STEP TOWARDS SUCCESS

AKASH'S

Guru Gobind Singh Indra Prastha University Series

SOLVED PAPERS

[PREVIOUS YEARS SOLVED QUESTION PAPERS]

[B.Tech]
THIRD SEMESTER
Digital Logic and Computer Design (ECC-207)

Rs.81.00/-

AKASH BOOKS
NEW DELHI

SYLLABUS
DIGITAL LOGIC AND COMPUTER DESIGN
PAPER CODE : ECC-207

Instructions for paper setter:

1. There should be 9 questions in the term end examinations question paper.
2. The first (1st) question should be compulsory and cover the entire syllabus. This question should be objective, single line answers or short answer type question of total 15 marks.
3. Apart from question 1 which is compulsory, rest of the paper shall consist of 4 units as per the syllabus. Every unit shall have two questions covering the corresponding unit of the syllabus. However, the student shall be asked to attempt only one of the two questions in the unit. Individual questions may contain upto 5 sub-parts / sub-questions. Each Unit shall have a marks weightage of 15.

UNIT – I

Boolean Algebra and Combinational Logic: Review of number systems , signed, unsigned, fixed point, floating point numbers, Binary Codes, Boolean algebra – basic postulates, theorems, Simplification of Boolean function using Karnaugh map and Quine-McCluskey method – Implementations of combinational logic functions using gates, Adders, Subtractors, Magnitude comparator, encoder and decoders, multiplexers, code converters, parity generator/checker, implementation of combinational circuits using multiplexers.

UNIT – II

Sequential Circuits: General model of sequential circuits, Flip-flops, latches, level triggering, edge triggering, master slave configuration, concept of state diagram, state table, state reduction procedures, Design of synchronous sequential circuits, up/down and modulus counters, shift registers, Ring counter, Johnson counter, timing diagram, serial adder, sequence detector, Programmable Logic Array (PLA), Programmable Array Logic (PAL), Memory Unit, Random Access Memory

UNIT – III

Basic Computer organization: Stored Program, Organization, Computer registers, bus system, instruction set completeness, instruction cycle, Register Transfer Language, Arithmetic, Logic and Shift Micro-operations, Instruction Codes, Design of a simple computer, Design of Arithmetic Logic unit, shifter, Design of a simple hardwired control unit, Programming the basic computer, Machine language instructions, assembly language, Microprogrammed control, Horizontal and Vertical Microprogramming, Central Processing Unit, instruction sets and formats, addressing modes, data paths, RISC and CISC characteristics.

UNIT – IV

Computer Arithmetic, addition, subtraction, multiplication and division algorithms, Input-Output Organization, Modes of data transfer, Interrupt cycle, direct memory access, Input-Output processor, Memory Organization, Memory Hierarchy, Associative Memory, Cache Memory, Internal and external Memory. Virtual Memory.

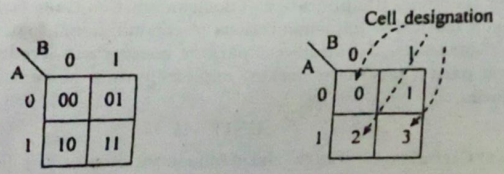
FROM ACADEMIC SESSION [2022-23]
THIRD SEMESTER [B.TECH]
DIGITAL LOGIC AND COMPUTER DESIGN (ECC-207)

UNIT - I

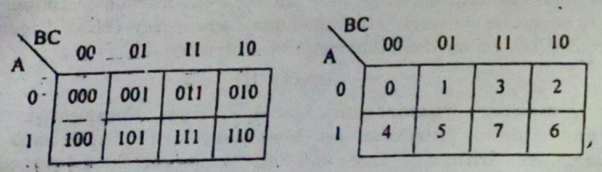
Q.1. Explain the simplification of switching functions using Karnaugh Map and Quine Mc-Clusky Methods. (2014)

Ans. Karnaugh Map Simplifications: The karnaugh map is a graphical representation that provides systematic method for simplifying and manipulating Boolean expressions. This technique is the most extensively used tool for simplification of Boolean functions. Fig. shows the 2 variable, 3 variable and 4 variable K maps.

For an n variable K-map 2^n cells are required. Each cells corresponds to one of the combination of minterms (or) maxterms of n variables. There are 2n possible combinations for n variables. For a 2 variable map, four (2^2) 4 cells are required. Similarly for 3 variables, (2^3) 8 cells are required to construct the 3 variable map. Gray code is used to identify the cells, the direct binary combinations of 2 variables is 00, 01, 10, 11. As a result of the coding, cells which have a common side correspond to the combinations that differ by the value of just a single variable. The direct binary is replaced by gray code (00, 01, 11, 10) to avoid error.



Three variable K- map



Four variable K- map

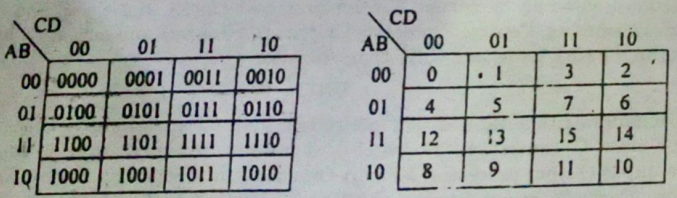


Fig. 2, 3 and 4 variable K-map and decimal equivalent.

Quine-MC Cluskey (Tabulation Method): K-maps are not suitable when the number of variables involved exceeds four. It may be used with difficulty up to five and

above variable systems. To avoid this difficulty, we use Quine-Mc Cluskey method. This method is suitable for problems where the number of variables exceeds four. The following steps are followed for simplification of Boolean function by using Tabulation method.

Step 1. List all minterms in equivalent binary form.

Step 2. Arrange the minterms based on the number of 1's.

Step 3. Compare each binary number from one group to other and if they differ by only one bit position put–(dash mark) and copy the remaining term, Place tick (3) mark after each comparison.

Step 4. Apply the same process described in step 3 for the resultant column and continue these cyoles until a single pass through cycle yields no further elimination ilterals.

Step 5. List the unticked prime implicant and form the prime implicant chart.

Each prime implicant is represented in a row and each minterm in a column. The cross mark (×) are placed in each row to show the comparison of minterms that make the prime implicant from this chart we can simply the expression.

(i) Search for single × column and select the prime implicant corresponding to that dot by putting the star mark in front of it.

(ii) Search for multiple dot columns one by one. If the corresponding minterm is already include in the final expression ignoring the minterms and go to next multi dot column otherwise include the corresponding prime implicant in the final expression.

Q.2. Explain De Morgan's Theorem. (2014)

Ans. DeMorgan's Theorem: DeMorgan's theorem are very important part of Booleam algebra. DeMorgan suggested two theorems.

Theorem 1. The complement of a product is equal to the sum of the complements. Table shows the truth table for Demorgan's theorem 1.

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

Table. Truth table for Demorgan's theorem 1.

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

This law can be extended to any number of variables or combinations of variables. For example

$$\overline{ABCD} + ... = \overline{A} + \overline{B} + \overline{C} + \overline{D} + ...$$

$$\overline{(AB)(CD)(EFG)...} = \overline{AB} + \overline{CD} + \overline{EFG} + ...$$

Theorem 2. The complement of a sum is equal to the product of the complements. Table shows the truth table for Demorgan's second theorem

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

Table: Truth table for Demorgan's second theorem

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

This law can be extended to any number of variables or combinations of variables. For example.

$$\overline{A+B+C+D+\ldots} = \bar{A}.\bar{B}.\bar{C}.\bar{D}\ldots$$

$$\overline{AB+CD+EFG+\ldots} = \overline{AB}.\overline{CD}.\overline{EFG}+\ldots$$

Q.3. Draw the logic diagram of 3-bit parity generator and respective truth table. **(2014)**

Ans. A parity bit of a '0' or a '1' is attached to the data bits such that total no. of 1's in the word is even parity and odd for odd parity. The parity bit can be attached to the code group either beginning or at the end depending on system design.

3-bit Even Parity Generator: In even parity the added bit (aprity bit) will make the number of 1's an even. Table shows the truth table of a 3 bit even parity generator.

Table. Truth Table for 3-bit even parity generator

3-bit message			Even parity bit
A	B	C	P
0	0	0	0
0	0	1	1
0	1	0	1

Q.4. Solve the following equation using Boolean algebra: **(2015)**

A'BC + ABC' + ABC' + ABC

Ans. Given that A'BC + AB'C + ABC' + ABC

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

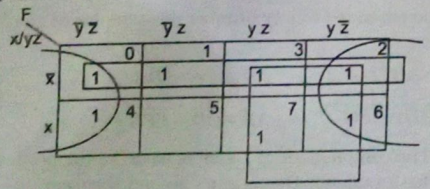
= BC + A (B ⊕ C)

Q.5. Simplify the given equation using K-map: **(2015)**

F = x' + xy + xz' + xy'z'

Ans. Given that F = x' + xy + xz' + xy'z'

It is 3-variable equation, hence



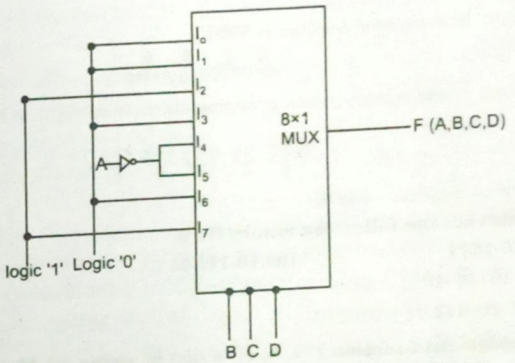
$F = \bar{x} + y + \bar{z}$

Q.6. (a) Implement using 8:1 multiplexer: **(2015)**

F(A,B,C,D) = Σ m (2,4,5,7,10,14)

Ans. Given that F (A, B, C, D) = Σm (2, 4, 5, 7, 10, 14) we have to implement using 8 × 1 MUX. Let A is used as an input and B,C,D are as selection lines, then imp'lementation table is

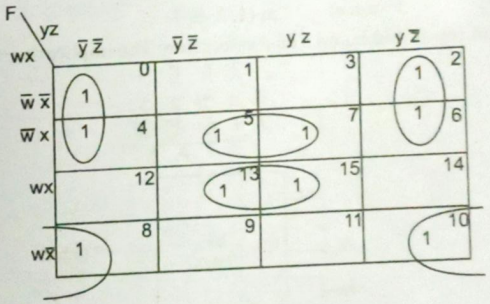
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	0	1	(2)	3	(4)	(5)	6	(7)
A	8	9	(10)	11	12	13	14	15
	0	0	1	0	$\bar{A}$	$\bar{A}$	0	1



Q.7. Find essential prime implicant in the given equation: **(2015)**

F (W,X,Y,Z) = Σm (0,2,4,5,6,7,8,10,13,15)

Ans. Given that F (W, X, Y, Z) = Σm (0, 2, 4, 5, 6, 7, 8, 10, 13, 15)

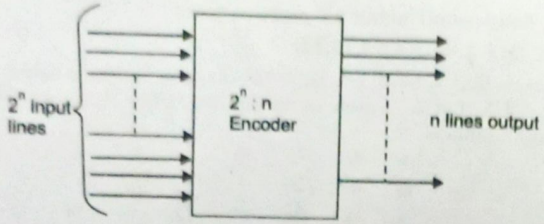


Essential prime implicants = 5

Q.8. Write short notes on **(2015)**

(i) Encoder

Ans. Encoder: An encoder is a combinational logic circuits. It is reverse of a decoder function. If has 2n input lines and 'n' output lines. An encoder accepts an active level on one of its inputs representing a digit such as a decimal/octal digit and converts it to a coded output.



Q.9 Convert the Hexadecimal number 68BE to Binary and then convert it to Octal.

Ans. Given hexadecimal number is 68BE

$$\underset{0110}{6}\ \underset{1000}{8}\ \underset{1011}{B}\ \underset{1110}{E}$$

To convert it into octal number, grouping starts from right to left in the form of 3 bit.

$$0\ \underset{6}{110}\ \underset{4}{100}\ \underset{2}{010}\ \underset{7}{111}\ \underset{6}{110}$$

Octal representation = 64276

Q.10 Subtract the following binary No's: **(2)**

(i) 10110-1011 **(ii) 1100.10-111.01**

Ans. (i) 10110–1011 = 1011

(ii) 1100. 10–111.01 = 0101.01

Q.11 Implement function F(a, b, c) = $ab + \bar{b}c$ using 4:1 Mux. **(2015)**

Ans. Given that $F(a,b,c,) = ab + \bar{b}c$

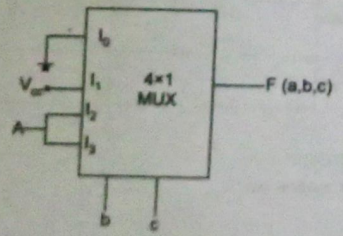
$$= ab(c + \bar{c}) + (a + \bar{a})\bar{b}c$$

$$= \underset{111}{abc} + \underset{110}{ab\bar{c}} + \underset{101}{a\bar{b}c} + \underset{001}{\bar{a}\bar{b}c}$$

$$F(a,b,c) = \Sigma m\,(1, 5, 6, 7)$$

Let a as an input and b and c as a select line. The implication table is shown below

	I_0	I_1	I_2	I_3
$\bar{A}$	o	①	2	3
A	4	⑤	⑥	⑦
	0	1	A	A



Q.12. With the help of equations explain 4 bit comparator. **(2015)**

Ans. A magnitude comparator is a combinational circuit that compares two numbers A and B, and determines their relative magnitudes. Consider 2 numbers. A and B with 4 digits each.

$$A = A_3A_2A_1A_0$$

$$B = B_3B_2B_1B_0$$

The 2 numbers are equal if all pairs of significant digits are equal i.e. $A_3 = B_3$, $A_2 = B_2$, $A_1 = B_1$ and $A_0 = B_0$.

The equality function of each pair of bits can be expressed logically with an equivalence function.

$$x_i = A_iB_i + A_i'B_i'$$

$$i = 0, 1, 2, 3$$

Where $x_i = 1$ only if the pair of bits in position i are equal i.e., if both are 1's or both are 0's.

For the equality condition to exist, all x_i variables must be equal to 1. This dictates an AND operation of all variables.

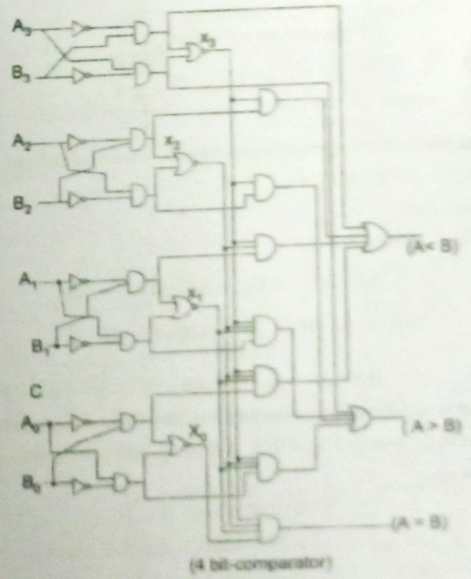
$$(A = B) = x_3x_2x_1x_0$$

The binary variable (A = B) is equal to 1 only if all pairs of digits of the two numbers are equal.

To determine if A is greater than or less than B, insect the relative magnitudes of pairs of significant digits starting from the most significant position. If the two digits are equal, compare the next lower significant pair of digits. This comparison continues until a pair of unequal digits is reached.

$$(A > B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0$$

$$(A < B) = A_3'B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1 + x_3x_2x_1A_0'B_0$$



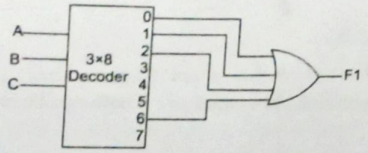
(4 bit-comparator)

Q.13. Implement the following multiple output combinational logic circuit using a 3-to-8 line Decoder? (2015)

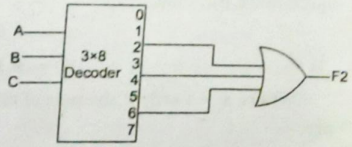
(i) $F_1 = \Sigma m\ (0,1,2,6)$ **(ii)** $F_2 = \Sigma m\ (2,4,6)$ **(iii)** $F_3 = \Sigma m(0,1,5,6)$

Ans. Given that

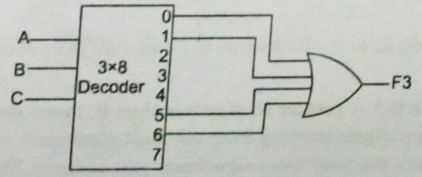
(i) $F_1 = \Sigma m\ (0, 1, 2, 6)$



(ii) $F_2 = \Sigma m(2, 4, 6)$



(iii) $F_3 = \Sigma m\ (0,1,5,6)$



Q.14. Subtract 748 from 983 using 9's complement method. (2016)

Ans. Equating the number of digits 983–748

9's complement of 748 = $(10^3-1) - 748 = 251$

Minuend = 983

9's complement of 748 = + 251

1234

Carry ← ↳ 1

235

Q.15. Simplify the expression using Boolean algebra: (2016)

$AB + (AC)' + AB'C(AB + C)$

Ans. Given that

$AB + (AC)' + AB'C(AB + C)$

$= AB + \overline{AC} + AB'C = AB + \bar{A} + \bar{C} + A\bar{B}C$

$= A(B + \bar{B}C) + \bar{A}\bar{C} = A(B + C) + \bar{A} + \bar{C}$ $[\because B + \bar{B}C = B + C]$

$= AB + AC + \bar{A} + \bar{C} = AB + AC + \overline{AC} = AB + 1 = 1$ $[\because A + \bar{A} = 1]$

Q.16. Give a Boolean expression for the following statement:

Y is a 1 only if A is a 1 and B is a 1 or if A is a 0 or B is a 0 (2016)

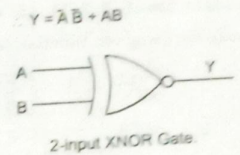
Ans. Given that

$Y = 1$ if $A = 1,\ B = 1$

$= 1$ if $A = 0,\ B = 0$

Truth Table

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1



Q.17. Convert 1011 to (i) Excess 3; (ii) Octal. (2016)

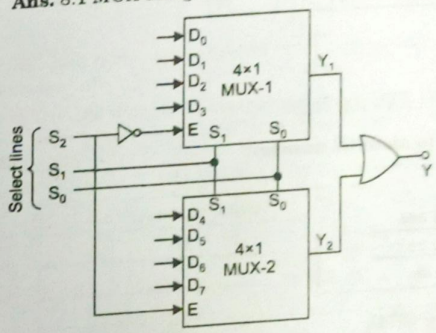
Ans. Given that 1011

(i) Excess 3 = 1110

(ii) Octal = 13

Q.18. Design an 8:1 multiplexer using 4:1 multiplexer and gates. (2016)

Ans. 8:1 MUX using 4:1 MUX



Truth table

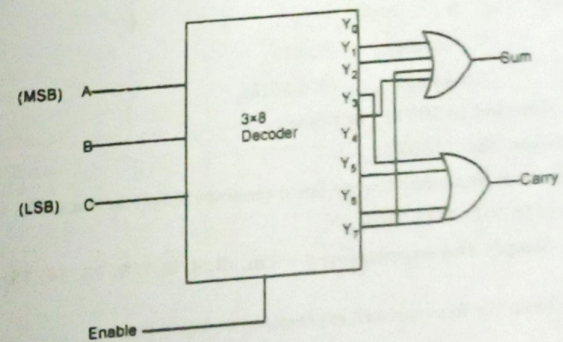
S_2	S_1	S_0	Output Y	
0	0	0	D_0	MUX-1 is enable
0	0	1	D_1	
0	1	0	D_2	
0	1	1	D_3	
1	0	0	D_4	MUX-2 is enable
1	0	1	D_5	
1	1	0	D_6	
1	1	1	D_7	

Q.19. Design a full adder using a 3:8 decoder and gates. (2015),(2016)

Ans. For Full adder, we have

Sum = $\Sigma m\ (1,2,4,7)$

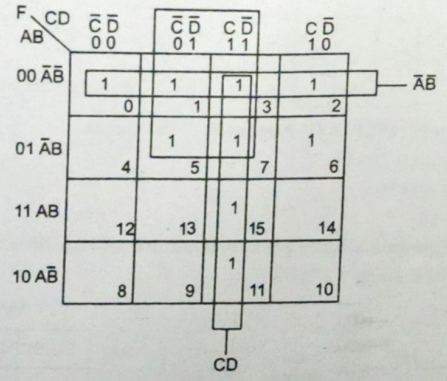
and Carry = $\Sigma m\ (3,5,6,7)$



Q.20. Simplify the Boolean function using K-map: **(2016)**

F(A, B, C, D) = Σm (1, 3, 7, 11, 15) + Σd (0, 2, 5)

Ans. K–map for the given function is



$$F = \bar{A}\bar{B} + \bar{A}D + CD$$

Q.21. Convert $(444.456)^{10}$ to an octal number. **(2016)**

Ans. Given that $(444.456)_{10}$

8 | 444
8 | 55 — 4
6 — 7

$$(444)_{10} = (674)_8$$

Real part		Fraction part		Real part
0.456×8	= 3	648	⇒	3
0.648×8	= 5	184	⇒	5
0.184×8	= 1	472	⇒	1
0.472×8	= 3	776	⇒	3

$$(0.456)_{10} = (0.3513)_8$$

$$\therefore (444.456)_{10} = (674.3513)_8$$

Q.22. Convert $(A6BF5)_{16}$ to binary. **(2016)**

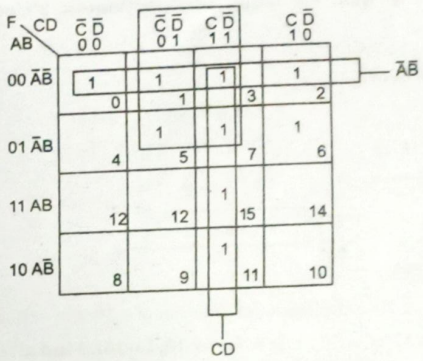
Ans. Given that $(A6BF5)_{16}$

To convert it into binary, write down separate binary no. for each

$(1010\ 0110\ 1011\ 1111\ 0101)_2$

Q.23. Simply the expression y = Σm. (3, 4, 5, 7, 9, 13, 14, 15) using k-map method. **(2016)**

Ans. Given the four variable expression, we know that



$$Y = \bar{A}B\bar{C} + A\bar{C}D + \bar{A}CD + ABC$$

Q.24. Describe the circuit and operation of single bit magnitude comparator. **(2016)**

Ans. One bit magnitude comparator is a combination logic circuit which compares the two inputs in the binary code and gives three outputs.

