table of data in memory, it is necessary to increment or decrement the register.**

- **Direct Addressing: In this mode the effective address is equal to the address part of the instruction. The operands resides in memory and its address is given directly by the address field of the instruction Second part of instruction holds the address of operand**

- **Ex- ADD 1000**
- **1000 (address)->50 (operand)**



- **Indirect Addressing: In this mode the address field of the instruction gives the address where the effective address is stored in memory**
  - **Effective address = address part of instruction + content of CPU register**
  - **Ex- ADD 1000(address) 100-CPU reg**
  - **1000(address)- 1100(address) 1100(address)- 50(operand)**

- **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**
  - **Effective address = PC + address part of the instruction**

- **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.**
  - **Effective address = Index Register value + address part of the instruction**

- **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.**
  - **Effective address = Base Register value + address part of the instruction**

28. Describe Data transfer and manipulation Instructions with example

29.**Data Transfer Instructions:** Data transfer instructions is used to transfer of data from one location to another without changing the information content.
The common transfers may be between memory and processor registers, between processor registers and input/output.

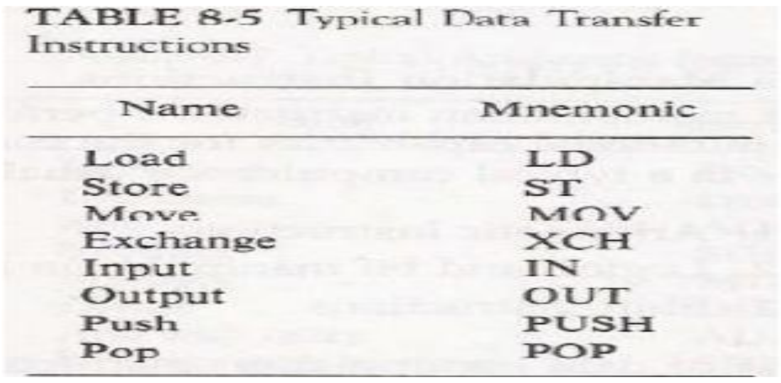


TABLE 8-5 Typical Data Transfer Instructions

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
Pop	POP

.
l. Load (LD): The *load* instruction is used to transfer data from memory in to a processor register, usually an accumulator.
AC← MEMORY

2. STORE (ST): The *store* instruction is used to transfer data from a processor register into memory.
AC→ MEMORY

3. MOVE (MOV): The *move* instruction has been used in computers with multiple CPU registers to transfer data from one register to another and also between CPU registers and memory or between two memory words.

- **Data Manipulation Instructions:**

  Data manipulation instructions perform operations on data and provide the computational capabilities for the computer.

The data manipulation instructions in a typical computer are usually divided into three basic types:

1. Arithmetic instructions
2. Logical and bit manipulation instructions
3. Shift instructions

• **Arithmetic instructions**

The four basic arithmetic operations are addition, subtraction, multiplication and division.