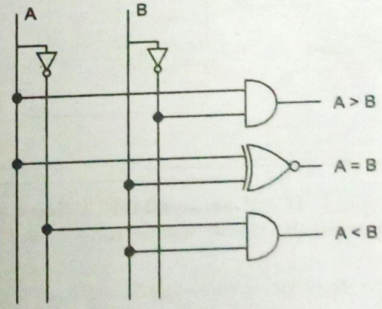
Truth Table of 1-bit comparator

Comparator Input		Comparator Outputs		
A	B	A>B	A=B	A<B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$$(A > B) = A\bar{B}$$

$$(A = B) = \bar{A}\bar{B} + AB = A \odot B$$

$$(A < B) = \bar{A}B$$



Q.25. Add - 10 with –10 using 8-bit arithmetic 2's complement method? (2017)

Ans. $(10)_{10} \rightarrow (00001010)_2$

2's complement of $(-10)_{10}$

$= (11110110)_2$

$(-10)_2 \longrightarrow$ 1111 0110

$(+)(-10)_2 \longrightarrow (+)$ 1111 0110

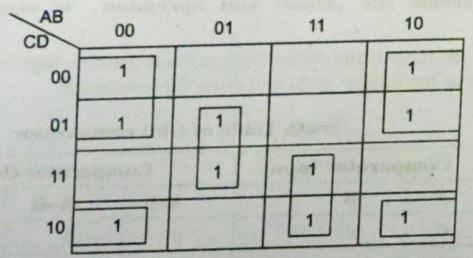
1 1110 1100

Carry be ignored

$(11101100)_2 \rightarrow$ 2's complement of addition of – 10 and –10 in 2's complement form.

Q.26. F(A, B, C, D) = Σ(0, 1, 2, 5, 7, 8, 9, 10, 14, 15). Find all the prime implicants, essential prime implicants and minimal SOP expression. (2017)

Ans. F(A, B, C, D) = Σ(0, 1, 2, 5, 7, 8, 9, 10, 14, 15)

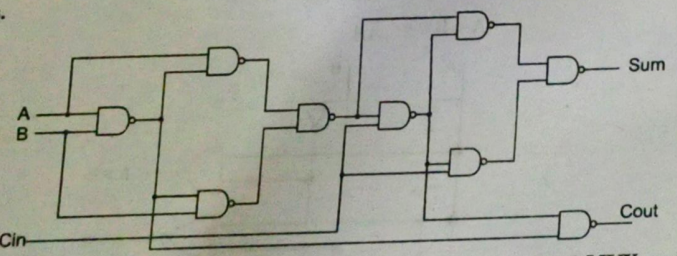


$F = \bar{B}\bar{C} + \bar{A}BD + ABC + B\bar{C}\bar{D}$ Minimal SOP expression.

There is no essential prime implicants in the given above output expression. All the minterms in the SOP expression is prime implicants.

Q.27. Realize full adder circuit using minimum number of two input NAND gates only. (2017)

Ans.

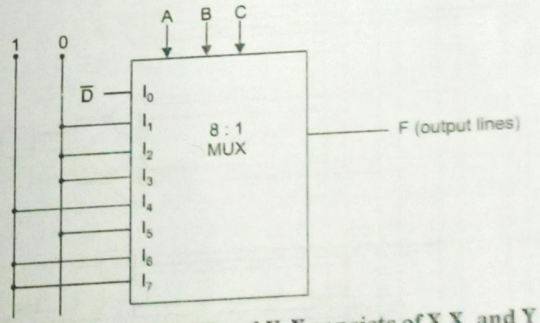


Q.28. Signals A,B, C, D and are available. Using a single 8 to 1 MUX and no other gates. Implement the Boolean function f(A, B, C, D) = $BC + AB\bar{D} + \bar{A}C\bar{D}$. (2017)

Ans. $f(A, B, C, D) = BC + AB\bar{D} + \bar{A}C\bar{D}$

	A	B	C	D	Outputs
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	1

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{D}$	(0)	0	0	0	(4)	0	(6)	(7)
D	8	8	8	8	(12)	8	(14)	(15)
	$\bar{D}$	0	0	0	1	0	1	1



Q.29. Consider two bit number X and Y. X consists of X_0X_1 and Y consists of Y_0Y_1. Design a circuit with three outputs Z_0, Z_1 and Z_2 such that: (2017)

(i) $Z_0 = 1$ when X < Y (ii) $Z_1 = 1$ when X = Y (iii) $Z_2 = 1$ when X > Y

Ans. Truth table for 2 - Bit Magnitude comparator.

Inputs	Outputs		
$X_1\ X_0\ Y_1\ Y_0$	$X_1X_0 = Y_1Y_0(Z_1)$	$X_1X_0 < Y_1Y_0(Z_0)$	$X_1X_0 > Y_1Y_0(Z_2)$
0 0 0 0	1	0	0
0 0 0 1	0	1	0
0 0 1 0	0	1	0
0 0 1 1	0	1	0
0 1 0 0	0	0	1
0 1 0 1	1	0	0
0 1 1 0	0	1	0
0 1 1 1	0	1	0
1 0 0 0	0	0	1
1 0 0 1	0	0	1
1 0 1 0	1	0	0
1 0 1 1	0	1	0
1 1 0 0	0	0	1
1 1 0 1	0	0	1
1 1 1 0	0	0	1
1 1 1 1	1	0	0

Expression for (Z_1)

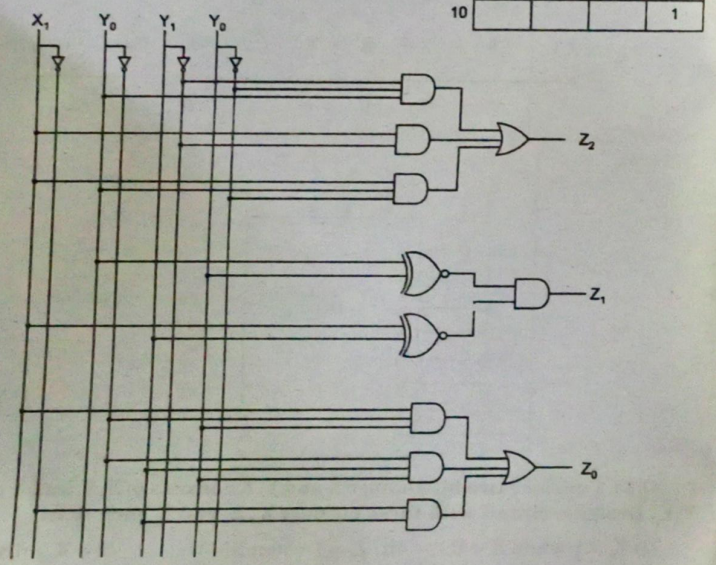
$$Z_1 = \bar{X}_1\bar{X}_0\bar{Y}_1\bar{Y}_0 + \bar{X}_1X_0\bar{Y}_1Y_0 + X_1X_0Y_1Y_0 + X_1\bar{X}_0Y_1\bar{Y}_0$$

$$= (X_1 \oplus Y_1)(X_0 \oplus Y_0)$$

Similarly, $Z_0 = \bar{X}_1\bar{X}_0Y_0 + \bar{X}_1Y_1 + \bar{X}_0Y_1Y_0$

$$Z_2 = X_0\bar{Y}_1Y_0 + X_1X_0\bar{Y}_0 + X_1\bar{Y}_1$$

X_1X_0 \ Y_1Y_0	00	01	11	10
00	1			
01		1		
11			1	
10				1



Q.30. Why is the ASCII code a 7 bit code? (2017)

Ans. The American standard code for Information Inter change (ASCII) pronounced as 'ASKEE' is a widely used alpha numeric code. This is basically a 7–bit code. Since the number of different bit patterns that can be created with 7–bits is $2^7 = 128$, the ASCII can be used to encode both the lower case and upper case characters of the alphabet (52 symbols) and some special symbols as well, in addition to the 10 decimal digits. It is used extensively for printers and terminals that interface with small computer systems.

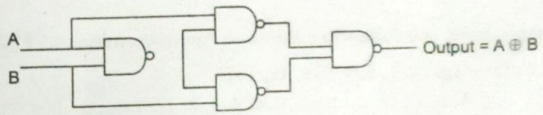
Q.31. What are the two basic form of the Boolean expression name them? (2017)

Ans. Two basic form of the Boolean expression are:-

(i) Sum Of Product (SOP) (ii) Product Of Sum (POS)

Q.32. Realize EX-OR gate using minimum number of NAND gate only. (2017)

Ans.



Q.33. Simplify the given Boolean expression and implement it with NOR gate circuit only F = $A\bar{B} + ABD + AB\bar{D} + \bar{A}\bar{C}\bar{D} + \bar{A}B\bar{C}$ (2017)

Ans. $F = A\bar{B} + ABD + AB\bar{D} + \bar{A}\bar{C}\bar{D} + \bar{A}B\bar{C}$

CD \ AB	00	01	11	10
00	1	1	1	1
01		1	1	1
11			1	1
10			1	1

$$F = A + B\bar{C} + \bar{A}\bar{C}\bar{D}$$

Q.33. What are the essential prime implicants? (2017)

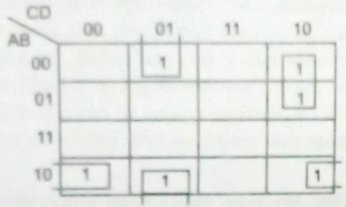
Ans. Each square or rectangle made up of the bunch of adjacent minterms is called a subcube. Each of these subcubes is called a prime implicants (PI). The PI which contains at atleast one 1 which cannot be covered by any other PI is called an essential prime implicant (EPI).

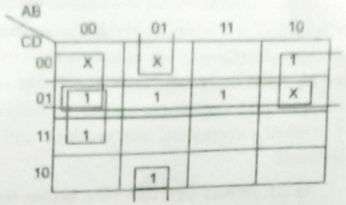
Q.34. Minimize the following function using K-map (2017)

(i) $F_1 = \Sigma m\ (1, 2, 6, 8, 9, 10)$

(ii) $F_2 = \Sigma m\ (2, 4, 5, 7, 9, 12) + d(0, 1, 6)$

Ans. (i) $F_1 = \Sigma m\,(1, 2, 6, 8, 9, 10)$ (ii) $F_2 = \Sigma m(2, 4, 5, 7, 9, 12) + d(0, 1, 6)$





$F_1 = \bar{A}C\bar{D} + \bar{B}\bar{C}D + A\bar{B}\bar{D}$ $F_2 = \bar{A}B + B\bar{C}\bar{D} + \bar{A}\bar{D} + \bar{B}\bar{C}\bar{D}$

Q.35. Obtain the decimal equivalent of the given hexadecimal number $(3A.2F)_{16}$. (2017)

Ans. $(3A.2F)_{16} = 3 \times 16^1 + 10 \times 16^0 + 2 \times 16^{-1} + 15 \times 16^{-2}$

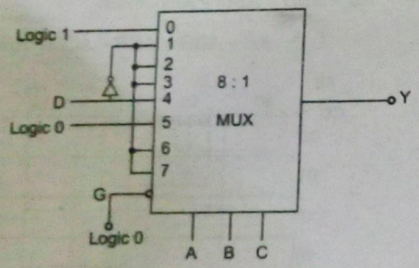
$= 48 + 10 + \frac{2}{16} + \frac{15}{256} = 58 + 0.125 + 0.058 = (58.183)_{10}$

Q.36. Implement the following Boolean function using 8:1 MUX. (2017)

F(A, B, C, D) = Σm (0, 1, 2, 4, 6, 9, 12, 14).

Ans. F(A, B, C, D) = Σm(0, 1, 2, 4, 6, 9, 12, 14)

Logic Table



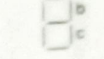
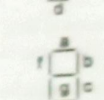
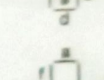
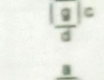
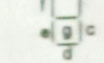
Logic Diagram

Q.37. Explain in brief the working of BCD to seven segment decoder. (2017)

Ans. Seven segment displays are used to give a visual indentification of the output states of digital ICs such as counters, registers etc. These outputs are usually in four bit BCD Binary Coded Decimal.

Form and thus not suitable for driving seven segment displays. A special decoder is designed to drive from BCD code to 7-segment display, it is called BCD to 7-segment decoder. This shows the segments activated during each digit display.

Table

Digit	Segment Activated	Display
0	a, b, c, d, e, f	
1	b, c	
2	a, b, d, e, g	
3	a, b, c, d, g	
4	b, c, f, g	
5	a, c, d, f, g	
6	a, c, d, e, f, g	
7	a, b, c	
8	a, b, c, d, e, f, g	
9	a, b, c, d, f, g	

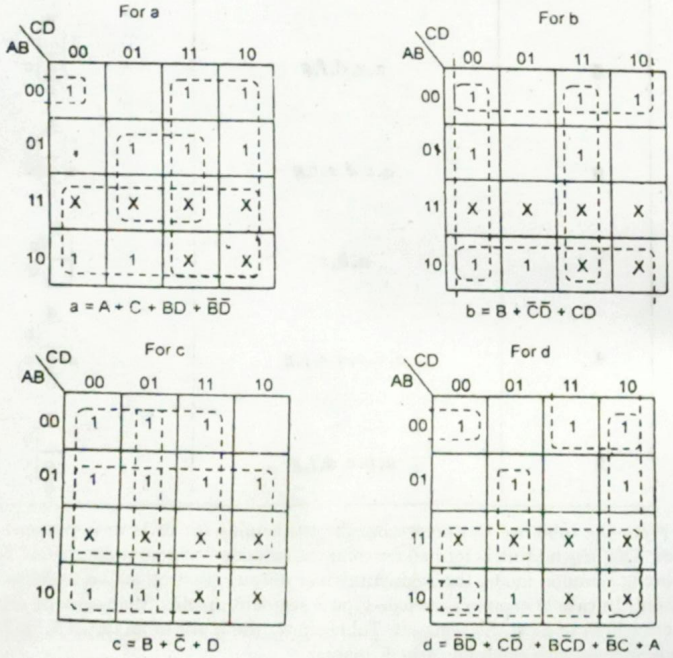
From the table we can determine the truth table for BCD to 7 segment decoder/driver. This truth table is formed for common cathode 7-segment display. If 7-segment display is common mode, the segment driver output must be active LOW to glow the segment. In case of common cathode type 7-segment display, the segment driver must be active high to glow the segment. Table shows the truth table for BCD to 7-segment decoder/driver with common cathode display.

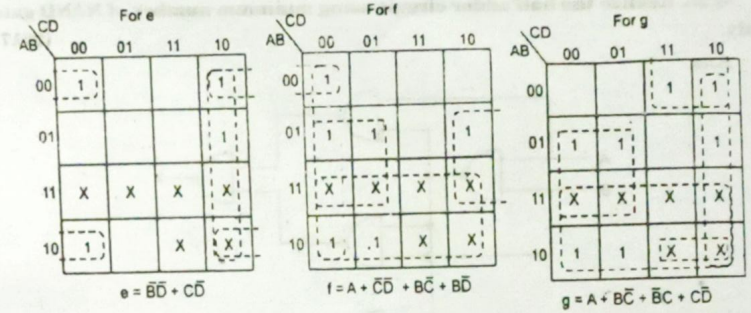
Table truth table common cathode 7-segment display.

Digit	BCD input				7-Segment						
	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

The unused BCD codes are 1010, 1011, 1100, 1101, 1110, 1111 so place X Don't care condition for these corresponding cells.

K-map Simplification





Logic diagram: Fig. shows the logic diagram of BCD to 7-segment display decoder/ driver.

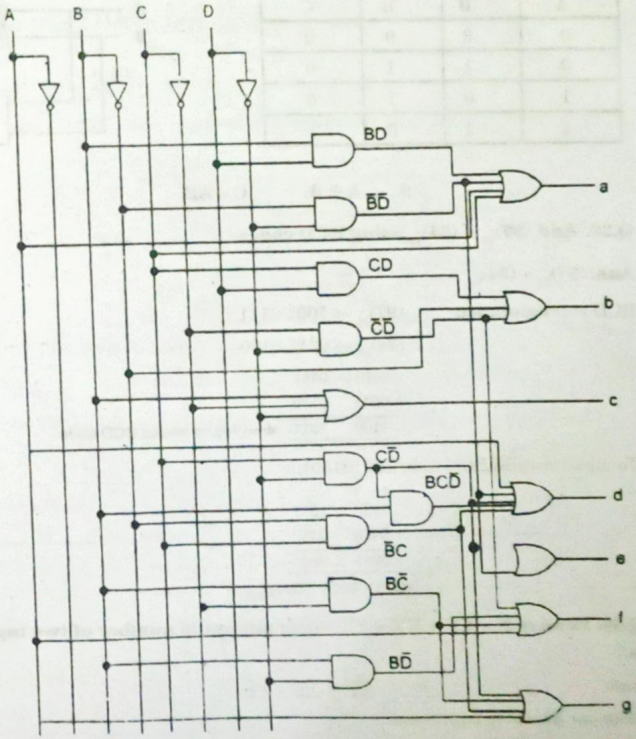
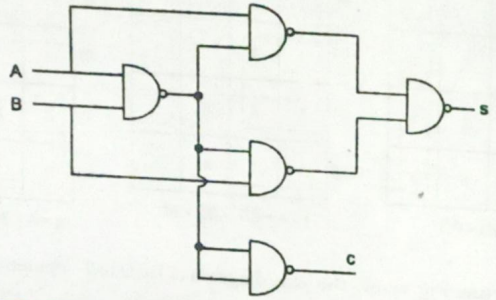


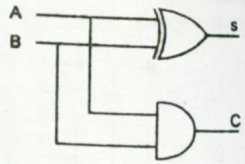
Fig. Logic diagram for BCD to 7-Segment, decoder display

Q.38. Realize the half adder circuit using minimum number of NAND gate only.

Ans. **(2017)**



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



$S = A \oplus B, \qquad C = AB$

Q.39. Add $(97)_{10} + (34)_{10}$ using BCD codes. **(2018)**

Ans. $(97)_{10} + (34)_{10}$

BCD representation $(97)_{10} \rightarrow 1001\ 0111$

$(34)_{10} \rightarrow 0011\ 0100$

```
  1001   0111
+ 0011   0110
-------------
  1100   1011   <-- No. is invalid BCD code
```

To make it valid BCD code add $(0110)_2$

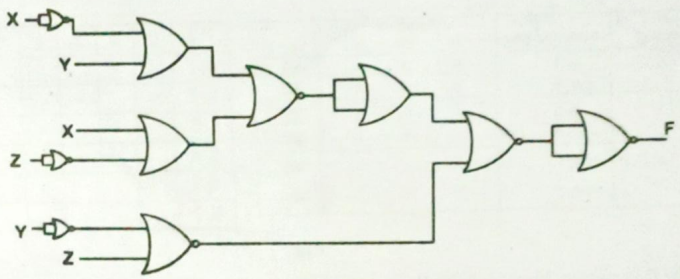
```
    1  1
  1100   1011
+ 0110   0110
-------------
 10011   0001
```

$(0001\ 0011\ 0001)_{BCD}$

Q.40. Realize F = XY′ + X′Z + YZ′ using minimum number of two input NOR gates. **(2018)**

Ans. $F = X\bar{Y} + \bar{X}Z + Y\bar{Z}$

Take bar on above expression.



$$= \overline{X\bar{Y} + \bar{X}Z + Y\bar{Z}}$$

$$= \overline{X\bar{Y}} + \overline{\bar{X}Z} + \overline{Y\bar{Z}}$$

$$= (\bar{X} + Y).(X + \bar{Z}).(\bar{Y} + Z)$$

Q.41. The minimal sum of product is given by **(2018)**

F = AC′ + A′D + BD

Find canonical S.O.P and canonical P.O.S

Ans. F = AC′ + A′D + BD, Minimal sum of product form

CD \ AB	00	01	11	10
00			1	1
01	1	1	1	1
11	1	1	1	1
10				

Canonical SOP form = $A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D + AB\bar{C}D +$
$A\bar{B}\bar{C}D + \bar{A}\bar{B}CD + \bar{A}BCD + ABCD + A\bar{B}CD$

Canonical POS form = $(A + B + C + D).(A + \bar{B} + C + D).(A + B + \bar{C} + D).$
$(A + \bar{B} + \bar{C} + D).(\bar{A} + \bar{B} + \bar{C} + D).(\bar{A} + B + \bar{C} + D)$

Q.42. Simplify F(A, B, C) = Σ m (0, 1, 2, 5, 6, 7) using Q-M method. **(2018)**

Ans. F(A, B, C) = Σ m (0, 1, 2, 5, 6, 7) Using Q.M. method

Group	Minterm		Variables A	B	C
0	0	✓	0	0	0
1	1	✓	0	0	1
	2	✓	0	1	0
2	5	✓	1	0	1
	6	✓	1	1	0
3	7	✓	1	1	1

Group	Matched pair	Variables A	B	C	
0	0, 1	0	0	–	$\bar{A}\bar{B}$
	0, 2	0	–	1	$\bar{A}\bar{C}$
1	1, 5	–	0	1	$\bar{B}C$
	2, 6	–	1	0	$B\bar{C}$
2	5, 7	1	–	1	AC
	6, 7	1	1	–	AB

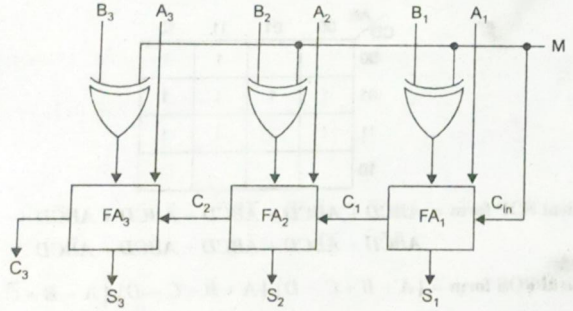
P.I	0	1	2	5	6	7
0,1	×	×				
0,2	×		×			
1,5		×		×		
2,6			×		×	
5,7				×		×
6,7					×	×

Fig.

In this particular question, there is no EPI, so all are PI.

Q.43. Draw and explain the circuit diagram of 3-bit 1's compliment adder-subtractor. **(2018)**

Ans. Here the addition and subtraction operations are combined into one circuit with one common binary adder. This is done by including an X-OR gate with each full-adder. The mode input M controls the operation. When M = 0, the circuit is an adder and when M = 1, the circuit becomes a substractor. Each X-OR gate receives input M and one of the inputs of B. when M = 0, we have $B \oplus 0 = B$. The full adder receives the value of B, the input carry is O and the circuit performs A + B. when M = 1, we have $B \oplus 1 =$ and $C_1 = 1$. The B inputs are complemented and a 1 is added through the input carry. The circuit performs the operation A plus the 2's complement of B.



Logic diagram of a 3-bit Binary Adder/Subtractor

Q.44. Implement three input XOR gate using 2:1 MUX and basic gate. **(2018)**

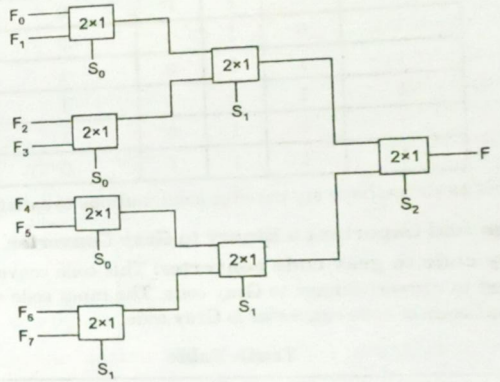
Ans. Truth Table for 3-input XOR gate

$$F = A \oplus B \oplus C$$

Inputs			Output	
A	B	C	F	
0	0	0	0	F_0
0	0	1	1	F_1
0	1	0	1	F_2
0	1	1	0	F_3

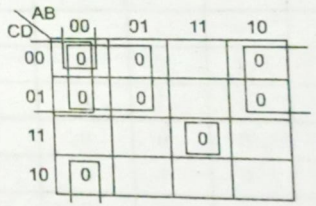
1	0	0	F_4
1	0	1	F_5
1	1	0	F_6
1	1	1	F_7

Lets treat these inputs as a select lines means A represent S_2, B represent S_1 and C represent S_0



Q.45. Using K-map simplify expression, Y = f(A, B, C, D) = π(0, 1, 2, 4, 5, 8, 9, 15) **(2018)**

Ans. $Y = f(A, B, C, D) = \pi(0, 1, 2, 4, 5, 8, 9, 15)$



$$Y = (A + C).(B + C).(A + B + D).(\bar{A} + \bar{B} + \bar{C} + \bar{D})$$

Q.46. How many combinations are there for output logic 1 of 2-bit comparator, if output Y is logic 1 whenever 2-bit input A is greater than 2-bit input B?

Ans. **(2018)**

Inputs				Output
A_1	A_0	B_1	B_0	A > B
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0

0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

According this table we have six possible combinations to satisfy the condition.

Q.47. Design and important a Binary to Gray Converter. **(2015),(2018)**

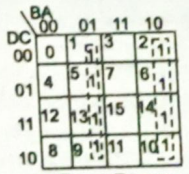
Ans. Binary code to gray code converter: This code converter combinational circuit is designed to convert binary to Gray code. The input code of code converter is binary. The output code of code converter is Gray code.

Truth Table

Binary Code				Gray Code			
D	C	B	A	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

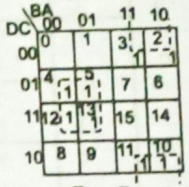
K-Map Simplification

Expression For G_0



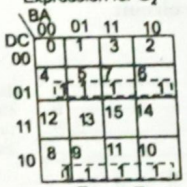
$G_0 = \bar{B}A + B\bar{A} = B \oplus A$

Expression for G_1



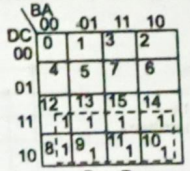
$G_1 = \bar{C}B + C\bar{B} = C \oplus B$

Expression for G_2



$G_2 = \bar{D}C + D\bar{C} = C \oplus D$

Expression for G_3



$G_3 = D$

We get the simplified boolean expression for the code converter of Binary of Binary to Gray code

$$G_0 = B\bar{A} + \bar{B}A = B \oplus A$$
$$G_1 = C\bar{B} + \bar{C}B = C \oplus B$$
$$G_2 = D\bar{C} + \bar{D}C = C \oplus D$$
$$G_3 = D$$

By using the above expression we can construct the binary to gray code converter as follows (see Fig.).

Logic Diagram

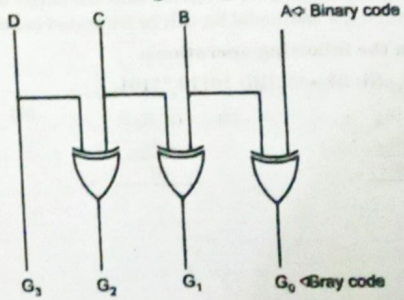
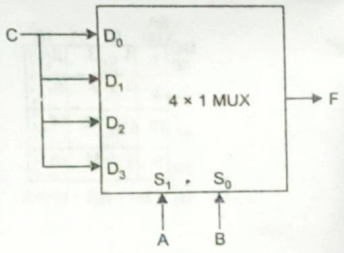


Fig. Logic circuit for binary to gray code converter

Q.48. Implement the logic expression using 4:1 MUX, F = Σm (1, 3, 5, 7) **(2018)**

Ans. $F = \Sigma m\,(1, 3, 5, 7)$

S_1	S_0		
A	B	C	F
0	0	0	0
0	0	1	1 F = C
0	1	0	0
0	1	1	1 F = C
1	0	0	0
1	0	1	1 F = C
1	1	0	0
1	1	1	1 F = C



Q.49. Design an even parity bit generator circuit. (2018)

Ans.

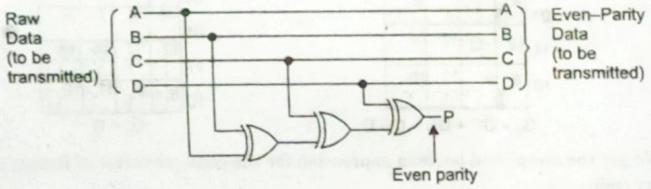


Fig. Even Parity generator

Error in the data can be detected using XOR gates. Binary data may be corrupted during transmission and processing.

This data corruption may be due susceptive to noise that can alter some of the '1' in to 'zeros' or 'zeros' or in to '1' s. It is important to detect such errors. Detection of such errors can be done by using additional bit called 'Parity bit' that is to be added before transmitting the original data. At the receiving side, the parity bit will be used to check for any errors and then the additional bit will be truncated before processing the data.

Q.50. Perform the following operations: (2018)

(i) $BC5_{16}$–$A3B_{16}$ (ii) 23_8+45_8 (iii) 10110_2*1101_2

Ans. (i) $(B\ C\ 5)_{16}$ − $(A\ 3\ B)_{16}$ = $1\ 8\ A$

(ii) $(2\ 3)_8$ + $(4\ 5)_8$ = 70

(iii)
$$\begin{array}{r} 10110 \\ \times\ 1101 \\ \hline 10110 \\ 00000\times \\ 10110\times\times \\ 10110\times\times\times \\ \hline 100011110 \end{array}$$

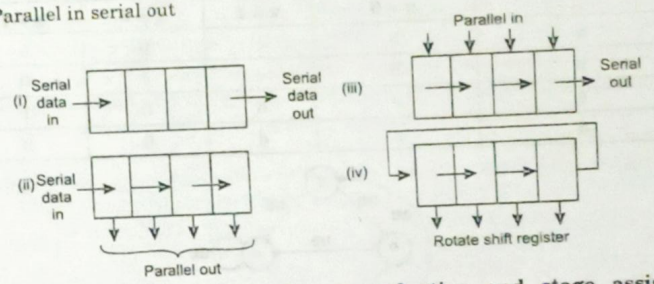
UNIT–II

Q.1. Explain various types of shift registers? (2014)

Ans. Shift Register: A register is a set of FFs used to store binary data. In shift register, FFs are connected together such that data may be shifted into and shifted out of them is called a shift register. This shifting may be in serial form or parallel form.

There are four basic types of shift register.

1. Serial – in serial out
2. Serial – in parallel out
3. Parallel in serial out



Q.2. Explain the process of state reduction and stage assignment synchronous sequential circuits. (2014)

Ans. State Reduction

:Two states are said to be redundant or equivalent, if every possible set of inputs generate exactly the same outputs and the same next states. When two states are equivalent one of them can be removed without altering input output relationship. Let us consider the state diagram as shown in Fig. The states are denoted by letter symbols instead of their binary values because in state reduction technique internal states are important, but input output sequences are important. The procedure contains two steps.

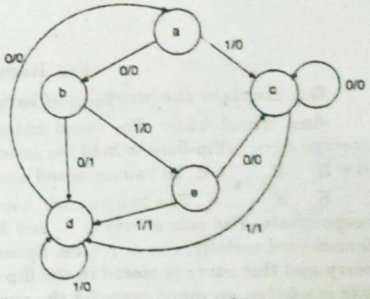


Fig. Example state diagram

Step 1: Finding the state table for the given state diagram.

First the given state diagram is converted into a state table. Fig. shows the example of state diagram.

Present state	*Next state*		*Output*	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	*b*	*c*	0	0
b	*d*	*e*	1	0
c	*c*	*d*	0	1
d	*a*	*d*	0	0
e	*c*	*d*	0	1

Both are equivalent states because of state *c* and *e* having same next state and same output

Step 2: Finding equivalent states. The two present states go to the same next state and have the same output for both the input combinations. We can easily find this from the state table, states c and e are equivalent. This is because both c and e states go to states c and d outputs of 0 and 1 for x = 0, x = 1 respectively. Therefore, the state e can be removed and replaced by c. The final reduced table and state diagram are given in the table and Fig. The second row have e state for the input x = 1, it is replaced by c because the states c and e are equivalent.

Table: Reduced State Table

Present state	Next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	b	c	0	0
b	d	c	1	0
c	c	d	0	1
d	a	d	0	0

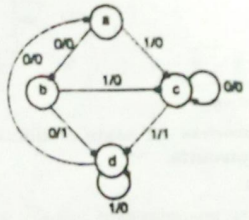
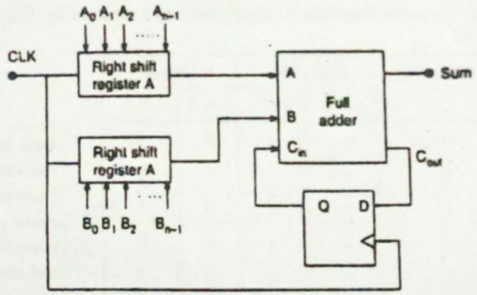


Fig. Reduced state diagram.

Q.3. Explain the working of serial adder. Draw the block diagram? (2014)

Ans. Serial Adder: The serial adder method uses only one full adder circuit and a storage device (flip-flop) to hold the generated output carry. Let $A = A_{n-1} A_{n-2} \ldots A_0$ and $B = B_{n-1} B_{n-2} \ldots B_0$ be two unsigned numbers that have to be added to produce sum $S_{n-1} S_{n-2} \ldots S_0$. The two number A and B are stored in the register A and register B respectively. The pair of bits in A_0 and B_0 (which are stored in right shift register) are transferred serially, one at a time, through the single full adder to produce a sum and carry and this carry is stored in the flip flop. The stored output carry from one pair of bits is used as an input carry for the next pair of bits as shown in Fig. The addition of two n-bit numbers starts from the least significant bits and progresses step by step to the most significant bits in different time sequences. Initially the D flip-flop is cleared.



Q.4. Explain various operations performed in RAM and ROM. Mention the data and control signals in RAM and ROM? (2014)

Ans. The basic memory operation follows the given functions.

(i) Enable the memory to be written or read the content in a memory. Select the address in memory that is being accessed for a read or write operation.

(ii) Select the control signals either read or write operation to be performed.

(iii) Supply the content to be stored in memory during a write operation.

(iv) Hold the data coming from the memory during a read operation.

Q.5. Explain the features of edge-triggered flip-flop. Draw the logic diagram of D-type positive edge-triggered flip-flop? (2015)

Ans. The inputs of flip flops may change even during the presence of a clock pulse due to certain operations in the system. This causes uncertainty in the outputs of the flip flop, which is eliminated by using edge triggered flip flops. The problem of race around condition can be sorted by 'edge triggering' the flip flop. The term edge triggering means that the flip flop changes state either at the positive edge (rising edge) or at the negative edge (falling edge) of the clock pulse and is sensitive to its inputs only at this transition of the clock. Fig.1 shows the positive and negative transition of the clock.

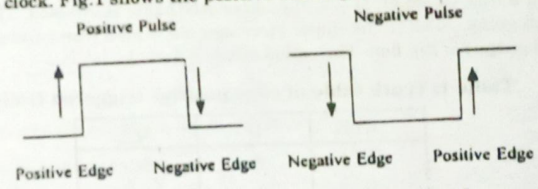


Fig. (1) Positive and Negative Edge of pulse

Edge Triggered D Flip Flop: This flip flop can be set or reset during the transition period of clock from low to high to low. Usually a capacitor in series with a resistor is used for this purpose. The positive or negative voltage across the resistor appears only during the positive transition or negative transition of the clock. Figure 2 shows the RC equivalent circuit (called as edge detector circuit) and its output waveform.

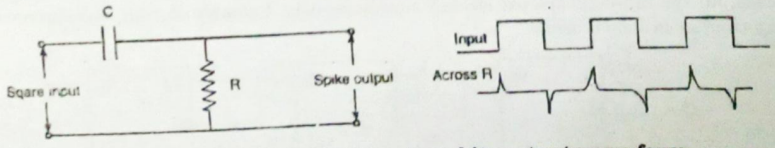


Fig. (2) RC equivalent Circuit and its output wave form.

The shorter pulse (spike) width at the clock input of a triggered edge flip flop increases the chance of the output being synchronized to the flip flop operation. This is because a spike gives less chance for the SET and RESET inputs to change during the time the flip flop is enabled.

Figure (4) shows a logical symbol of positive edge and negative edge triggered flip flops and fig. (3) shows the edge triggered D flip. Table (1) shows the truth table of edge triggered D flip flop.

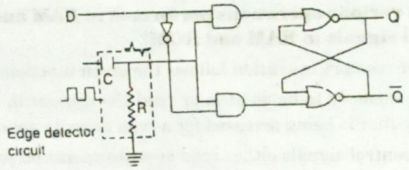


Fig. (3) Edge triggered D flip flop.

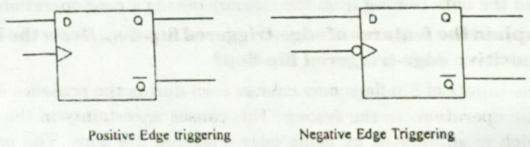


Fig. (4) Logic symbol of positive and negative edge triggered of D flip flop.

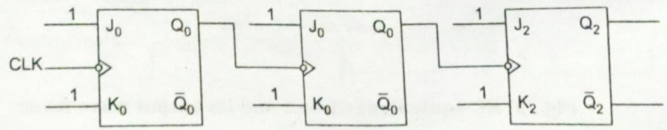
When D = 0, and the edge detector senses a positive edge at the CLK input, the output of the lower AND gate steers a low going pulse to the RESET input of flip flop, thus storing a 0 at Q. When D = 1, the upper AND gate is enabled. The edge detector sends a high going pulse to the upper steering gate, which transmits a low going SET pulse to the output of flip flop. The action stores a 1 at Q.

Table 1: Truth table of edge positive triggered D flip flop

CLK	D	Q_{n+1}
↑	0	0
↑	1	1

Q.6. Compare the design features of synchronous and Asynchronous Counters. Give an illustration for each. (2014)

Ans. Asynchronous Counter: To design an asynchronous counter, the number of flip-flops required depends on the number of states. The maximum number of state of a counter is 2^n, where n is the number of flip-flops in the counter. If we have two flip-flops, the maximum possible number of output states of the counter is z^2 i.e., 4. In this case, all the flip-flops are not clocked simultaneously. Example of 3-bit asynchronous up counter is shown below.



Synchronous Counter Design:

(i) Find the number of flip-flops required.

(ii) Write the count sequence in the tabular form.

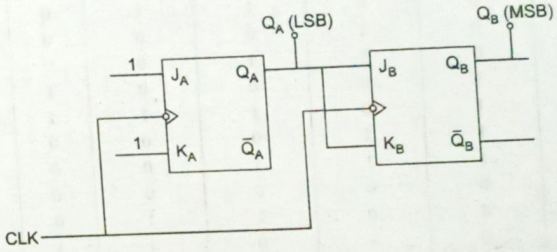
(iii) Determine the flip-flop inputs which must be present for the desired next state from the present state using the excitation table of the flip-flops.

(iv) Prepare k-map for each flip-flop input in terms of flip-flop outputs as the input variables.

(v) Simplify the k-maps and obtain the minimized expressions.

(vi) Connect the circuit using flip-flops and other gates corresponding to the minimized expressions.

A2-bit synchronous counter is shown below.



Q.7. A sequential circuit with two D flip-flops A to and B, two inputs x and y, one output z is specified by the following next-state and output equations:

$$A(t+1) = x'y + xA$$

$$B(t+1) = x'B + xA$$

$$z = B$$

(i) Draw the logic diagram of the circuit. (2)

(ii) Derive the state table. (2)

(iii) Derive the state diagram. (2.5)

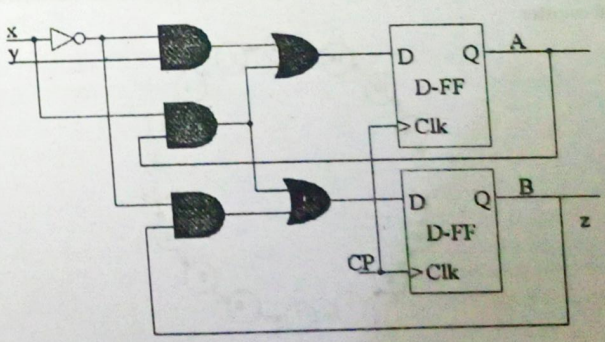
The next state and output equation are as follows: (2014)

$$A(t+1) = x'y + xA$$

$$B(t+1) = x'B + xA$$

$$z = B$$

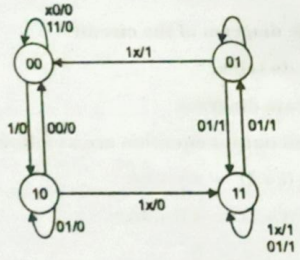
Ans. (i) The logic diagram is as follows:



(ii) The state table is as follows:

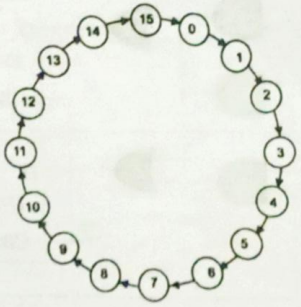
Present		Inputs		Next State		Output
A	B	x	y	A	B	z
0	0	0	0	0	0	0
0	0	0	1	1	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	1	1
0	1	0	1	1	1	1
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	0	0	0
1	0	0	1	1	0	0
1	0	1	0	1	1	0
1	0	1	1	1	1	0
1	1	0	0	0	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

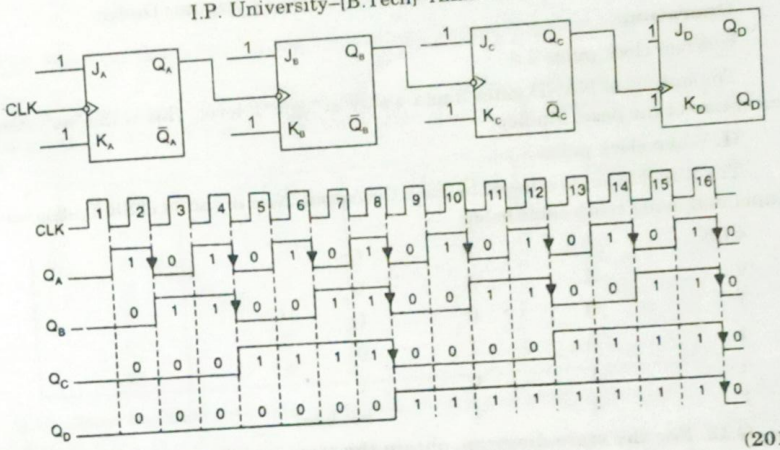
The state diagram is as follows:



Q.8. Design a 4-bit asynchronous counter. **(2015)**

Ans. 4-bit asynchronous counter, counts 0000 to 1111, i.e. 16 states; it is also called MOD-16 counter.



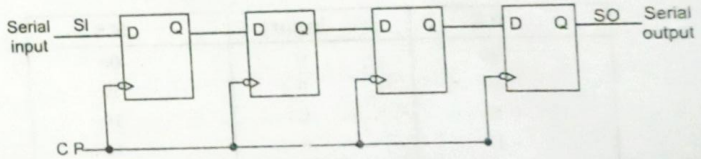


(2015)

Q.9. Shift Registers?

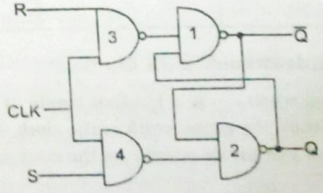
Ans. Shift Register: A register capable of shifting its binary information either to the right or to the left is called a shift register. The logical configuration of a shift register consists of a chain of flip-flops connected in cascade, with the output of the flip flop connected to the input of the next flip. Flop-All flip flops receive a common clock pulse that causes the shift from one stage to the next.

The simplest possible shift register in one that uses only flip-flops, as shown in Fig. The Q_i output of the given flip-flop is connected to the D input of the flip-flop at its right. Each clock pulse shifts the contents of the register one bit position to the right. The serial input determines what goes in to the left most flip-flop during the shift. The serial output is taken from the output of the right most flip-flop prior to the application of a pulse. Although this register shifts its contents to the right, if we turn the page up- side down. We find that the register shifts its contents to the left. Thus, a unidirectional shift register can function either as a shift-right or as shift-left register.



Q.10. Explain SR Flip flop using diagram and truth table. **(2015)**

Ans. SR flip-flop using NAND gate



Operation:

I. When clock pulse = 0

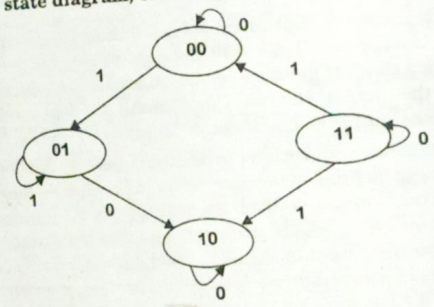
The output of NAND gates 3 and 4 stay at logic '1' level. This is the "no" change" condition of the basic flip-flop.

II. When clock pulse = 1

The S or R input is allowed to reach the output. Now the effect of SR flip-flop can be explained using truth-table below

CLK	S	R	Q_{n+1}
0	X	X	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	not used

Q.12. For the state diagram, obtain the state table? (2015)



Ans. For the given state diagram, the state table is given below

P.S	Input	N.S.
00	0	00
01	0	10
10	0	10
11	0	11
00	1	01
01	1	01
10	1	11
11	1	00

Q.13. What is the drawback of JK flip-flop?

Ans. In JK flip-flop, when J = K = 1 output toggle. If $t_p < \Delta t$ i.e. propagation delay time of flip-flop is less than the pulse width of the clock, the output will oscillate back and forth between 0 and 1 in the duration t_p of the clock pulse width. This is known as "race-around condition".

Q.14. What do you mean by the term State Table? Explain with Example. (2015)

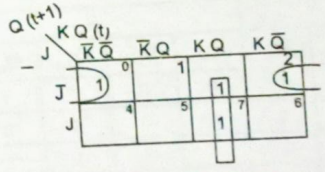
Ans. State table gives complete description about present state, next state, input and output of a sequential circuit. The state table of sequence detector 011 is shown below.

P.S	N.S	Input	Output
A	A	1	0
A	B	0	0
B	B	0	0
B	C	1	0
C	C	1	1
C	A	0	0

Q.15. Show the characteristic equation for the complement output of a JK flips-flop is $\bar{Q}(t+1) = \bar{J}\,\bar{Q} + \bar{K}Q$. (2015)

Ans. The characteristic table of Jk flip-flop is

J	K	Q(t)	Q(t + 1)	Q (t + 1)
0	0	0	0	1
0	0	1	1	0
0	0	0	0	1
0	1	1	0	1
0	1	0	1	0
1	0	1	1	0
1	0	0	1	0
1	1	1	0	1
1	1	0	1	0



$$\bar{Q}(t+1) = \bar{J}\,\bar{Q} + \bar{K}Q$$

Q.16. Explain four bit Bi-directional Register with the help of Multiplexers. (2015)

Ans. Fig shows a 4-bit bi-directional shift register. This register can serially shift data right or left, we need one control signal to perform the bidirectional data shift because we cannot shift the data simultaneously. The control signal RIGHT/LEFT is used to shift the data right side or left side. When the Right/Left control input is High, gates G_1, G_2, G_3 and G_4 are enabled and the output of each flip flop is moved through to the D input, when the clock pulse is applied, the data bits are shifted from left to right side. When the Right/left control input is LOW, gates G_5, G_6, G_7 and G_8 are enabled and flip flop output of each bits is moved towards left side.