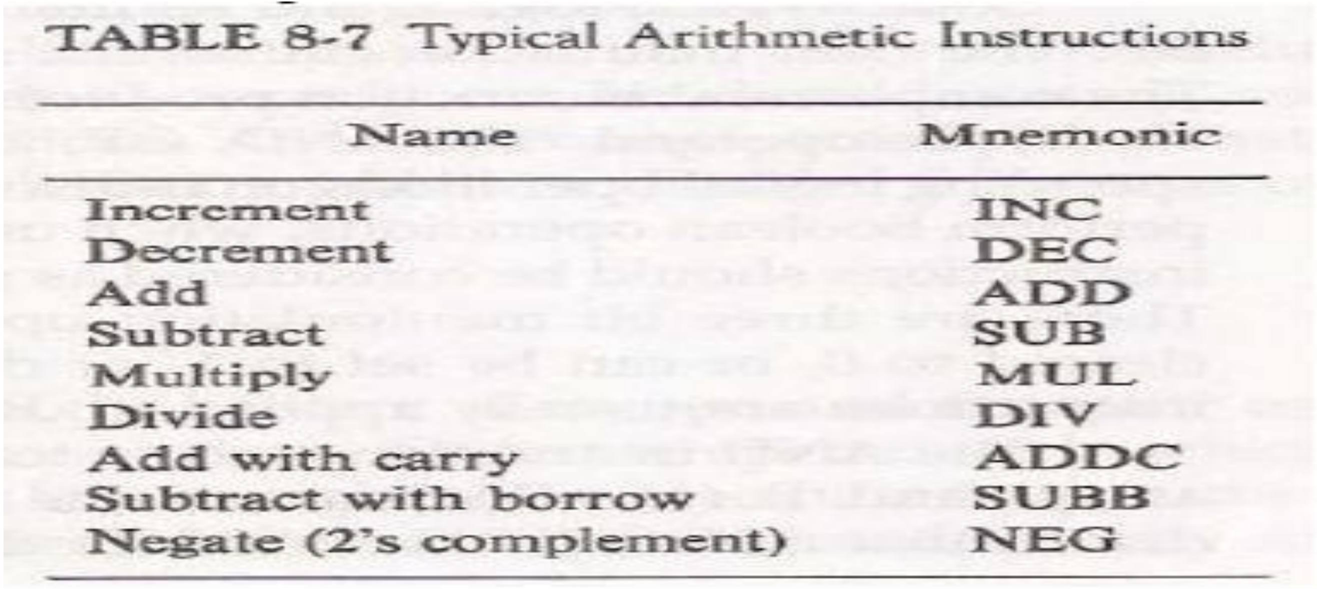


TABLE 8-7 Typical Arithmetic Instructions

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with carry	ADDC
Subtract with borrow	SUBB
Negate (2's complement)	NEG

3. The increment instruction adds 1 to the value stored in a register or memory word. **A number with all 1's, when incremented, produces a number with all 0's.**

4. The decrement instruction subtracts 1 from a value stored in a register or memory word. **A number with all 0's, when decremented, produces number with all 1's.**

Add, subtract, multiply, and divide instructions may use different types of data.

The data type assumed to be in processor register (AC) during the execution of these arithmetic operations is defined by an operation code.

- ADD: The data(register/ memory) is added with AC and result is stored AC
- SUB: The Register/memory content is SUBTRACTED FROM AC and result is stored AC

Logical instructions perform binary operations on bits.

•They are useful for manipulating individual bits or a group of that represent binary-coded information.

• By proper application of the logical instructions it is possible to change bit values, to clear a group of bits, or to insert new bit values into operands stored in register memory words.

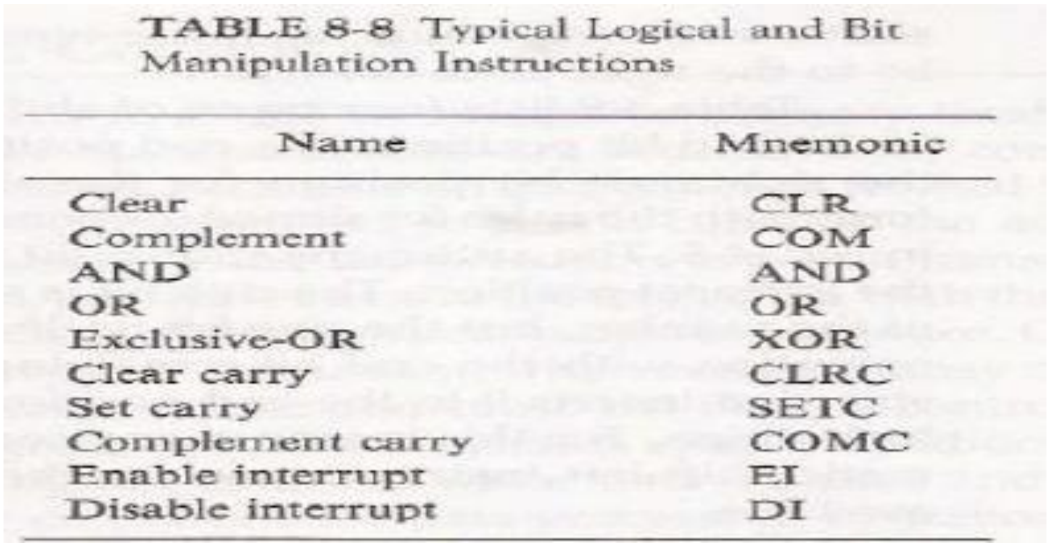


TABLE 8-8 Typical Logical and Bit Manipulation Instructions

Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-OR	XOR
Clear carry	CLRC
Set carry	SETC
Complement carry	COMC
Enable interrupt	EI
Disable interrupt	DI