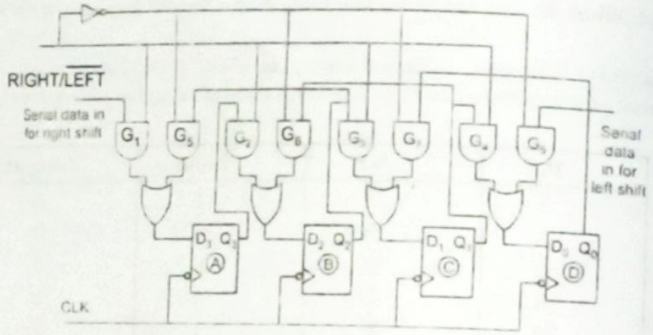
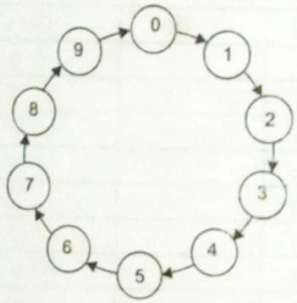


Fig. Logic Diagram for 4-bit bidirectional shift register

Q.17. Design a decade synchronous UP counter. Use T flip-flops (2015),(2017),(2018

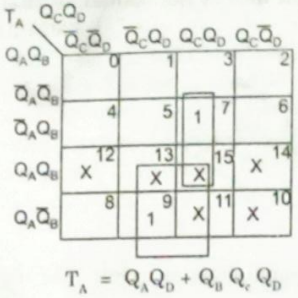
Ans. Decade synchronous up counter counts 0000 to 1001



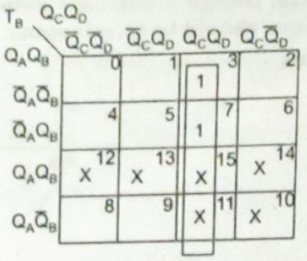
State table

P.S.				N.S.				Flip-flop inputs			
Q_A	Q_B	Q_C	Q_D	Q_{A+1}	Q_{B+1}	Q_{C+1}	Q_{D+1}	T_A	T_B	T_C	T_D
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	0	0	0	0	1	0	0	1

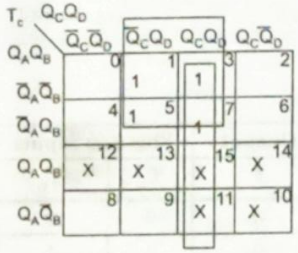
The unused 10,11,12,13,14 and 15 states are used as don't care in K-map simplification



$T_A = Q_A Q_D + Q_B Q_C Q_D$

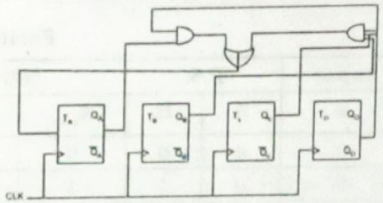


$T_B = Q_C Q_D$



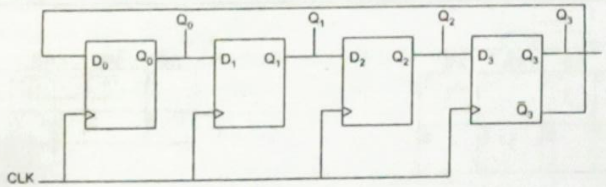
$T_C = Q_C Q_D + \bar{Q}_A Q_D$ and $T_D = 1$

Logic diagram



Q.18. Explain Twisted Ring counter with the help of Timing Diagram. (2015)

Ans. It is a variation of a shift register counter. In this counter, the complement of the output of the last flip-flop is connected back to the input of the first flip-flop. This counter have basic counting cycles of length 2N, where N is the number of flip-flop. The logic circuit of 4-bit Johnson counter is shown below.

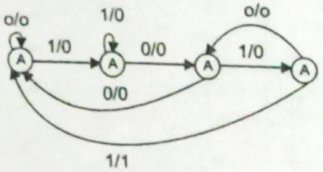


Truth Table

CLK	Q_0	Q_1	Q_2	Q_3	$\bar{Q}_3$
0	0	0	0	0	1
1	1	0	0	0	1
2	1	1	0	0	1
3	1	1	1	0	1
4	1	1	1	1	0
5	0	1	1	1	0
6	0	0	1	1	0
7	0	0	0	1	0
8	0	0	0	0	1

Q.19. Design a sequence detector that will detect the sequence 1011 and sequence should be Overlapping. **(2015),(2016)**

Ans. Given sequence is 1011



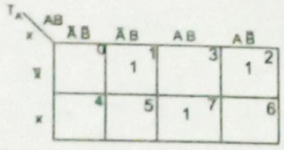
Assuming two flip-flops to construct the sequence detector circuit. The sequence detector has a total of 4 states.

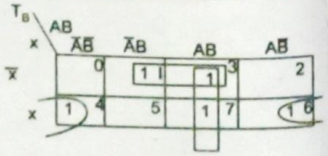
We choose T flip flop

Let A = 00, B = 01, C = 10, D = 11

Excitation table

Input	P.S.		N.S		Output	Flip-flop inputs	
x	A	B	A_{n+1}	B_{n+1}	Y	T_A	T_B
0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1
0	1	0	0	0	0	1	0
0	1	1	1	0	0	0	1
1	0	0	0	1	0	0	1
1	0	1	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	0	1	1	1



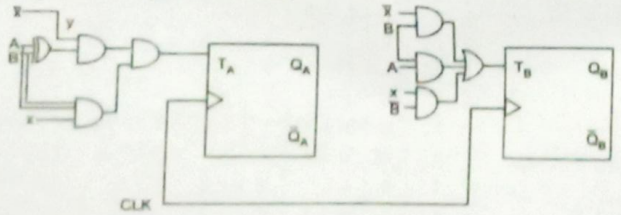


$T_A = \bar{x}\bar{A}B + \bar{x}A\bar{B} + xAB = \bar{x}(A \oplus B) + xAB$

$T_B = \bar{x}B + AB + x\bar{B}$

and $Y = xAB$

Logic diagram for sequence detector



Q.20. What is race around condition? How can we overcome this condition? **(2014),(2016),(2017)**

Ans. Race Around Condition: It is important to note that in JK flip-flop, the output is feedback to the input, and therefore change in the-output results in a change in the input, if the inputs J = K = 1 and Q = 0.

When a clock pulse with width t_p as shown in Fig. (a) is applied, the output will change from 0 to 1 after the time interval Δt,

where Δt = propagation delay of two level NAND gates

t_p = pulse width

Now, after Δt, we have J = K = 1 and Q = 1 after another Δt, output Q will become 0. Hence, the output will oscillate back and forth between 0 and 1 in the duration tp of the clock pulse width. So, at the end of the clock pulse, the value of Q is ambiguous. This is known as a "race-around condition".

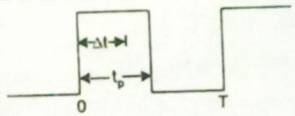


Fig. (a)

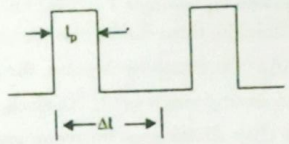


Fig (b)

The race around condition can be avoided when $t_p < \Delta t$ as shown in Fig. (b). This condition can be obtained by two ways.

1. If t_p is reduced 2. If Δt is increased

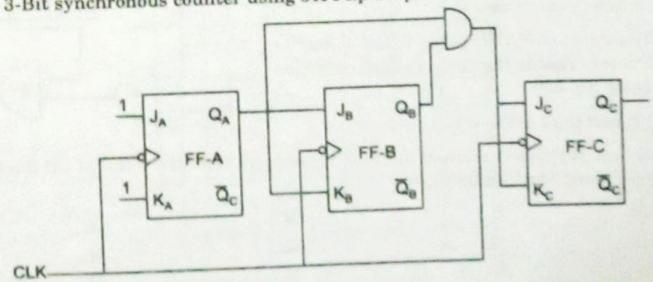
Reduction of t_p means, we have to use a pulse generator to produce less pulse width waveform, but is difficult to get such type of circuit. The value of Δt can be increased by using the lumped delay in series with feedback connection, which is again worthless.

There are two solutions for reducing race around condition

1. Use of edge triggering, which we have already stated
2. Master slave flip flop configuration

Q.21. Design a 3 bit synchronous counter using JK Flip Flop. **(2016)**

Ans. 3-Bit synchronous counter using JK Flip-Flop



Operation: Initially all the FFs are in their reset state. Therefore Q_C Q_B Q_A = 000.

(i) When clock is applied, FF-A toggles and Q_A changes to 1 from 0. Therefore Q_C Q_B Q_A = 001

Truth Table

Clock	Q_C	Q_B	Q_A
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

(ii) When second clock pulse is applied FF-A toggles and Q_A becomes 'O', and FF-B will toggle and Q_B becomes 1. Therefore $Q_C\ Q_B\ Q_A = 010$.

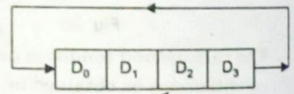
(iii) After the third clock pulse, the outputs are $Q_C\ Q_B\ Q_A = 011$.

(iv) After the fourth clock pulse, the outputs are $Q_C\ Q_B\ Q_A = 100$.

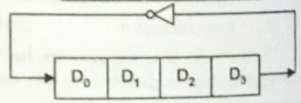
This counting progress till 7th clock.

Q.22. Give difference between ring and Twisted ring Counter. **(2016)**

Ans. In ring counter the 'Q_n' output of last stage flip-flop is connected to the input of first flip-flop.

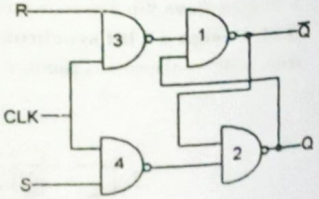


In twisted ring counter the '$\bar{Q}n$' output of last stage flip-flop is connected to the input of first flip-flop.



Q.23. Design an SR Flip Flop and explain how it works using truth table. Show how an SR Flip Flop can be converted to D Flip Flop. **(2016)**

Ans. SR flip-flop using NAND gate



Operation:

I. When clock pulse = 0.

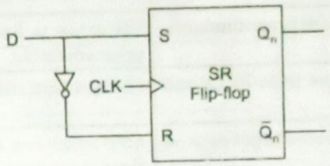
The output of NAND gates 3 and 4 stay at logic '1' level. This is the "no" change" condition of the basic flip-flop.

II. When clock pulse = 1

The S or R input is allowed to reach the output. Now the effect of SR flip-flop can be explain using truth-table below

CLK	S	R	Q_{n+1}
0	X	X	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	not used

Conversion of SR into D.



Q.24. Discuss types of shift registers. **(2016),(2017)**

Ans. Classification of Shift Registers

I. Classification based on the direction of data movement

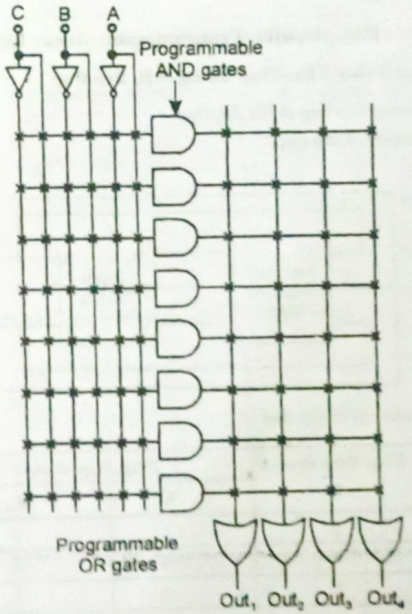
1. Shift left register 2. Shift right register 3. Bidirection shift register

II. Classification based on the mode of input and output

1. Serial in serial out shift register (SISO)
2. Serial in parallel out shift register (SIPO)
3. Parallel in serial out shift register (PISO)
4. Parallel in parallel out shift register (PIPO)
5. Universal shift register.

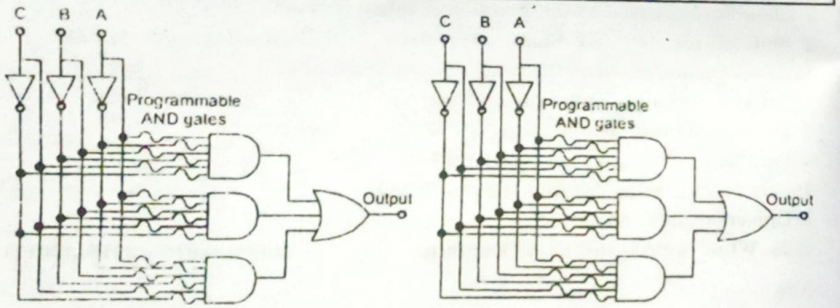
Q.25. What is PAL and PLA? Explain. **(2014),(2015),(2016),(2017)**

Ans.



Structure of (an unprogrammed) PLA circuit

PLA	PAL
Both AND and OR arrays are programmable.	OR array is fixed and AND array is programmable.
Costliest and more complex than PALs and PROMs.	Cheaper and simpler.
AND array can be programmed to get desired minterms.	AND array can be programmed to get desired minterms.
Any Boolean function in SOP from van be implemented using PLA.	Any Boolean function in SOP from can be implemented using PAL.



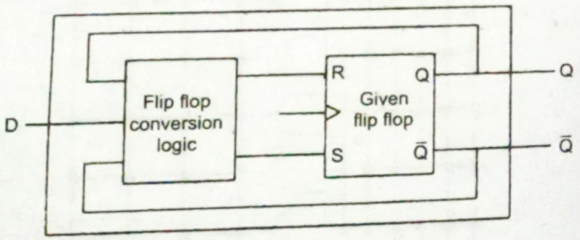
PAL circuits (Programmable Array Logic)

Q.26. Design a Delay Flip-Flop using S-R flip flop.

Ans. Step1. Given flip flop is RS flip flop

Required flip flop is D flip flop

Block diagram



Step 2. Truth table of D flip flop

Flip flop input D	Flip flop states Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

Excitation table of RS flip flop

Present state Q_n	Next state Q_{n+1}	Flip flop inputs R	S
0	0	X	0
0	1	0	1
1	0	1	0
1	1	0	X

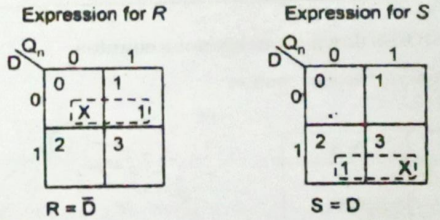
Step 3: Conversion table

Compute the flip flop inputs by using excitation table of RS flip flop

Required flip flop			Given flip flop	
Flip flop input	Present state	Next state	Flip flop inputs	
D	Q_n	Q_{n+1}	R	S
0	0	0	X	0
0	1	0	1	0
1	0	1	0	1
1	1	1	0	X

Step 4: K-map simplification

The required flip flop inputs and present state are considered for K-map simplification.



Step 5: The obtained expression are S = D and R = $\bar{D}$

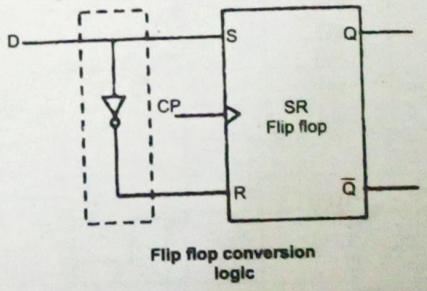


Fig.

Example. Convert the RS flip flop to T flip flop.

Step 1: Given flip flop is RS flip flop

Required flip flop is T flop

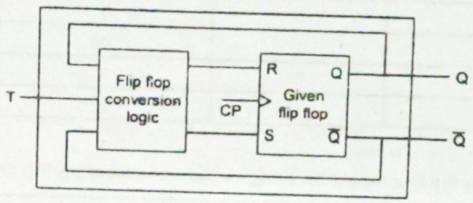


Fig. (a)

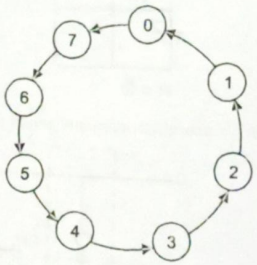
Step 2. Truth table of T flip flop

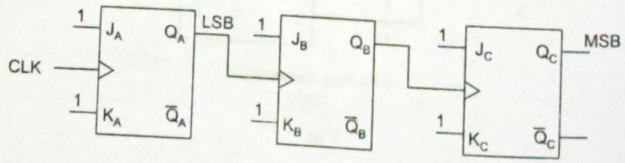
Flip flop input	Present state	Next state
T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

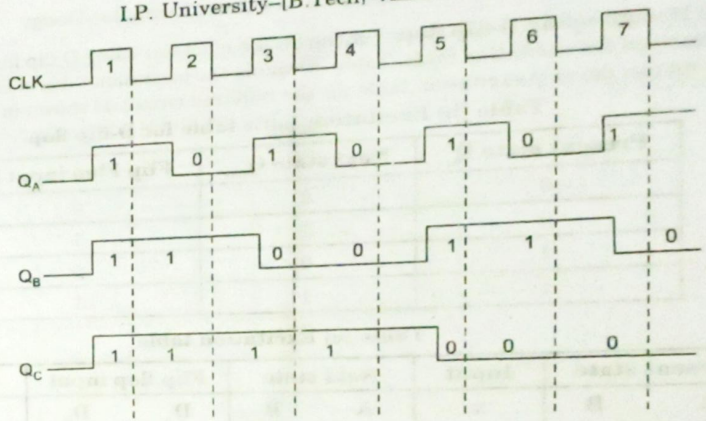
Q.27. Design a MOD-8 down asynchronous counter. **(2016)**

Ans. Mod-8 down asynchronous counter

P.S.	N.S
000	111
001	000
010	001
011	010
100	011
101	100
110	101
111	110







Q.28. A sequential circuit has one input and one output and its state diagram is shown in Fig. (a). Design the sequential circuit using (i) D flip flop and (ii) JK flip flop. **(2016)**

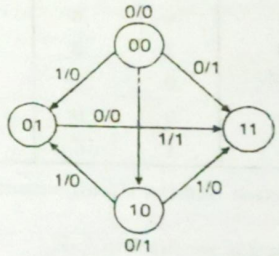


Fig. (a)

Ans. The given state diagram consist of four states. It has one input (x) and one output y. The state table for the given state diagram is shown in Table (a). It is clear that there are no equivalent states. Therefore, there is no reduction in the state diagram. As the state diagram contains 4-states, it required 2 flip-flop which are named as A and B.

Table (a)

Present state	Next state		Output	
	x = 0	x = 1	x = 0	x = 1
AB	AB	AB	y	Y
00	00	10	0	1
01	11	00	0	0
10	10	01	1	0
11	00	10	1	0

(i) Design using D-flip flop: For the design of circuit using D flip flop (or any flip flop), we need the excitation table. Table (b) shows the excitation table of Dflip flop from which we can develop excitation table for the required circuit as shown in Table (c).

Table (b) Excitation table table for D-flip flop

Present state Q_n	Next state Q_{n+1}	Flip Flop input D
0	0	0
0	1	1
1	0	0
1	1	1

Table (c) Excitation table

Present state		Input	Next state		Flip flop input		Output
A	B	x	A	B	D_A	D_B	y
0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	1
0	1	0	1	1	1	1	0
0	1	1	0	0	0	0	0
1	0	0	1	0	1	0	1
1	0	1	0	1	0	1	0
1	1	0	0	0	0	0	1
1	1	1	1	0	1	0	0

The flip-flop input function and the circuit output functions are obtained by using K-map simplification.

Input equation (or) function for flip flop A (Z_D)

$$DA = \bar{A}B\bar{x} + \bar{A}\bar{B}x + A\bar{B}\bar{x} + ABx$$

$$= \bar{A}(\bar{B}x + B\bar{x}) + A(\bar{B}\bar{x} + Bx)$$

Considering $z = \bar{B}x + B\bar{x}$, then $\bar{B}\bar{x} + Bx = \bar{z}$

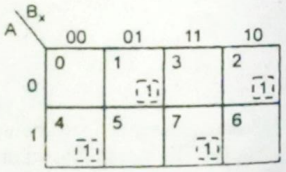
Simplify the above equation

$$D_A = \bar{A}z + A\bar{z}$$

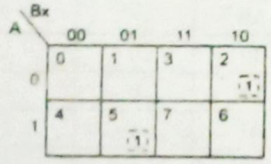
$$= A \oplus z$$

Substitute $z = \bar{B}x + B\bar{x} = B \oplus$ in the above equation

$$\boxed{D_A = A \oplus B \oplus x}$$

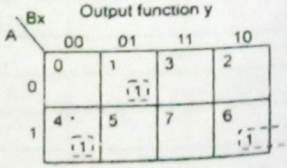


Input equation for flip flop B (D_B)



$D_B = \bar{A}B\bar{x} + A\bar{B}x$

Output function y



$Y = A\bar{x} + \bar{A}\bar{B}x$

The input equation for flip flop and output equation are summarized as follows

$D_A = A \oplus B \oplus x$ $D_B = \bar{A}B\bar{x} + A\bar{B}x$ $y = A\bar{x} + \bar{A}Bx$

A sequential circuit using D flip-flop is obtained by using above equations as shown in Fig. (b).

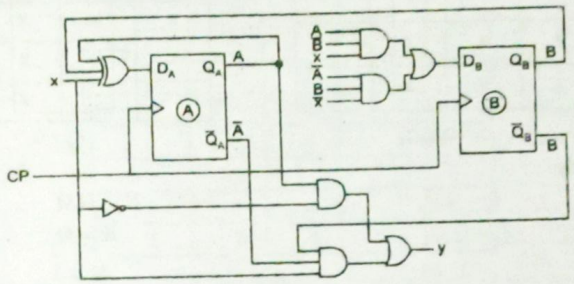
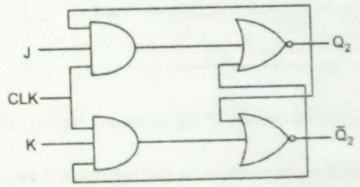


Fig. (*b*) Sequential logic diagram using *D* flip-flop.

Q.29. Explain JK flip-flop with the help of truth table, characteristic equation and waveform. **(2017)**

Ans. JK flip Flop: It is used to remove the invalid condition of S-R Flip-Flop. The logic diagram and truth table is shown below.



CLK	J	K	Q_{n+1}
0	X	X	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	$\bar{Q}_n$

Q.30. Design a synchronous BCD counter with JK flip-flop. **(2017)**

Ans. A BCD counter is nothing but a mod-10 counter. It has 10 states (0000 to 1001). It requires n = 4 flip flop.

PS	NS	Required Corresponding Excitations			
$Q_4\ Q_3\ Q_2\ Q_1$	$Q_4\ Q_3\ Q_2\ Q_1$	$J_4\ K_4$	$J_3\ K_3$	$J_2\ K_2$	$J_1\ K_1$
0 0 0 0	0 0 0 1	0 ×	0 ×	0 ×	1 ×
0 0 0 1	0 0 1 0	0 ×	0 ×	1 ×	× 1
0 0 1 0	0 0 1 1	0 ×	0 ×	× 0	1 ×
0 0 1 1	0 1 0 0	0 ×	1 ×	× 1	× 1
0 1 0 0	0 1 0 1	0 ×	× 0	0 ×	1 ×
0 1 0 1	0 1 1 0	0 ×	× 0	1 ×	× 1
0 1 1 0	0 1 1 1	0 ×	× 0	× 0	1 ×
0 1 1 1	1 0 0 0	0 ×	× 1	× 1	× 1
1 0 0 0	1 0 0 1	× 0	0 ×	0 ×	1 ×
1 0 0 1	0 0 0 0	× 1	0 ×	0 ×	× 1

Q_4Q_3 \ Q_2Q_1	00	01	11	10
00				
01			1	
11	X	X	X	X
10	X	X	X	X

$J_4 = Q_3Q_2Q_1$

Q_4Q_3 \ Q_2Q_1	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	X	X
10		1	X	X

$K_4 = Q_1$

Similarly,

$J_3 = Q_2Q_1$ $\quad$ $K_3 = Q_2Q_1$

$J_2 = \bar{Q}_4Q_1$ $\quad$ $K_2 = Q_1$

$J_1 = 1$ $\quad$ $K_1 = 1$

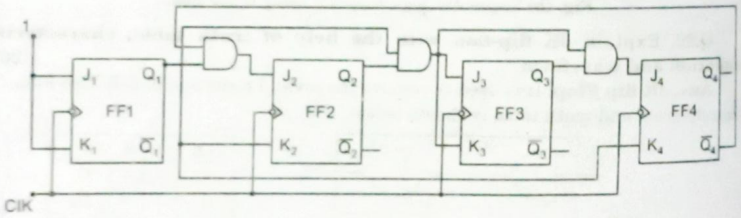


Fig. Logic Diagram for synchronous BCD counter using JK Flip-Flop

Q.31. How ring counter will be converted in to Johnson's counter. (2017)

Ans. In the shift register, if $\bar{Q}_0$ is connected to the serial input the resulting circuit is referred to as a Twisted Ring Johnson counter.

Q.32. What are the applications of shift register? (2017)

Ans. Applications of shift registers are; Delay line, Serial to parallel convertor, Parallel to serial convertor, Ring counter, Johnson counter and sequence generator etc.

Q.33. Write the difference between RAM and ROM? (2017)

Ans.

RAM	ROM
(i) Random Access Memory	(i) Read Only memory
(ii) Ram allows the computer to read data quickly to run applications.	(ii) ROM stores the program required to initially boot the computer.
(iii) RAM is volatile	(iii) ROM is non-volatile.

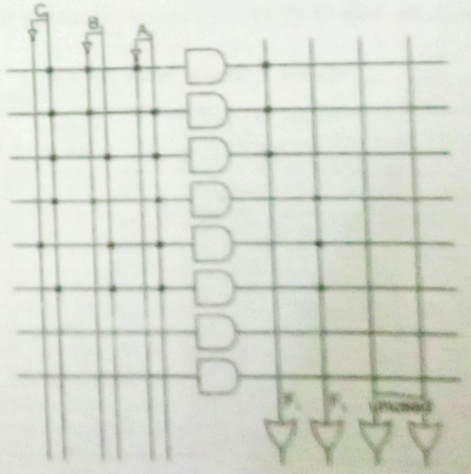
Q.34. A combinational circuit is defined by the function $F_1 = \Sigma m(1, 5, 7)$, $F_2 = \Sigma m(5, 6, 7)$ implement the circuit with a PLA. (2017)

Ans. $F_1 = \Sigma m(1, 5, 7)$

Inputs			Output
A	B	C	F_1
0	0	0	0
0	0	1	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$F_2 = \Sigma m(5, 6, 7)$

Inputs			Output
A	B	C	F_2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Q.35. Draw the circuit diagram of a Master Slave J K flip flop and explain its operation with the help of truth table, how is it different from the edge triggering explain. (2017)

Ans. J-k FLIP-FLOP: The uncertainty in the state of an S–R FLIP-FLOP when $S_n = R_n = 1$ (fourth row of the truth table) can be eliminated by converting it into a J–K FLIP-FLOP. The data inputs are J and K which are ANDed with $\bar{Q}$ and Q, respectively, to obtain S and R inputs i.e.,

$$S = J\bar{Q} \quad \text{(i)}$$

$$R = K.Q \quad \text{(ii)}$$

A J–K FLIP-FLOP thus obtained is shown in Figure 1. Its truth table is given in Table 1 which is reduced to Table 2 for convenience. Table 1 has been prepared for all the possible combinations of J and K inputs, and for each combination both the states of the output have been considered.

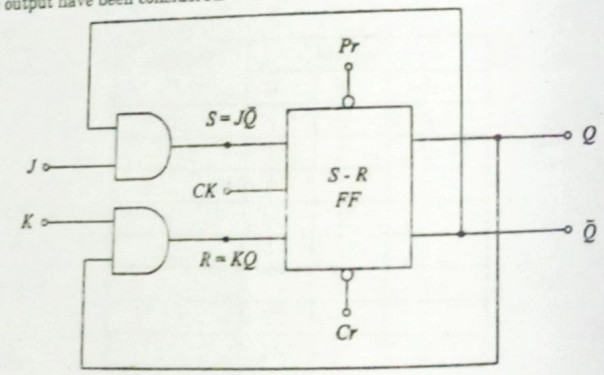


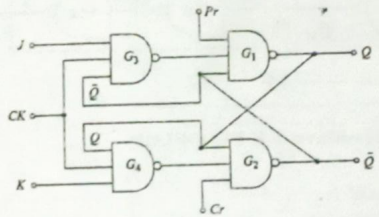
Fig. 1: An S–R FLIP-FLOP Converted into J–K FLIP-FLOP

Table 1: Truth Table for Fig. 1

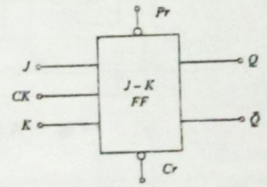
Data inputs		Outputs		Inputs to S–R FF		Output
J_n	K_n	Q_n	$\bar{Q}_n$	S_n	R_n	Q_{n+1}
0	0	1	0	0	0	Q_n
0	0	0	1	0	0	Q_n
1	0	1	0	1	0	1
1	0	0	1	0	0	1
0	1	1	0	0	0	0
0	1	0	1	0	1	0
1	1	1	0	1	0	$\bar{Q}_n$
1	1	0	1	0	1	$\bar{Q}_n$

Table 2: Truth table of J-K FLIP-FLOP

Inputs		Output
J_n	K_n	Q_{n+1}
0	0	Q_n
1	0	1
0	1	0
1	1	$\bar{Q}_n$



A J-K FLIP-FLOP Using NAND Gates



Logic Symbols of J–K FLIP-FLOP

The Race-Around Condition: The difficulty of both inputs 1(S = R = 1) being not allowed in an S–R FLIP-FLOP is eliminated in a J–K FLIP– FLOP by using the feedback connection from outputs to the inputs of the gates G_3 and G_4 figure 2. Table 1 and 2 assumes that the inputs do not change during the clock pulse (CK = 1), which is not true because of the feedback connections. Consider, for example, that the inputs are J = K = 1 and Q = 0, and a pulse as shown in Figure 2 is applied at the clock input. After a time interval Δt equal to the propagation delay through two NAND gates in series, the output will' change to Q = 1 see fourth row of Table 2. Now we have J= K = 1 and Q = 1 and after another time interval of Δt the output will change back to Q = 0. Hence, we conclude that for the duration t_p of the clock pulse, the output will oscillate back and forth between 0 and 1. At the end of the clock pulse, the value of Q is uncertain. This situation is referred to as the race-around condition.

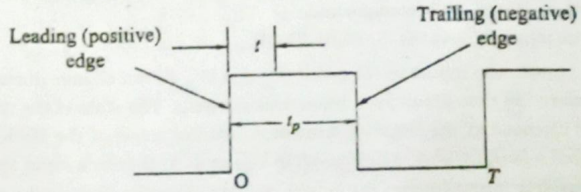


Fig. 2: A Clock Pulse

The race-around condition can be avoided if $t_p < \Delta t < T$. However, it may be difficult to satisfy this inequality because of very small propagation delays in ICs.

The Masters-Slave J–K FLIP–FLOP

A master-slave J–K FLIP-FLOP is a cascade of two S–R FLIP-FLOPs, with feedback from the outputs of the second to the inputs of the first as illustrated in Figure 5. Positive clock pulses are applied to the first FLIP–FLOP and the clock pulses are inverted before these are applied to the second FLIP–FLOP.

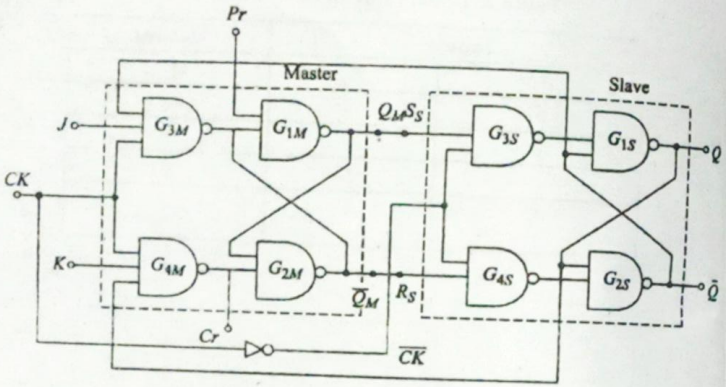


Fig. 3: A Master-Slave J–K FLIP-FLOP

When CK = 1, the first FLIP–FLOP is enabled and the outputs Q_M and $\bar{Q}_M$ respond to the inputs J and K according to Table 1, At this time, the second FLI P-FLOP is inhibited because its clock is LOW ($\overline{CK} = 0$). When CK goes LOW ($\overline{CK} = 1$), the first FLIP–FLOP is inhibited and the second FLIP–FLOP is enabled, because now its clock is HIGH ($\overline{CK} = 1$). Therefore, the outputs Q and $\bar{Q}$ follow the outputs Q_M and $\bar{Q}_M$, respectively (second and third rows of Table 2). Since the second FLIP–FLOP simply follows the first one, it is referred to as the slave and the first one as the master. Hence, this configuration is referred to as master–slave (M–S) FLIP–FLOP.

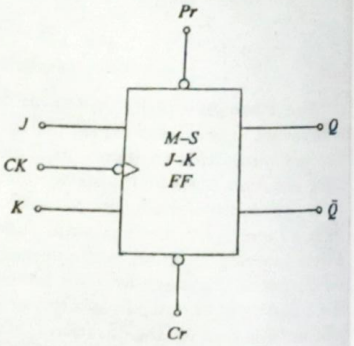


Fig. 4. A Master-Slave J-K flip flop logic symbol

In this circuit, the inputs to the gates G_{3M} and G_{4M} do not change during the clock pulse, therefore the race-around condition does not exits. The state of the master-slave FLIP-FLOP changes at the negative transition (trailing edge) of the clock pulse. The logic symbol of a M–S FLIP–FLOP is given in Figure 4. At the clock input terminal, the symbol > is used to illustrate that the output changes when the clock makes a transition and the accompanying bubble signifies negative transition (change in CK from 1 to 0).

Q.36. Design a 3 bit binary UP/Down synchronous counter with direction control signal M when M = 0 for up counting and M = 1 for down counting. (2017)

Ans. The count sequence is given in Table 1. For M = 0, it acts as an UP counter and for M = 1 as a DOWN counter. The number of FLIP-FLOPs required is 3. The inputs of the FLIP-FLOPs are determined in a similar manner.

Table 1

Direction control	Counter state			FLIP-FLOP inputs					
M	Q_2	Q_1	Q_0	J_0	K_0	J_1	K_1	J_2	K_2
0	0	0	0	1	×	0	×	0	×
0	0	0	1	×	1	1	×	0	×
0	0	1	0	1	×	×	0	0	×
0	0	1	1	×	1	×	1	1	×
0	1	0	0	1	×	0	×	×	0
0	1	0	1	×	1	1	×	×	0
0	1	1	0	1	×	×	0	×	0
0	1	1	1	×	1	×	1	×	1
1	0	0	0	1	×	1	×	1	×
1	1	1	1	×	1	×	0	×	0
1	1	1	0	1	×	×	1	×	0
1	1	0	1	×	1	0	×	×	0
1	1	0	0	1	×	1	×	×	1
1	0	1	1	×	1	×	0	0	×
1	0	1	0	1	×	×	1	0	×
1	0	0	1	×	1	0	×	0	×
	0	0	0						

From Table 1. we obtain $J_0 = K_0 = 1$

The K-maps for J_1, K_1, J_2 and K_2 are shown in Figure 1. From the K-maps the minimized expression are obtained as

$J_1 = K_1 = Q_0\bar{M} + \bar{Q}_0 M \qquad J_2 = K_2 = \bar{M}Q_1Q_0 + M\bar{Q}_1\bar{Q}_0$

The counter circuit can be drawn using the above expressions

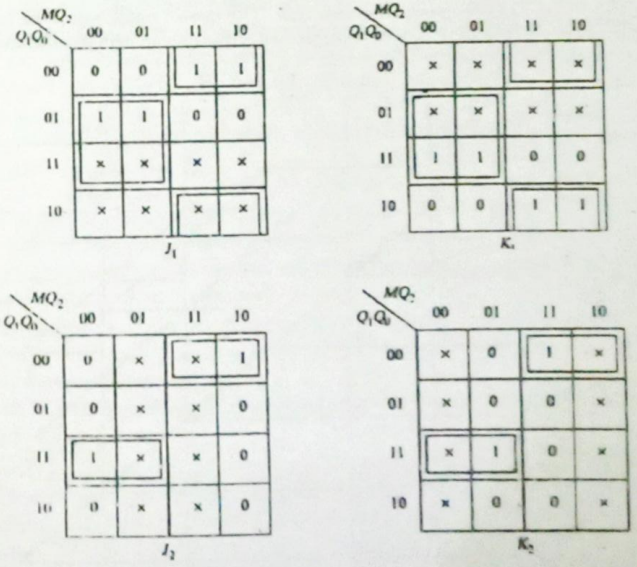


Fig. 1 K-Maps

Q.37. Explain the working of serial in an serial out shift register with logic diagram and wave form. **(2017**

Ans. Shift Register: A 5-bit shift register using five master-slave S–R (or J – K) FLIP-FLOPs is shown in Figure 1. This circuit can be used in any of the four modes. The operation of this circuit is explained by assuming the 5-bit data 10110. For any other 5-bit data, the operation will be similar to the one explained.

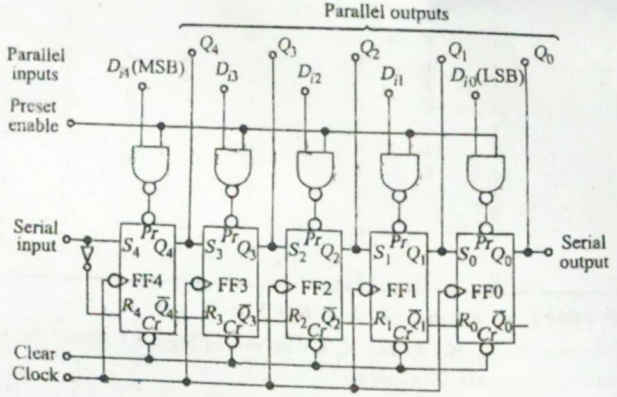


Fig. 1: A 5-bit Shift Register (7496)

Serial Input: The data word in the serial form is applied at the serial input after clearing the FLIP-FLOPs using the clear line. The preset enable is to be held at 0 so that Pr for every FLIP-FLOP is 1. The input and output waveform are illustrated Fig 2.

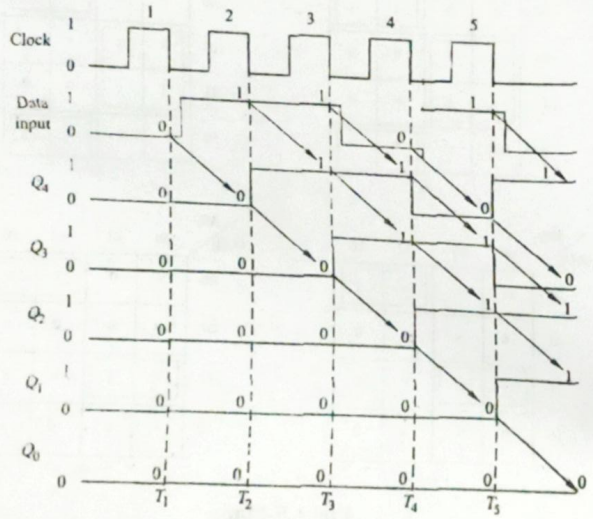


Fig. 2: Waveforms of shift register for serial input

The process of entering the digital word starts with the data input corresponding to the least-significant bit (0) at the serial input and first clock pulse. At the falling edge (T1) of the first clock pulse the output of FF4 (Q_4) will be 0 and the outputs of all other FLIP-FLOPs are 0 since their inputs are 0. Next, the input corresponding to next bit is applied and at the falling edge (T_2) of the second clock pulse, the FLIP-FLOP outputs will be

$$Q_4 = 1$$
$$Q_3 = Q2 = Q1 = Q0 = 0$$

Similarly, the input corresponding to each bit is applied till the MSB and the bits go on shifting from left to right at the falling edge of each clock pulse as illustrated in Fig 2. At the end of fifth clock pulse, the outputs of FLIP-FLOPs are which is the same as the number to be stored. The number of clock pulse required for entering the data, is the same as the number of bits. The process of entering the data is also referred to as writing into the register.

$$Q_4 = 1$$
$$Q_3 = 0$$
$$Q_2 = 1$$
$$Q_1 = 1$$
$$Q_0 = 0$$

The data stored can be retrieved (also referred to as reading) in two ways: serial-out and parallel-out. The data in the serial form is obtained at Q_0 when clock pulses are applied. The number of clock pulses required will be same as the number of bits (five in this case).

In the parallel form, the data is available at $Q_4\ Q_3\ Q_2\ Q_1\ Q_0$ and clock is not required for reading. In the case of serial output, after the nth clock pulse, for an n-bit word, each FLIP-FLOP output is 0. This means that once the data is retrieved the register is empty. On the other hand, in the case of parallel output, the contents of the register can be read any number of times until new data is stored in the register.

The clock rate may be different for the input data and the output data in case of a serial-in, serial-out shift register. Hence, this method can be used for changing the spacing in time of a binary code which is referred to as buffering.

Parallel Input: Data can be entered in the parallel form making use of the preset inputs. After clearing the FLIP-FLOPs if the data lines are connected to the parallel inputs (D_{i4}, D_{i3} D_{i2}, D_{i1}, and D_{i0}) and a 1 is applied at the preset input, the data are written into the register. This is referred to as asynchronous loading.

The stored word may be read in the serial form at Q_0 by applying five clock pulses or in the parallel form at Q outputs.

The data can also be entered in parallel form by using D-type FLIP-FLOPs connected as shown, The data is loaded when a clock pulse is applied and hence, it is referred to as synchronous loading.

Q.38. An a synchronous sequential circuit is described by excitation function

$Y = X_1\bar{X}_2 + (X_1\bar{X}_2)Y$ and output function $Z = Y$ **(2017)**

(i) Draw the logic diagram of the circuit. (ii) Derive the transition table and output (iii) Obtain the two state flow tables.

Ans.

$$Y = X_1\bar{X}_2 + (X_1\bar{X}_2)Y$$
$$Z = Y$$

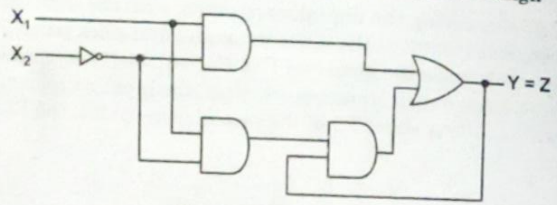
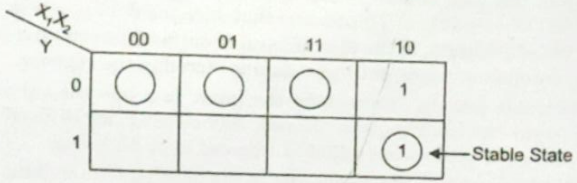


Fig. Logic Diagram

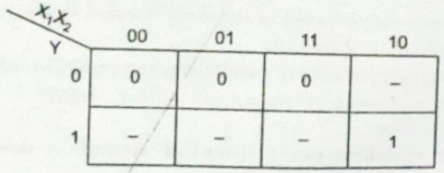
Table: Transition Table and its corresponding output

Present state			Next state			Stable state	Z
X_1	X_2	Y	X_1	X_2	Y	Yes/No	
0	0	0	0	0	0	Yes	0
0	0	1	0	0	0	No	0
0	1	0	0	1	0	Yes	0
0	1	1	0	1	0	No	0
1	0	0	1	0	1	No	1
1	0	1	1	0	1	Yes	1
1	1	0	1	1	0	Yes	0
1	1	1	1	1	0	No	0



K-Map from Transition Table (uncircled)

Circle shows stable state. And the rest (uncircled) is unstable state.



Output map

Output is mapped for all stable states, for unstable states output is mapped unspecified.

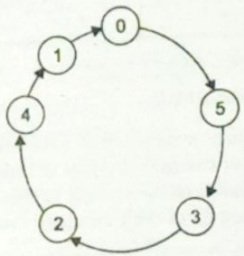
Q.39. Differentiate between combinational logic circuit and sequential logic circuit. **(2015),(2018)**

Ans.

Combination Circuit	Sequential Circuit
1. Output is depends only on present input.	1. Output depends on present input and past output.
2. No need of memory element.	2. Memory element is needed.
3.Faster is speed.	3. Slower than combination circuit.
4. Designer has less flexibility.	4. Designer has more flexibility.

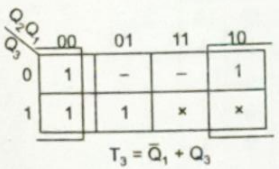
Q.40. Design a MOD - 6, synchronous counter to count in the sequence 0, 5, 3, 2, 4, 1 and 0. Use T flip-flops. **(2018)**

Ans. This counter has only six stable states, but it requires three FFs.