LOGICAL

1. Clear (CLR): The clear instruction causes the specified operand to be replaced by 0’s.
2. COMPLEMENT (COM): The complement instruction produces the 1's complement by inverting all bits of the operand.
3. The AND, OR, and XOR instructions produce the corresponding logical operations on individual bits of the operands•

Shift Instructions:

Shifts are operations in which the bits of a word are moved to the left or right. Shift instructions may specify logical shifts, arithmetic shifts, and rotate-type operations.

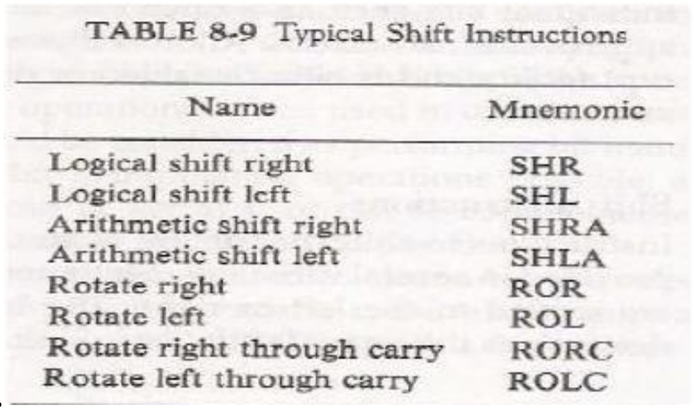


TABLE 8-9 Typical Shift Instructions

Name	Mnemonic
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right through carry	RORC
Rotate left through carry	ROLC

.

30. Explain Shift operation instructions with example

Shift Instructions:

Shifts are operations in which the bits of a word are moved to the left or right. Shift instructions may specify logical shifts, arithmetic shifts, and rotate-type operations.

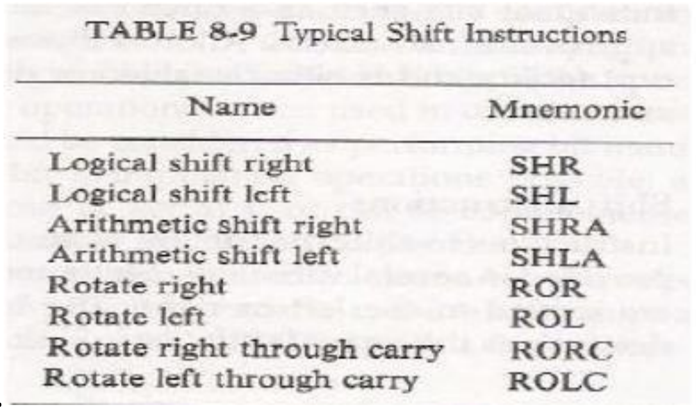


TABLE 8-9 Typical Shift Instructions

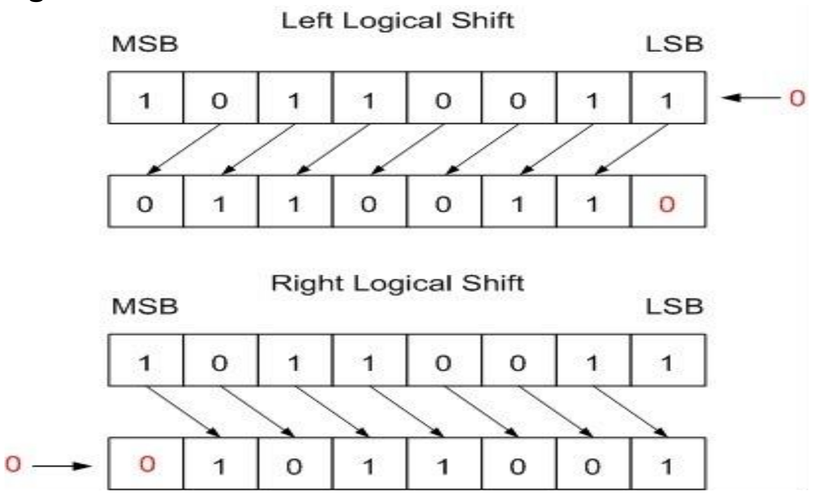
Name	Mnemonic
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right through carry	RORC
Rotate left through carry	ROLC

.

a) Logical Shift

1. ***Left Logical Shift*** (SHL) moves each bit to the left by one. The vacant least significant bit (LSB) is filled with zero and the most significant bit (MSB) is discarded.