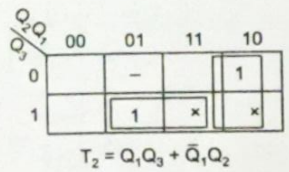


PS			NS		
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1
0	0	0	1	0	1
1	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	0	0
1	0	0	0	0	1
0	0	1	0	0	0

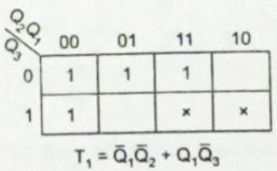
Required Excitations		
T_3	T_2	T_1
1	0	1
1	1	0
0	0	1
1	1	0
1	0	1
0	0	1



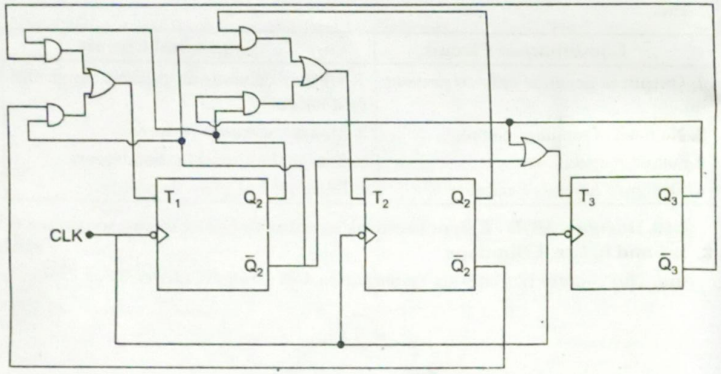
$T_3 = \bar{Q}_1 + Q_3$



$T_2 = Q_1Q_3 + \bar{Q}_1Q_2$

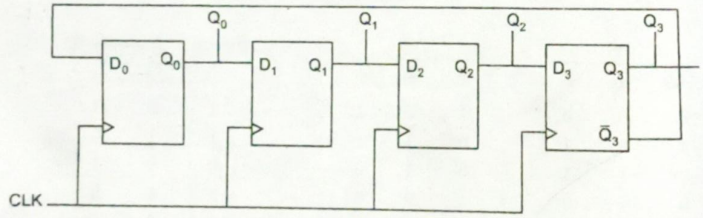


$T_1 = \bar{Q}_1\bar{Q}_2 + Q_1\bar{Q}_3$



Q.41. What is Johnson counter? **(2018)**

Ans. Johnson Counter: It is a variation of a shift register counter. In this counter, the complement of the output of the last flip-flop is connected back to the input of the first flip-flop. This counter have basic counting cycles of length 2N, where N is the number of flip-flop. The logic circuit of 4-bit Johnson counter is shown below.



Truth Table

CLK	Q_0	Q_1	Q_2	Q_3	$\bar{Q}_3$
0	0	0	0	0	1
1	1	0	0	0	1
2	1	1	0	0	1
3	1	1	1	0	1
4	1	1	1	1	0
5	0	1	1	1	0
6	0	0	1	1	0
7	0	0	0	1	0
8	0	0	0	0	1

Q.42. Distinguish between (i) SRAM and DRAM (ii) PROM and EPROM **(2018)**

Ans.

SRAM	DRAM
The memory cell is essential a latch can store data indefinitely, as long as the DC power is supplied.	Memory cell that stores data in the form of charge on a capacitor.
Duration of retaining data is long.	Cannot retain data long.
Higher in speed to perform operation	Less speed as compare to SRAM.
Less storage capacity as compare to DRAM	More storage capacity.
PROM	**EPROM**
Chip is one time programmable	Chip is reprogrammable.
Inexpensive	Costly as compared to PROM.
Storage high	Low storage.

Q.43. Design a Mod-5 asynchronous counter using T-FF.

Ans. MOD-5 Asynchronous counter using T-FF: A mod-5 counter has five stable states 000, 001, 010, 011, 100. when the fifty pulse is applied the counter temporily goes to 101 state, but immediately resets to 000 because of the feedback provided.

R = 0 for 000 to 100, R=1 for 101, and 110 and R = X for 111

$$R = Q_3\bar{Q}_2Q_1 + Q_3Q_2\bar{Q}_1 + Q_3Q_2Q_1 = Q_3\bar{Q}_2Q_1 + Q_3Q_2 = Q_3(\bar{Q}_2Q_1 + Q_2)$$

Table of R

After Pulses	State Q_3	Q_2	Q_1	R
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

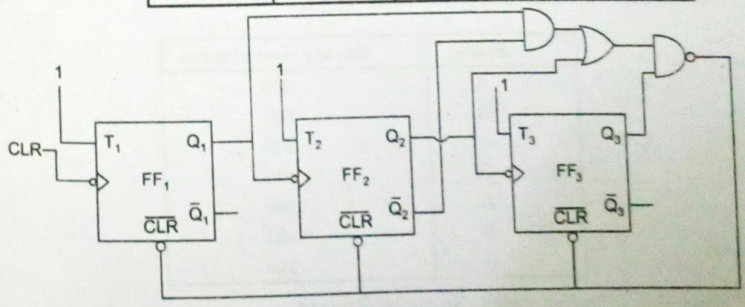
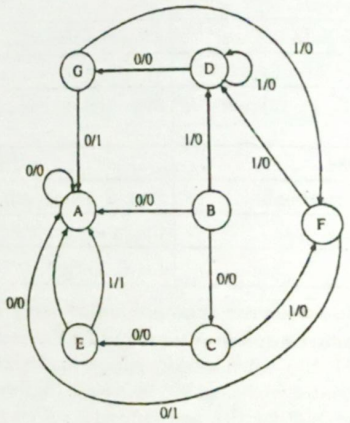


Fig. Logic Diagram

Q.44. Design a 2-input 2-output synchronous sequential circuit which produces an output z = 1, whenever any input sequence –1100, 1010 or 1000 occurs. The circuit resets to the initial state after output reaches 1.

(2014), (2016)

Ans. (i) The following state diagram can be obtained for the required function.



Using this, following state table can be obtained.

Present State	Next State		Output z	
	x=0	*x=1*	*x=0*	*x=1*
A	A	B	0	0
B	C	D	0	0
C	E	F	0	0
D	G	D	0	0
E	A	A	0	1
F	A	D	1	0
G	A	F	1	0

(ii) Consider the following assignment:

State	Binary Assignment
A	000
B	001
C	010
D	011
E	100
F	101
G	110

Using this, following state table can be obtained.

Present State $Q_2, Q_1 Q_0$	Next State $Q_2^+ Q_1^+ Q_0^+$		Output z	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
000	000	001	0	0
001	010	011	0	0
010	100	101	0	0
011	110	011	0	0
100	000	000	0	1
101	000	011	1	0
110	000	101	1	0

Exciting Table

P.S.	N.S.		Flip-Flop inputs									
	$x = 0$	$x = 1$	$x = 0$				$x = 1$					
$Q_2, Q_1 Q_0$	$Q_2^+ Q_1^+ Q_0^+$	$Q_2^+ Q_1^+ Q_0^+$	J_2	k_2	J_1	k_0	J_2	k_2	J_1	k_1	J_0	k_0
000	000	001	0	X	0	X	0	X	0	X	1	X
001	010	011	0	X	1	X	X	1	0	X	1	X
001	100	101	1	X	X	1	0	X	1	X	X	0
011	110	011	1	X	X	0	X	1	0	X	0	X
100	000	000	X	1	0	X	0	X	X	1	0	X
101	000	011	X	1	0	X	X	1	X	1	X	0

After solving k-maps we get

$$J_2 = (\bar{x} + \bar{Q}_0)Q_1 \quad k_2 = (\bar{x} + \bar{Q}_1)Q_2$$

$$J_1 = (x + \bar{x}_2)Q_0 \quad k_1 = (\bar{x} + \bar{Q}_1)Q_2$$

$$J_0 = (Q_1 + \bar{Q}_2)x \quad k_0 = \bar{x}$$

UNIT–III

Q.1. Represent the floating point in the IEEE standard format for the given number $1.00010100 \times 2^{-10}$? **(2015)**

Ans. The IEEE 754 single precision representation is given by

1	00010100	10000000000
1 bit	8 bits	23 bits
S	E	M

Q.2. Represent the following conditional statement by two register transfer statements with control functions. **(2015)**

If (P = 1) then (R1 → R2) else if (Q = 1) then (R1 ← R3)

Ans. P : R1 ← R2 ; If (P = 1) then (R1 ← R2)

P'Q : R1 ← R3 ; else if (Q = 1) the (R1 ← R3)

Q.3. Starting from an initial value of R = 11011101, determine the sequence of binary values in R after a logical shift left, followed by a circular shift right, followed by a logical shift Right and a circular shift left. **(2015),(2016),(2018),(2019)**

Ans. Starting from an initial value of R = 11011101, determine the sequence of binary values in R after each operation in the sequence: (1) a logical shift-left, (2) followed by an arithmetic shift-right, (3) followed by another arithmetic shift-right, and (4) followed, finally, by a circular shift-left. Show all your work.

Left shift (11011101) leads to 10111010

Arithmetic Shift Right (10111010) leads to 11011101

Arithmetic Shift Right (01011101) leads to 11101110

Circular Shift Left (00101110) leads to 11011101

Q.4. Explain what operation will take place when the following instructions are executed LX1 rp, data; LDA addr, LHLD addr & STA addr **(2015)**

Ans. LXI rp, data :- (Load register pair immediate) [rp] <—— data

LDA addr :- (Load Accumulator Direct) [A] <—— [addr]

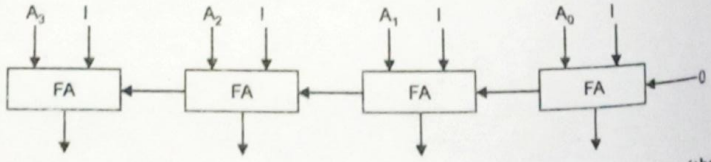
LHLD addr :- (Load H-L Pair Direct) [L] <——[addr]

[H] <——[addr +1]

STA addr :- (Store Accumulator Direct) [addr] <——[A]

Q.5. Design a 4-bit combinational circuit decrementer using four full-adder circuits. **(2015),(2018),(2019)**

Ans. A – 1 = A + 2's complement of 1 = A + 1111



Q.6. Show the step by step multiplication process using booth algorithm. When the binary numbers (+ 15) × (– 13) are multiplied. Assume 5-bit registers that hold signed numbers. The multiplicand is + 15. **(2015)**

Ans.

A		0	1	1	1	1				15	
X	x	1	0	0	1	1				–13	
Y		–1	0	1	0	–1				recoded multiplier	
Add –A	+		1	0	0	0	1				
Shift			1	1	0	0	0	1			
Shift Only			1	1	1	0	0	0	1		
Add A	+		0	1	1	1	1				
			0	1	0	1	1	0	1		
Shift			0	0	1	0	1	1	0	1	
Shift Only			0	0	0	1	0	1	1	0	1
Add –A	+		1	0	0	0	1				
	1		0	0	1	1	1	1	0	1	
Shift	1		1	0	0	1	1	1	1	0	1 –195

Q.7. What are the various registers of 8085? **(2015)**

Ans. 8085 Microprocessor has 8-bit registers: A,B,C,D,E,H,L,F and two 16-bit registers PC and SP. These registers can be classified as :

1. General Purpose Registers.
2. Temporary Registers:
   (a) Temporary Data Registers
   (b) W and Z Registers
3. Special Purpose Registers:
   (a) Accumulator
   (b) Flag Registers
   (c) Instruction Registers
4. Sixteen Bit Registers
   (a) Program Counter (PC)
   (b) Stack Pointer (SP)

(1) General Purpose Registers: B,C,D,E,H and L are 8 bit General Purpose Registers can be used as a separate 8-bit register or as 16-bit register pairs BC, DE and HL. When used in register pair mode, the high order bytes resides in the first register i.e in B and the lower order bytes resides in the second register when BC is used as a register pair.

HL pair is also function as a data pointer or memory pointer. They are also called Scratchpad Register, as user can store data in them. The efficient programmer prefers to use these registers to store intermediate results.

(2) Temporary Registers:

(a) Temporary Data Registers: The ALU has two Inputs, one input is supplied by the accumulator and other from temporary data registers. The programmer cannot access this temporary data registers. However, it is internally used for execution of most of the arithmetic and logic instructions.

(b) W and Z Registers: W and Z register are temporary registers. These registers are used to hold 8 bit data during execution of some instruction. These registers are not available for programmer since 8085 microprocessor uses them internally.

(3) Special Purpose Registers:-

(a) Accumulator (Register A): It is a tri-state 8-bit register. It is extensively used in arithmetic logic, load and store operation, as well as in Input-Output operation. Result of ALU are stored in this register.

(b) Flag Registers: It is 8 bit register in which 5 of the bit carry significant information in the form of flags: S(Sign flag), Z(Zero flag), AC (Auxiliary flag), P(Parity flag) and CY (Carry flag) as shown in fig.

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	X	AC	X	P	X	CY

(Flag register)

(c) Instruction Registers: In a typical processor operation, the processor first fetches the opcode of instruction from memory. The CPU stores this opcode in a register called the instruction register.

(4) Sixteen Bit Registers:-

(a) Program counter (PC): The program counter is a special purpose register which a t a given time stores the address of the next instruction.

(b) Stack Register (SP): The stack is reserved area of the memory in the RAM where temporary information may be stored. A 16-bitnstack pointer is used to hold the address of the most recent stack entry.

Q.8. Explain BCD subtraction using 9's complement for (87 – 39). (2015)

Ans. 9's complement of 39 is 60

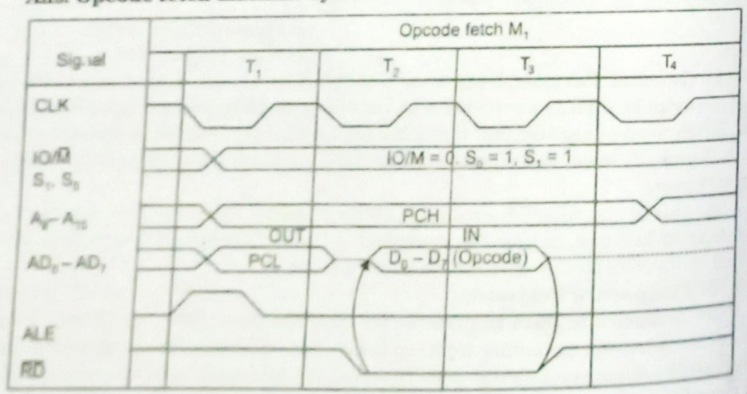
Adding 60 to 87

$$\begin{array}{r} 87 \\ +60 \\ \hline 147 \\ \quad\;\; 1 \\ \hline 48 \end{array}$$

BCD Subtraction using 9's complement of (87-39) is 48

Q.9. Draw and explain the timing diagram for fetch operation ? (2015)

Ans. Opcode fetch machine cycle



The Opcode fetch cycle, fetches the instructions from memory and delivers it to the instruction register of the microprocessor. For any instruction cycle, Opcode fetch is the first machine cycle. We know that each machine cycle may have 3 to 6 T-states. This Opcode fetch machine cycle consists of 4 T-states.

T_1 State: During the T_1 state, the contents of the program counter are placed on the 16 bit address bus. The higher order 8 bits are transferred to address bus (A_8-A_{15}) and lower order 8 bits are transferred to multiplexed A/D (AD_0-AD_7) bus.

After the address bits are transferred, the ALE (address latch enable) signal goes high. As soon as ALE goes high, the memory latches the AD_0-AD_7 bus. At the middle of the T state the ALE goes low and the complete 16-bit address is made available for the Opcode fetch machine cycle.

T_2 State: During the beginning of this state, the RD' signal goes low to enable memory. It is during this state, the selected memory location is placed on D_0-D_7 of the Address/Data multiplexed bus.

T_3 State: In the previous state the Opcode is placed in D_0-D_7 of the A/D bus. In this state of the cycle, the Opcode of the A/D bus is transferred to the instruction register of the microprocessor. Now the RD' goes high after this action and thus disables the memory from A/D bus.

T_4 State: In this state the Opcode which was fetched from the memory is decoded. Thus the cycle completes after 4 T-states.

Q.10. The content of accumulator are 93H and the contents of register C are B7H, add both contents. (2015)

Ans. If accumulator content = 93H and content of register C = B7H, then addition of two will be as:

		CY	D7	D6	D5	D4	D3	D2	D1	D0	
A = 93H	=		1	0	0	1	0	0	1	1	
C = B7	=		1	0	1	1	0	1	1	1	
		1	1	1	1		1	1	1		← Carry
Sum (A) = 1 4AH	=	1	0	1	0	0	1	0	1	0	

Status of the flag register after addition:

Flag Register	D7	D6	D5	D4	D3	D2	D1	D0
	S	Z	–	AC	–	P	–	CY
	0	0		0		0		0

Q.11. Define Microoperation ? (2015, 2017)

Ans. A microoperation is an elementary operation performed with the data stored in registers. The microoperations in digital computers are classified into four categories:-

1. Register transfer microoperations which transfer binary information from one register to another.

2. Arithmetic microoperation which performs arithmetic operations on numeric data stored in registers.

3. Logic microoperations which perform bit manipulation operations on non-numeric data stored in registers.

4. Shift microoperation which perform shift operations on data stored in registers.

Q.12. Define Microinstruction? (2015, 2017)

Ans. Each word in control memory contains within it a microinstruction. The microinstruction specifies one or more microoperations for the system. A sequence of microinstructions constitutes a microprogram.

The Microinstruction code format for the control memory is :

3	3	3	2	2	7
F1	F2	F3	CD	BR	AD

Here,

F1,F2,F3 : Microoperation fields

CD : Condition for Branching

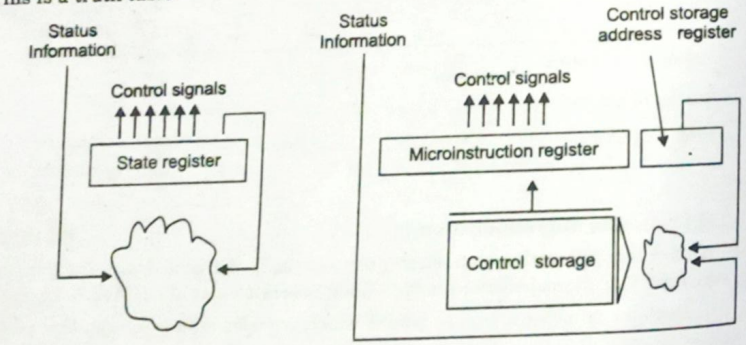
Br : Branch field

AD: Address field

Q.13. Explain the difference between hardwired control and microprogrammed control? (2015),(2016),(2018)

Ans. Micro programmed control: Micro programmed control is a control mechanism to generate control signals by using a memory called control storage (CS), which contains the control signals. Although micro programmed control seems to be advantageous to CISC machines, since CISC requires systematic development of sophisticated control signals, there is no intrinsic difference between these 2 control mechanisms.

Hard-wired control: Hardwired control is a control mechanism to generate control signals by using appropriate finite state machine (FSM). The pair of "microinstruction-register" and "control storage address register" can be regarded as a "state register" for the hardwired control. Note that the control storage can be regarded as a kind of combinational logic circuit. We can assign any 0, 1 values to each output corresponding to each address, which can be regarded as the input for a combinational logic circuit. This is a truth table.



(a) Hardwired control (b) Microprogrammed control

Q.14. An instruction is stored at location 300 with its address field at location 301. The address field has the value 400. A processor register R1 contains the number 200. Evaluate the effective address if the addressing mode of the instruction is:

(i) Direct (ii) Immediate (iii) Relative (iv) Register Indirect (v) Index with R1 as the index register? (2015)

Ans. Location _ Contents

300 _ opcode; the instruction operation code

301 _ 400; address field of the above instruction

(a) Direct addressing: Direct addressing means that the address field contains the address of memory location the instruction is supposed to work with (where an operand "resides").

Effective address would therefore be 400.

(b) Immediate addressing: Immediate addressing means that the address field contains the operand itself.

Effective address would therefore be 301.

(c) Relative addressing: Relative addressing means that the address field contains offset to be added to the program counter to address a memory location of the operand.

Effective address would therefore be 301 + 400 = 701.

(d) Register indirect addressing: Register indirect addressing means that the address of an operand is in the register. The address field in this case contains just another operand.

Effective address would therefore be in R1 = 200.

(e) Indexed addressing with R1 as index register: There are several possible indexed addressing modes but in this case (there is an address field) it is co called "indexed absolute" addressing.

In indexed absolute addressing the effective address is calculated by taking the contents of the address field and adding the contents of the index register.

Effective address would therefore be 400 + R1 = 400 + 200 = 600.

Q.15. What is an instruction format? Explain different types of instruction format? (2015),(2016),(2017)

Ans. It is the function of the control unit within the CPU to interpret each instruction code and provide the necessary control functions needed to process the instruction The bits of the instruction are divided into group called fields. The most common fields found found in instruction formats are:-

1. An operation code field that specifies the operation to be performed.
2. An address field that designated a memory address or a processor register.
3. A mode field that specifies the way the operand or the effective address is determined.

Opcode	Mode	Address

Computers may have instructions of several different lengths containing varying number of addresses. The number of address field in the instruction format of a computer depends on the internal organization of its registers. Most computers fall into one of three types of CPU organization.

(1) Single Accumulator organization ADD X AC ® AC + M [×]

(2) General Register Organization ADD R1, R2, R3 R ® R2 + R3

(3) Stack Organization PUSH X

Three address Instruction: Computer with three addresses instruction format can use each address field to specify either processor register are memory operand.

ADD R1, A, B A1 ® M [A] + M [B]

ADD R2, C, D R2 ® M [C] + M [B] X = (A + B) * (C + A)

MUL X, R1, R2 M [X] R1 * R2

The advantage of the three address formats is that it results in short program when evaluating arithmetic expression. The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

Two Address Instruction: Most common in commercial computers. Each address field can specify either a processes register on a memory word.

MOV	R1, A	R1 ® M [A]	
ADD	R1, B	R1 ® R1 + M [B]	
MOV	R2, C	R2 ® M [C]	X = (A + B) * (C + D)
ADD	R2, D	R2 ® R2 + M [D]	
MUL	R1, R2	R1 ® R1 * R2	
MOV	X1 R1	M [X] ® R1	

One Address instruction: It used an implied accumulator (AC) register for all data manipulation. For multiplication/division, there is a need for a second register.

LOAD	A	AC ® M [A]
ADD	B	AC ® AC + M [B]
STORE	T	M [T] ® AC X = (A +B) × (C + A)

All operations are done between the AC register and a memory operand. It's the address of a temporary memory location required for storing the intermediate result.

LOAD	C	AC ® M (C)
ADD	D	AC ® AC + M (D)
ML	T	AC ® AC + M (T)
STORE	X	M [×]® AC

Zero – Address Instruction: A stack organized computer does not use an address field for the instruction ADD and MUL. The PUSH & POP instruction, however, need an address field to specify the operand that communicates with the stack (TOS ® top of the stack)

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

Q.16. Explain BUN (Branch unconditionally) and BSA (Branch and save return address) instruction with example? (2014)

Ans. BUN: Branch Unconditionally

This instruction has the responsibility to transfer the entire program to the instruction which is specified by the effective address. We now know that program counter (PC) holds the address of the instruction to be read from the memory and program is a set of instructions to be carried out to accomplish the particular task. BUN instruction allows the programmer to specify an instruction out of the program and modify the program.

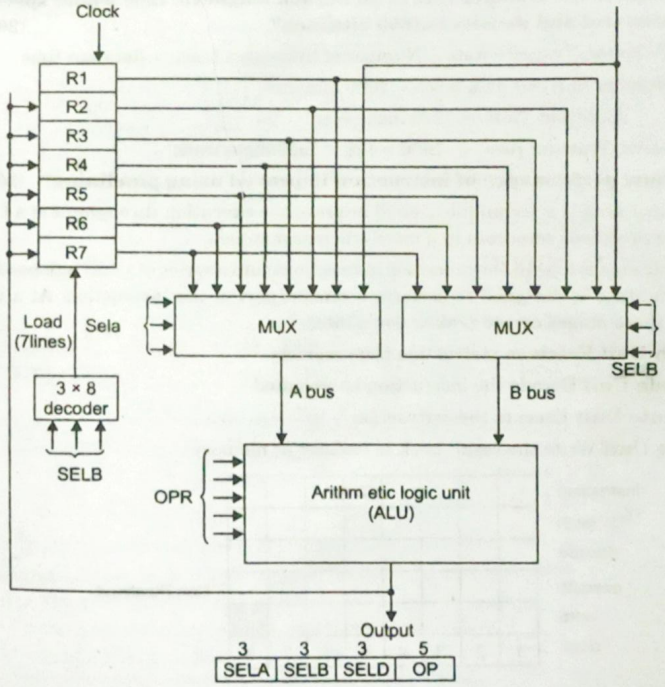
Example: D_4T_4: PC <– AR, SC <– 0.

BSA : Branch and Save Return Address

As the name tells the function of this instruction, it allows the branching in the execution of instruction. By branching we mean that the instructions can have sub routine or procedure. When this instruction is executed, it stores the address of the next instruction to be executed as PC (Program counter).

Q.17. Draw the general register organization of basic computer, having seven registers. Also show all connections with Arithmetic Logical Unit (ALU) and control word? (2015)

Ans.



Control word

Q.18. What is Accumulator? (2015),(2017)

Ans. In a computer's central processing unit (CPU), an accumulator is a register in which intermediate arithmetic and logic results are stored.

Without a register like an accumulator, it would be necessary to write the result of each calculation (addition, multiplication, shift, etc.) to main memory, perhaps only to be read right back again for use in the next operation. Access to main memory is slower than access to a register like the accumulator because the technology used for the large main memory is slower (but cheaper) than that used for a register. Early electronic computer systems were often split into two groups, those with accumulators and those without.

Modern computer systems often have multiple general purpose registers that operate as accumulators, and the term is no longer as common as it once was. However, a number of special-purpose processors still use a single accumulator for their work, in order to simplify their design

Q.19. What is the role of implied mode in addressing mode? (2015),(2018)

Ans. Implied mode: In this mode the operands are specified implicitly in the definition of the instruction. For example, the instruction "complement accumulator" is an implied mode instruction because the operand in the accumulator register is implied in the definition of the instruction.

Q.20. What is the transfer rate of an 8-track magnetic tape whose speed is 120 inches/second and density is 1600 bits/inch? (2015)

Ans. We Know, Transfer Rate = Number of bytes on a track × Rotation time

Here, Number of Bytes on a track = 1600 bits/inch

Rotation Time = 120 inches/sec.

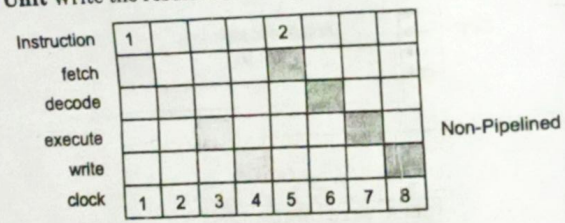
Therefore, Transfer rate = 1600 × 120 = 192000ms/track

Q.21. How performance of instruction improved using pipelining? (2015)

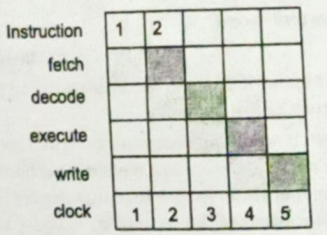
Ans. Pipelining is a technique used to improve the execution throughput of a CPU by using the processor resources in a more efficient manner.

The basic idea is to split the processor instructions into a series of small independent stages. Each stage is designed to perform a certain part of the instruction. At a very basic level, these stages can be broken down into:

- **Fetch Unit Fetch** an instruction from memory
- **Decode Unit** Decode the instruction be executed
- **Execute Unit** Execute the instruction
- **Write Unit** Write the result back to register or memory



Non-Pipelined



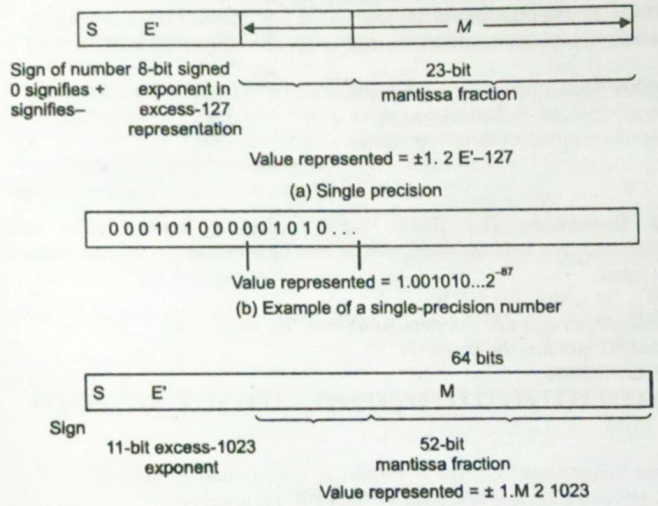
Pipelined

On a non-pipelined CPU, when a instruction is being processed at a particular stage, the other stages are at an idle state – which is very inefficient. If you look at the diagram, when the 1st instruction is being decoded, the Fetch, Execute and Write Units of the CPU are not being used and it takes 8 clock cycles to execute the 2 instructions.

On the other hand, on a pipelined CPU, all the stages work in parallel. When the 1st instruction is being decoded by the Decoder Unit, the 2nd instruction is being fetched by the Fetch Unit. It only takes 5 clock cycles to execute 2 instructions on a pipelined CPU. Increasing the number of stages in the pipeline will not always result in an increase of the execution throughput.

Q.22. Represent floating point in IEEE standard format for given No. $1.001010....0\times2^{-87}$ and explain difference in Single precision and double precision numbers?? (2015),(2019)

Ans.



Single Precision: The IEEE single precision floating point standard representation requires a 32 bit word, which may be represented as numbered from 0 to 31, left to right.

- The first bit is the **sign** bit, S,
- the next eight bits are the **exponent** bits, 'E', and
- the final 23 bits are the **fraction** 'F':

S EEEEEEEE FFFFFFFFFFFFFFFFFFFFFFF

0 1 8 9 31

The value V represented by the word may be determined as follows:

- If E=255 and F is nonzero, then V=NaN ("Not a number")
- If E=255 and F is zero and S is 1, then V=-Infinity
- If E=255 and F is zero and S is 0, then V=Infinity
- If 0<E<255 then V=(-1)**S * 2 ** (E-127) * (1.F) where "1.F" is intended to represent the binary number created by prefixing F with an implicit leading 1 and a binary point.
- If E=0 and F is nonzero, then V=(-1)**S * 2 ** (-126) * (0.F). These are "unnormalized" values.

- If E=0 and F is zero and S is 1, then V=-0
- If E=0 and F is zero and S is 0, then V=0

In particular,

0 00000000 00000000000000000000000 = 0
1 00000000 00000000000000000000000 = -0

0 11111111 00000000000000000000000 = Infinity
1 11111111 00000000000000000000000 = -Infinity

0 11111111 00000100000000000000000 = NaN
1 11111111 00100010001001010101010 = NaN

0 10000000 00000000000000000000000 = +1 * 2**(128-127) * 1.0 = 2
0 10000001 10100000000000000000000 = +1 * 2**(129-127) * 1.101 = 6.5
1 10000001 10100000000000000000000 = -1 * 2**(129-127) * 1.101 = -6.5

0 00000001 00000000000000000000000 = +1 * 2**(1-127) * 1.0 = 2**(-126)
0 00000000 10000000000000000000000 = +1 * 2**(-126) * 0.1 = 2**(-127)
0 00000000 00000000000000000000001 = +1 * 2**(-126) *
0.00000000000000000000001 =
2**(-149) (Smallest positive value)

Double Precision: The IEEE double precision floating point standard representation requires a 64 bit word, which may be represented as numbered from 0 to 63, left to right.

- The first bit is the sign bit, S,
- the next eleven bits are the **exponent** bits, 'E', and
- the final 52 bits are the **fraction** 'F':

S EEEEEEEEEEE
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0 1 11 12
63

The value V represented by the word may be determined as follows:

- If E=2047 and F is nonzero, then V=NaN ("Not a number")
- If E=2047 and F is zero and S is 1, then V=-Infinity
- If E=2047 and F is zero and S is 0, then V=Infinity
- If 0<E<2047 then V=(– 1)**S * 2 ** (E – 1023) * (1.F) where "1.F" is intended to represent the binary number created by prefixing F with an implicit leading 1 and a binary point.
- If E=0 and F is nonzero, then V=(– 1)**S * 2 ** (– 1022) * (0.F) These are "unnormalized" values.
- If E=0 and F is zero and S is 1, then V = – 0
- If E=0 and F is zero and S is 0, then V = 0

Q.23. An 8 bit register contains binary value 10011100. What is register value after arithmetic shift right? Starting from initial no. 10011100 determine register value after an arithmetic shift left and state whether this is overflow? (2015),(2017)

Ans. R = 10011100
Arithmetic shift right: 11001110
Arithmetic shift left: 00111000 overflow because a negative number changed to positive

Q.24. Register A holds the 8-bit binary value 11011001. Determine the B operand and the logic micro-operation to be performed in order to change the value in A to: (2015)

Ans. a. 01101101
b. 11111101

(a)

01101101	(b) 11111101
11011001	11011001
XOR	OR
10110100	11111101
01101101	11111101

Q.25. What are the features of 8085 microprocessor? Explain addressing mode of 8085. (2015), (2017)

Ans. (1) 8085 microprocessor is an 8 bit microprocessor. i.e. it can accept or provide 8 bit data simultaneously.

(2) 8085 microprocessor is a single chip, NMOS device implemented with 6200 transistors.

(3) 8085 microprocessor requires a single +5V DC power supply.

(4) 8085 microprocessor provides on chip clock generator, therefore there is no need of external clock generator, but it requires external tuned circuit like LC, RC or crystal.

(5) 8085 microprocessor requires two phase, 50% duty cycle, TTL clock. These clock signals are generated by an internal clock generator.

(6) The maximum clock frequency of 8085 microprocessor is 3MHz where as minimum clock frequency is 500 KHz.

1. Immediate Addressing Mode: - An immediate is transferred directly to the register.

Eg: MVI A, 30H (30H is copied into the register A)
MVI B,40H(40H is copied into the register B).

2. Register Addressing Mode: - Data is copied from one register to another register.

Eg: - MOV B, A (the content of A is copied into the register B)
MOV A, C (the content of C is copied into the register A).

3. Direct Addressing Mode: - Data is directly copied from the given address to the register.

Eg: - LDA 3000H (The content at the location 3000H is copied to the register A).

4. Indirect Addressing Mode: - The data is transferred from the address pointed by the data in a register to other register.

Eg: - MOV A, M (data is transferred from the memory location pointed by the regiser to the accumulator).

5.Implied Addressing Mode: - This mode doesn't require any operand. The data is specified by opcode itself.

Eg: - RAL
CMP

Q.26. Derive the control gates associated with the programm counter PC in the basic computer. (2015)

Ans. (Assume that ZDR = 1 if DR = 0; ZAC = 1, if AC = 0)

$INR\ (PC) = R'T_1 + RT_2 + D_6T_6ZDR + PB_9(FGI) + PB_8(FGO) + rB_4(AC_{12})' + rB_3(AC_{15})' + rB_2ZAC + rB_1E'$

$LD\ (PC) = D_4T_4 + D_5T_5$

$CLR\ (PC) = RT_1$

Q.27. An output program resides memory starting from address 2300. It is executed after the computer recognizes an interrupt when FGO becomes as 1 (while IEN=1).

(a) What instruction must be placed at address 1?

(b) What must be the last two instructions of the output program? (2015)

Ans. (a) 0 BUN (2300)10

(b) ION

1 BUN 0 (Branch indirect with address 0)

Q.28. Write a program loop, using a pointer and a counter, that clears to 0 the contents of hexadecimal locations 500 through 5FF.

Ans.

```
LDA NBR     /Initialize counter with
CMA         /2's compl. of NBR
INC
STA CTR     /save -NBR to counter
LDA ADR     /Save start address to
STA PTR     /PTR
LOP,CLA     /Clear AC
STA PTR I   /Reset memory word
ISZ PTR     /Increment pointer
ISZ CTR     /increment counter
BUN LOP     /Branch to LOP (CTR < 0)
HLT         /Halt when CTR = 0
NBR,HEX FF  /Nbr of cleared words
CTR,-       /Counter
ADR,HEX 500 /Start address
PTR,-       /Pointer
```

Q.29. Explain the three types of CPU organization with the help of example? (2015)

Ans. Computers may have instructions of several different lengths containing varying number of addresses. The number of address fields in the instruction format of a computer depends on the internal organization of its registers. Most computers fall into one of the three types of CPU organizations:

(i) Single Accumulator Organization: In this type of organization all operations are performed on an implied accumulator. The instruction format uses only one address field. For example, the instruction that loads the accumulator with the contents of a memory location.

LoadX

Where X is the address of the source operand. This results in the operation AC ÷- M (X). AC is the accumulator and M(X) symbolizes the memory word located at address X.

(ii) General Register Organisation : In this organization, the instruction format needs 2 or 3 register address fields according to the operation.

For example, an instruction for addition may be written as

ADD R1, R2, R3,

It denotes the operation R1 <-R2 -f- R3

The same ADD instruction needs only two register address fields if the destination register is one of the source registers, i.e. if the operation is

R1 R1 + R2

Then the instruction is ADD R1, R2

The instruction may also contain one memory address field and one register address field. For example, the instruction,

ADD R1, X

Specifies the operation R1 -R1 + M [X]

(iii) Stack Organization : In this organization, the computers will have PUSH and POP instructions which require an address field. For Kathplê, the instruction PUSH X will push the word at address Xonto the top of the stack. The operation - type instructions do not need any address field. For example, the instruction

ADD

Consists of only opcode and no address field. It has the effect of popping the top two numbers from the stack, adding them, and pushing the sum onto the stack. Thus all the operands are implied to be in the stack.

Most of the computers fall into one of the above three types of organizations. Some computers combine features from more than one organizational structure.

Q.30. Define register transfer language (RTL). (2016)

Ans. Register transfer language (RTL) is a kind of intermediate representation (IR) that is very close to assembly language, such as that which is used in a compiler. It is used to describe data flow at the register-transfer level of an architecture. Micro-operations can be expressed in terms of a Register Transfer Language (RTL).

Q.31. Explain instruction cycle. (2016)

Ans. An instruction cycle (sometimes called a fetch–decode–execute cycle) is the basic operational process of a computer. It is the process by which a computer retrieves a program instruction from its memory, determines what actions the instruction dictates, and carries out those actions. This cycle is repeated continuously by a computer's central processing unit (CPU), from boot-up to when the computer is shut down.

Q.32. Explain the different shift operations. (2016)

Ans. Shift micro operations are the operations in which the contents of the register can be shifted to left or right. Shift micro operations are used for serial transfer of data. They can also be used in lieu with arithmetic, logic and other data-processing operations. There are three types of shift operations:

1. Logical shift: Logical shift can be defined as the shift of the bits to the right or left serially. Let us suppose the symbol for logical right shift as shr and for logical left shift as shl.

Ex: Shift micro operationR1 shl R1

2. Circular shift: Circular shift also named as rotate shift circulates the bits among the ends of the register without losing information. We can achieve this by connecting the output terminal to the input terminal of the register. They also shift only one bit at a single time. For example

Ex: Circular micro operationR cir R

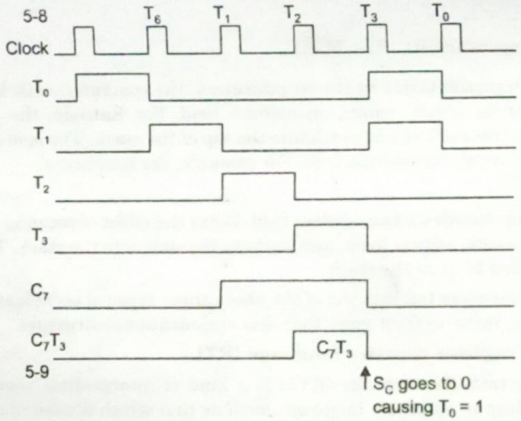
3. Arithmetic Micro-operation: The main purpose of Arithmetic Micro-operation is to perform arithmetic operation on numeric data.

Q.33. Draw & explain a timing diagram assuming that SC is cleared to 0 at time T_3 if control signal C_7 is active

$$C_7T_3 : SC4–0$$

C_7 is activated with the positive clock transition associated with T (2016)

Ans.



Q.34. Explain the three different types of instruction code formats of a basic computer namely-memory reference, register-reference and input/output instruction. (2016),(2017)

Ans. All Basic Computer instruction codes are 16 bits wide. There are 3 instruction code formats:

Memory-reference instructions take a single memory address as an operand, and have the format:

```
15 14  12 11     0
+----------------+
| I | OP  | Address |
+----------------+
```

Here 12-bits are used for storing the address of the operands. Three bits are used for operation code and one bit is used for addressing mode. The opcode has the binary value from 000 to 110. The addressing mode I has two values 0 and 1. example: BUN- Branch Unconditionally

Register-reference instructions operate solely on the AC register, and have the following format :

```
15  14 12 11    0
+---------------+
| 0 |111 | OP    |
+---------------+
```

These Instructions are Recognised by opcode 111 with addressing mode 0.
example: INC- Increment AC

Input/output instructions have the following format:

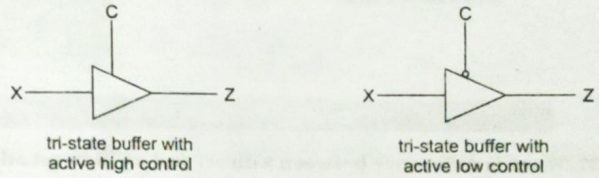
```
15  14 12 11    0
+---------------+
| 0 |111 | OP    |
+---------------+
```

In the Input/Output Instruction, The MSB is 1 . The op-code bits is 111 and the remaining bits are used to store the other address. Example: INP- Input character to AC

Q.35. Explain three-state bus buffer with diagram. (2016)

Ans. A tri-state buffer is a useful device that allows us to control when current passes through the device, and when it doesn't.

Here's two diagrams of the tri-state buffer.



A tri-state buffer has two inputs: a data input x and a control input c. The control input acts like a valve. When the control input is active, the output is the input. That is, it behaves just like a normal buffer. The "valve" is open.

When the control input is not active, the output is "Z". The "valve" is open, and no electrical current flows through. Thus, even if x is 0 or 1, that value does not flow through.

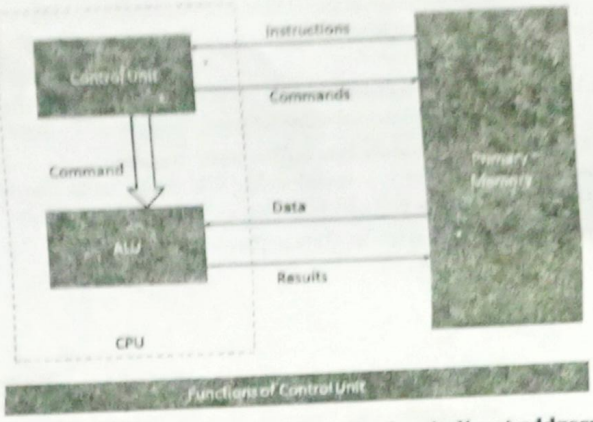
Here's a truth table describing the behavior of a active-high tri-state buffer.

c	x	z
0	0	Z
0	1	Z
1	0	0
1	1	1

In this case, when the output is Z, that means it's high impedance, neither 0, nor 1, i.e., no current.

Q.36. Draw & explain control unit of basic computer. (2016)

Ans. The control unit (CU) is a component of a computer's central processing unit (CPU) that directs the operation of the processor. It tells the computer's memory, arithmetic/logic unit and input and output devices how to respond to a program's instructions. It directs the operation of the other units by providing timing and control signals. Most computer resources are managed by the CU. It directs the flow of data between the CPU and the other devices. John von Neumann included the control unit as part of the von Neumann architecture. Thus control unit controls the complete work-flow of the CPU. But control unit doesn't take inputs, give outputs, process data or store data itself, what control unit do is, it controls these operations when they are performed by respective devices. The purpose of control unit to run the whole computer. And control unit is ran by the instructions stored in RAM and ROM. So, control unit receives instructions which are stored in RAM and ROM and controls operations of other connected units or devices through those instructions.



Q.37. What is difference between a direct and an indirect address. Explain with example. **(2016)**

Ans. Direct addressing means the instruction refers directly to the address being accessed. That is, the instruction encoding itself contains the address of the location. Depending on the instruction set, it may also allow computing a small index relative to the address.direct addressing has the address of a memory location as the operand in an instruction.

Ex:

8086:

MOV AX, [01234h] ; direct. Loads loc DS:01234h into AX

MOV AX, [01234h+BX]; direct-indexed. Loads loc DS:(01234h+BX) into AX

Indirect addressing uses an address held in a register or other location to determine what memory location to read or write. The idea here is that the instruction itself isn't directly telling you the address to access, but rather indirectly telling the CPU where to find that address. The processor may also allow you to add a small offset to the indirect address, giving an indirect-indexed addressing mode indirect addressing has a pointer to the memory location as the operand in an instruction.

Ex:

8086:

MOV AX, [BX] ; Indirect via BX. Loads DS:BX into AX

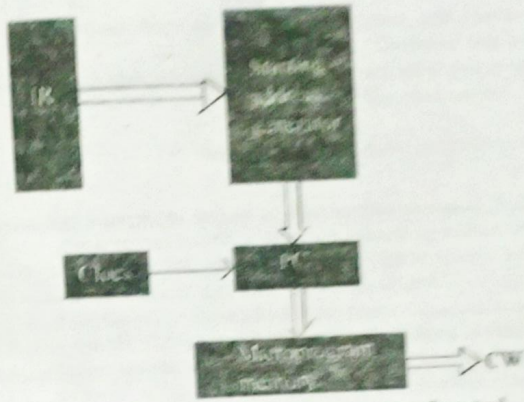
MOV AX, [01234h + BX] ; Indirect-indexed. Loads DS:(01234h + BX) into AX

Q.38. What is control word? **(2016)**

Ans. There are 14 binary selection inputs in the units, and their combined value specified a control word. It consists of four fields three fields contain three bits each, and one field has five bits. The three bits of SEL A select a source register for the A input of the ALU. The three bits of SEL B select a source register for the B input of the ALU. The three bit of SEC D select a destination register using the decoder and its seven load outputs. The five bits of OPR select one of the operations in the ALU. The 14-bit control word when applied to the selection inputs specify a particular micro-operation.

Q.39. Draw block diagram of general configuration of a microprogrammed control unit. **(2016)**

Ans.



Basic organization of a microprogrammed control

Q.40. Evaluate the following arithmetic expressions using 2-address, 1-address and 0-address instructions to show the effect on computer program. X = (A+B) * < (C – D) , **(2016)**

Ans. 2 address :

MOV R1,A
ADD R1,B
MOV R2,C
SUB R2,D
MUL R1,R2
MOV X,R1

1 address:

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

0 address:

PUSH A
PUSH B
ADD
PUSH C
PUSH D
SUB
MUL
POP X

Q.41. The control memory has 4096 words of 24 bits each.