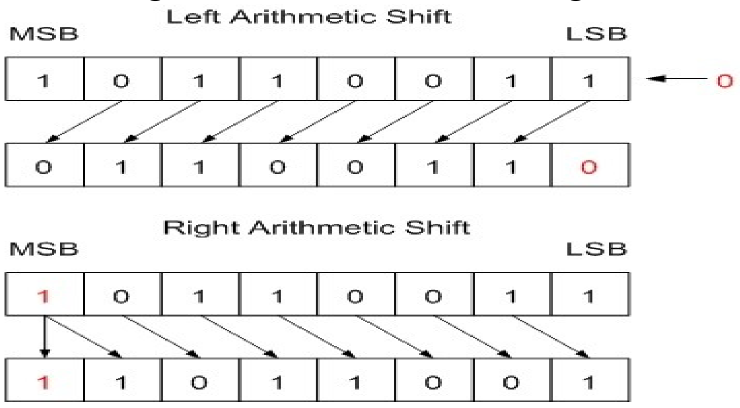
2. ***Right Logical Shift (SHR)*** moves each bit to the right by one. The least significant bit is discarded and the vacant MSB is filled with zero.



b) Arithmetic Shift

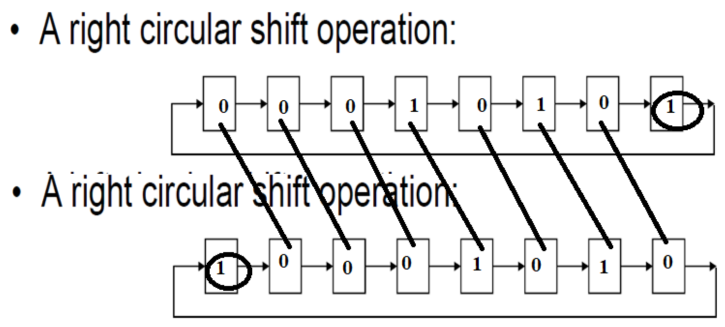
3. ***Left Arithmetic Shift (SHLA)*** moves each bit to the left by one. The vacant least significant bit (LSB) is filled with zero and the most significant bit (MSB) is discarded. It is identical to Left Logical Shift.

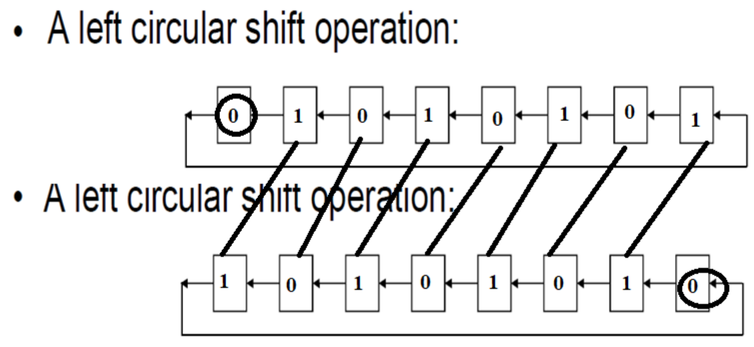
4. ***Right Arithmetic Shift (SHRA)*** moves each bit to the right by one. The least significant bit is discarded and the vacant MSB is filled with the value of MSB. but the sign bit itself remains unchanged.



5. Rotate Instructions

The rotate instructions produce a circular shift. Bits shifted out at one of the word are not lost as in a logical shift but are circulated back into the other end.


A right circular shift operation:
0 0 0 1 0 1 0 1
A right circular shift operation:
1 0 0 0 1 0 1 0


A left circular shift operation:
0 1 0 1 0 1 0 1
A left circular shift operation:
1 0 1 0 1 0 1 0