(i) How many bits are there in each of the four inputs going into the multiplexers?

(ii) What are the number of inputs in each multiplexer and how many multiplexers are needed?

(iii) How many bits are there in the control address register? **(2016)**

Ans. Control memory = 2^12 * 24

(i) 12 bits

(ii) 12 multiplexers, each of size 4-to-1 line

(iii) 12 bits

Q.42. What is basic advantage of using interrupt-initiated data transfer over transfer under program control without interrupt **(2016)**

Ans. In the interrupt initiated data transfer, the processor verifies the request and transfer the control ISR to perform the task and its resumes back with the useful task while, the processor has to waste its time by performing all the task, for example when a print command is given in the interrupt initiated , it gives control over to ISR and resumes the work back where as without interrupt the processor has to wait unless the print document is transferred to the printer.

It does not require the continuous checking of I/O port by the computer . The computer checks the port only when an I/O interrupt is encountered . It saves processing time. The CPU issues commands to the I/O module then proceeds with its normal work until interrupted by I/O device on completion of its work.

For input, the device interrupts the CPU when new data has arrived and is ready to be retrieved by the system processor. The actual actions to perform depend on whether the device uses I/O ports, memory mapping.

For output, the device delivers an interrupt either when it is ready to accept new data or to acknowledge a successful data transfer. Memory-mapped and DMA-capable devices usually generate interrupts to tell the system they are done with the buffer.

Although Interrupt relieves the CPU of having to wait for the devices, but it is still inefficient in data transfer of large amount because the CPU has to transfer the data word by word between I/O module and memory. Below are the basic operations of Interrupt:

- CPU issues read command
- I/O module gets data from peripheral whilst CPU does other work
- I/O module interrupts CPU
- CPU requests data
- I/O module transfers data

Q.43. A computer uses a memory unit with 256K words of 32 bits each. A binary instruction code is stored in one word of memory. The instruction has four parts: an indirect bit, an operation code, a register code part to specify one of 64 registers, and an address part.

(i) How many bits are there in the operation code, the register code part and the address part?

(ii) Draw the instruction word format and indicate the number of bits in each part.

(iii) How many bits are there in the data and address inputs of the memory. **(2016)**

Ans.

$256K = 2^3 \times 2^{10} = 218$

$64 = 2^6$

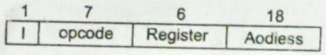
(a)Address : 18 bits $64 = 2^6$

Register code : 6 bits

Indirect bit : $\frac{1}{25}$ bits

$32 - 25 = 7$ sits for opcode,

(b)



= 32 bits

(c) Data : 32 bits ; address: 18 bits.

Q.44. Differentiate between Micro-operation and Macro operation.

Ans.

Micro Operation	Macro Operation
Micro-operations (also known as a micro-ops or μops) are detailed low-level instructions used in some designs to implement complex machine instructions	A Macro instruction is a line of computer program coding that results in one or more lines of program coding in the target programming language, sets variables for use by other statements
Micro-operations perform basic operations on data stored in one or more registers, including transferring data between registers or between registers and external buses of the central processing unit (CPU), and performing arithmetic or logical operations on registers.	In a typical fetch-decode-execute cycle, each step of a macro-instruction is decomposed during its execution so the CPU determines and steps through a series of micro-operations.
The execution of micro-operations is performed under control of the CPU's control unit, which decides on their execution	The use of macro instructions was initiated for two main purposes: to reduce the amount of program coding that had to be written by generating several assembly language statements from one macro instruction and to enforce program writing standards
The micro-operations in computers are classified into the following categories: Register transfer micro-operations, Arithmetic micro-operations, Logic micro-operations and Shift micro operations Example: R3 ← R1 + R2 R3 ← R1 + (1's complement of R2)	A macro-operation is usually divided into • operation code, operand address, addressing mode, etc. • basic addressing modes Immediate, Direct, Indirect Example: for the EXIT macro instruction, a list of define constant instructions and peripheral access by access methods including macros such as • PEN, CLOSE, READ and WRITE

Q.45. The 8-bit registers A, B, C and D are loaded with the value $(F2)_H$, $(B9)_H$ and $(EA)_H$ respectively. Determine the register content after the execution of the following sequence of micro-operations sequentially. Where Shl=shift left.shr=shift right and cir=circular. **(2016)**

(i) A ← A + B, C ← C + shl(d **(ii) C ← C + D, B ← B + 1**

(iii) A ← A-C . **(iv) A ← shr(B) ⊕ cir(d)**

Ans. A = 11110010
B = 10111001
C = 11101010
D = 10010101 (assumed value)
(i) A ← A+B = 1-CARRY 10101011
C ← C+SHL(D) = 11101010 + 00101010 = 1-CARRY 00010100
(ii) C ← C+D = 1-CARRY 01101001 , B ← B+1= 10111010
(iii) A ← A-C = 00001000
(iv) A ← SHR(B) exor cir(D)
= 01011100 EXOR 00101011= 01110111

Q.46. Explain 8085 instruction set architecture and its organization in detail. **(2016)**

Ans. Instruction sets are instruction codes to perform some task. It is classified into five categories.

CONTROL INSTRUCTIONS:

NOP - No operation is performed, i.e., the instruction is fetched and decoded.

HLT - Halt and enter wait state. The CPU finishes executing the current instruction and stops further execution. An interrupt or reset is necessary to exit from the halt state.

LOGICAL INSTRUCTIONS:

CMP - Compare the register or memory with the accumulator

CPI - Compare immediate with the accumulator

ARITHMETIC INSTRUCTIONS:

ADD - Add register or memory, to the accumulator

SUB - Subtract the register or the memory from the accumulator

DATA TRANSFER INSTRUCTIONS:

MOV - Copy from the source (Sc) to the destination(Dt)

LDA - Load the accumulator

8085 consists of the following functional units

Accumulator: It is an 8-bit register used to perform arithmetic, logical, I/O & LOAD/STORE operations. It is connected to internal data bus & ALU.

Arithmetic and logic unit: As the name suggests, it performs arithmetic and logical operations like Addition, Subtraction, AND, OR, etc. on 8-bit data.

General purpose register: There are 6 general purpose registers in 8085 processor, i.e. B, C, D, E, H & L. Each register can hold 8-bit data.

Program counter: It is a 16-bit register used to store the memory address location of the next instruction to be executed.

Stack pointer: It is also a 16-bit register works like stack, which incremented/decremented by 2 during push & pop operations.

Temporary registers: It is an 8-bit register, which holds the temporary data of arithmetic and logical operations.

Flag register: It is an 8-bit register having five 1-bit flip-flops, which holds either 0 or 1 depending upon the result stored in the accumulator.

It has the following configuration:

-8-bit data bus

-16-bit address bus, which can address upto 64KB

-A 16-bit program counter

- 16-bit stack pointer

-Six 8-bit registers arranged in pairs: BC, DE, HL

Q.47. Discuss the procedure to implements a simple CPU. **(2016)**

Ans. Hardwired into a CPU's circuitry is a set of basic operations it can perform, called an instruction set. Such operations may involve, for example, adding or subtracting two numbers, comparing two numbers, or jumping to a different part of a program. Each basic operation is represented by a particular combination of bits, known as the machine language opcode; while executing instructions in a machine language program, the CPU decides which operation to perform by "decoding" the opcode. In general, a CPU executes an instruction by fetching it from memory, using its ALU to perform an operation, and then storing the result to memory. Beside the instructions for integer mathematics and logic operations, various other machine instructions exist, such as those for loading data from memory and storing it back, branching operations, and mathematical operations on floating-point numbers performed by the CPU's floating-point unit (FPU).

Control unit: The control unit of the CPU contains circuitry that uses electrical signals to direct the entire computer system to carry out stored program instructions. The control unit does not execute program instructions; rather, it directs other parts of the system to do so. The control unit communicates with both the ALU and memory.

Arithmetic logic unit: The arithmetic logic unit (ALU) is a digital circuit within the processor that performs integer arithmetic and bitwise logic operations. The inputs to the ALU are the data words to be operated on (called operands), status information from previous operations, and a code from the control unit indicating which operation to perform.

Memory management unit: Most high-end microprocessors (in desktop, laptop, server computers) have a memory management unit, translating logical addresses into physical RAM addresses, providing memory protection and paging abilities, useful for virtual memory. Simpler processors, especially microcontrollers usually don't include an MMU.

Q.48. Bus Architecture and Bus Arbitration? **(2016)**

Ans. Bus Architecture: Bus is a group of wires that connects different components of the computer. It is used for transmitting data, control signal and memory address from one component to another. A bus can be 8 bit, 16 bit, 32 bit and 64 bit. A 32 bit bus can transmit 32 bit information at a time. A bus can be internal or external.

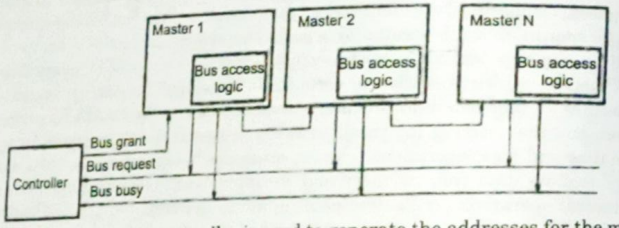
Types of bus:

- **Data bus:** Data bus carries data from on component to another. It is uni-directional for input and output devices and bi-directional for memory and CPU.
- **Control bus:** Control bus carries control signal. CU of CPU uses control signal for controlling all the components. It is uni-directional from CPU to all other components.
- **Address bus:** Address bus carries memory address. A memory address is a numerical value used for identifying a memory location. Computer performs all its task through the memory address. CU of CPU sends memory address to all the components. So, address bus is also uni-directional from CPU to all other components.

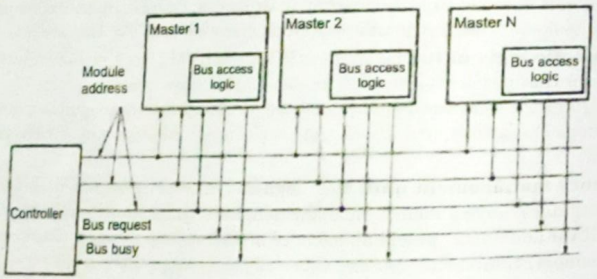
Bus Arbitration: Bus arbitration is the process by which the next device to become the bus master is selected and bus mastership is transferred to it. The selection of bus master is usually done on the priority basis. There are two approaches to bus arbitration: Centralized and distributed.

1. CENTRALIZED ARBITRATION: In centralized bus arbitration, a single bus arbiter performs the required arbitration. The bus arbiter may be the processor or a separate controller connected to the bus.

(a) Daisy chaining: It is simple and cheaper method. All masters make use of the same line for bus request. In response to the bus request the controller sends a bus grant if the bus is free.



(b) Polling: In this the controller is used to generate the addresses for the master. Number of address line required depends on the number of master connected in the system.



2. DISTRIBUTED ARBITRATION: In distributed arbitration, all devices participate in the selection of the next bus master. In this scheme each device on the bus is assigned a 4-bit identification number. The number of devices connected on the bus when one or more devices request for the control of bus, they assert the start-arbitration signal and place their 4-bit ID numbers on arbitration lines, ARB0 through ARB3.

Q.49. What are the advantages of RTL? **(2017)**

Ans. In computer science, register transfer language (RTL) is a kind of intermediate representation (IR) at the register-transfer level of an architecture. Academic papers and textbooks often use a form of that is very close to assembly language, such as that which is used in a compiler. It is used to describe data flow RTL as an architecture-neutral assembly language.

Q.50. Register A holds the 8-bits binary 11011001. Determine the B operand and the logic micro-operation to be performed in order to change the value in A to: **(2017),(2019)**

(i) 01101101 **(ii) 11111101**

Ans. (a) selective complement A= 11011001 B=10110100

(b) selective set A= 11011001 B= 11111101

Q.51. Derive the control gates associated with program counter PC in basic computer. **(2017)**

Ans. The inputs to this circuit come from the two decoders, the I flip-flop, and bits 0-11 of IR. The other inputs are bits 0-15 of AC to check if AC = 0 and to detect the sign bit AC(15); bits 0-15 of DR to check if DR = 0; and the values of seven flip-flops.

- The outputs of the control logic circuit are:

1. Signals to control the inputs of the nine registers
2. Signals to control the read and write inputs of memory
3. Signals to set, clear, or complement the F/Fs
4. Signals for S2 S1 S0 to select a register for the bus
5. Signals to control the AC adder and logic circuit

Control of Registers and Memory:

- Suppose that we want to derive the gate structure associated with the control inputs of AR. We scan to find all the statements that change the content of AR:

$$R'T_0: AR \leftarrow PC$$
$$RT_2: AR \leftarrow IR(0\text{-}11)$$

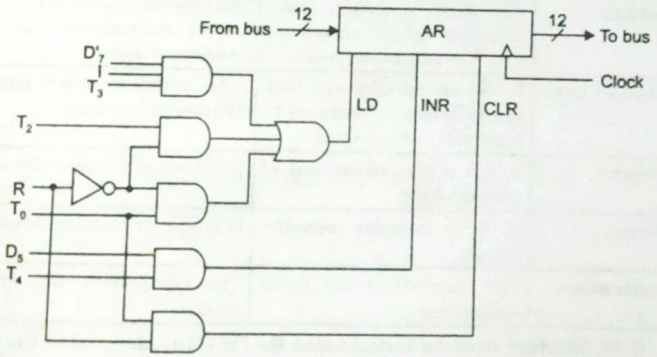


Fig. Control gates associated with AR.

$$D_7'IT_3: AR \leftarrow M[AR]$$
$$RT_0: AR \leftarrow 0$$
$$D_5T_4: AR \leftarrow AR + 1$$

- The control functions can be combined into three Boolean expressions as follows:

$$LD(AR) = R'T_0 + R'T_2 + D_7'IT_3$$
$$CLR(AR) = RT_0$$
$$INR(AR) = D_5T_4$$

The control logic gate associated with AR is shown in Fig.

Q.52. Differentiate between direct and indirect instruction. **(2015),(2017),(2018),(2019)**

Ans.

Direct Address	Indirect Address
If the Address part has the address of an operand, then the instruction is said to have a direct address.	If the address bits of the instruction code are used as an actual operand, it is termed as indirect addressing.
Two memory references are needed to access data	Three memory references are made to access the actual data
Limited address space	Large address space
No additional calculation to workout effective address	Effective address = Address (Address Operand)
Faster memory access	Usually slower than direct addressing mode.

Q.53. Differentiate between RISC and CISC. **(2017)**

Ans.

	RISC	CISC
Acronym	It stands for 'Reduced Instruction Set Computer'.	It stands for 'Complex instruction Set Computer'.
Definition	The RISC processors have a smaller set of instructions with few addressing nodes.	The CISC processors have a larger set of instructions with many addressing nodes.
Memory unit	It has no memory unit and uses a separate hardware to implement instructions	It has a memory unit to implement complex instructions.
Program	It has a hard-wired unit of programming	It has a micro-programming unit.
Design	It is a complex complier design.	It is an easy complier design
Calculations	The calculations are faster and precise	The calculations are slow and precise

Q.54. Starting from an initial value R= 11011101, determine the sequence of binary values in R after a logical shift left, followed by a circular shift right, followed by a logical shift right and a circular shift. **(2017)**

Ans. Operation 1 1 0 1 1 1 0 1

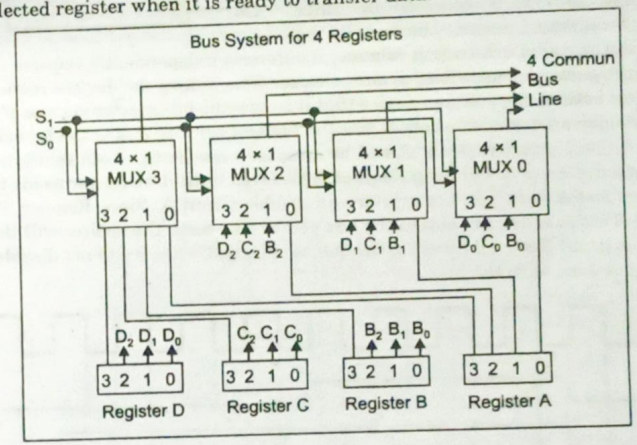
Logical shift left	1	0	1	1	1	0	1	0
Circular shift right	0	1	0	1	1	1	0	1
logical shift right	0	0	1	0	1	1	1	0
Circular shift left	0	1	0	1	1	1	0	0

So final answer would be 01011100

Q.55. Draw an explain bus system for four registers. Also represent the conditional statement by two register transfer statements with control functions. **(2017)**

- **If (P = 1) then (R –> R2) else if (Q = 1) then (R1 <– R3)**

Ans. A bus is a set of common wires that carries data between registers. – There is a separate wire for every bit in the registers. – There are also a set of control signals which determines which register is selected by the bus at a particular time. A bus can be constructed using multiplexer which enable a sets of registers to share a common bus for data transfer. The select lines S1 and S0 indicate which of four register will have its contents transferred to the bus. In general, a bus system will multiplex k registers of n bit each to produce a n–line common bus. It will require n k x 1 multiplexers. The bus is connected to the inputs of all destination registers. and will activate the load control of the selected register when it is ready to transfer data.



Statement

P : R <– R2 , Q: R1 <– R3

Q.56. Convert the decimal 61.5867 into its binary equivalent. **(2017)**

Ans. Conversion steps:

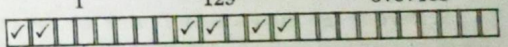
1. Divide the number by 2.
2. Get the integer quotient for the next iteration.
3. Get the remainder for the hex digit.
4. Repeat the steps until the quotient is equal to 0.

$(61.587)_{10} = (111101.1001011001000101101)_2$

Q.57. Convert – 6.75 (written in decimal) to floating point representation (single precision). **(2017)**

Ans.

	sign	Exponent	Mantissa
Value	–1	2^2	1.6875
Encoded as:	1	129	5767168

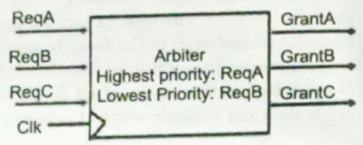


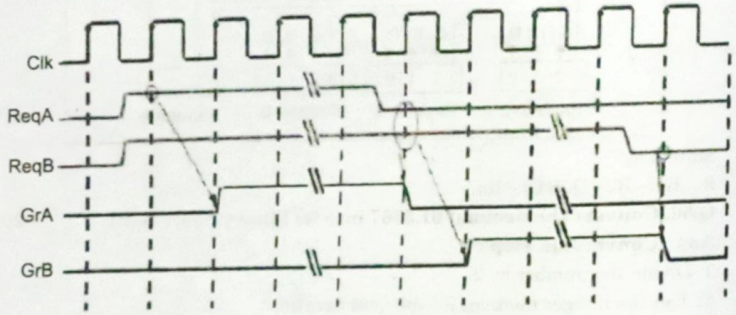
Binary:

You entered	–6.75
Value actually stored in float:	–6.75
Error due to conversion:	0.00
Binary Representation	11000000110110000000000000000000
Hexadecimal Representation	0xc 0d80000

Q.58. Explain serial and parallel bus arbitration in detail. **(2017)**

Ans. In serial bus, if there are multiple masters, then multiple master decides whether it acquires the bus or not using data pattern. In case where master needs to drive 0, it would drive other wise it would release. Master would always figure out whether it has acquired the bus or not by checking the bus being equal to 1 when it is not driving. If it is driven as 0, then it would understand that it is occupied by other master. So it would release the bus. In the centralized, parallel arbitration scheme, the devices independently request the bus by using multipliers. A centralized arbiter chooses from among the devices requesting bus access and notifies the selected device that it is now the bus master via one of the grant line. Consider an example where A has the highest priority and C the lowest. Since A has the highest priority, Grant A will be asserted even though both requests A and B are asserted. Device A will keep Request A asserted until it no longer needs the bus so when Request A goes low, the arbiter will disable Grant A. Since Request B remains asserted (Device B has not gotten the bus yet) at this time, the arbiter will then assert Grant B to grant Device B access to the bus. Similarly, Device B will not disable Request B until it is done with the bus.





Q.59. Explain instruction cycle(fetch) and (decode) in detail. Also explain the working of computer registers used in it. **(2017)**

Ans. Each computer's CPU can have different cycles based on different instruction sets, but will be similar to the following cycle:

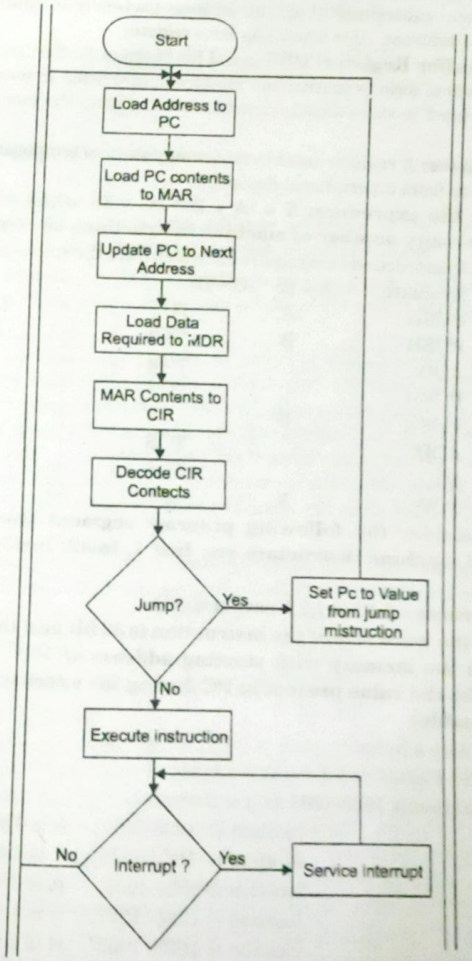
1. Fetch the instruction: The next instruction is fetched from the memory address that is currently stored in the program counter (PC), and stored in the instruction register (IR). At the end of the fetch operation, the PC points to the next instruction that will be read at the next cycle.

2. Decode the instruction: During this cycle the encoded instruction present in the IR (instruction register) is interpreted by the decoder.

3. Read the effective address: In case of a memory instruction (direct or indirect) the execution phase will be in the next clock pulse. If the instruction has an indirect address, the effective address is read from main memory, and any required data is fetched from main memory to be processed and then placed into data registers (Clock Pulse: T_3). If the instruction is direct, nothing is done at this clock pulses. If this is an

I/O instruction or a Register instruction, the operation is performed (executed) at clock Pulse.

4. Execute the instruction: The control unit of the CPU passes the decoded information as a sequence of control signals to the relevant function units of the CPU to perform the actions required by the instruction such as reading values from registers, passing them to the ALU to perform mathematical or logic functions on them, and writing the result back to a register. If the ALU is involved, it sends a condition signal back to the CU. The result generated by the operation is stored in the main memory, or sent to an output device. Based on the condition of any feedback from the ALU, Program Counter may be updated to a different address from which the next instruction will be fetched.



Instruction cycle

The cycle is then repeated. The various registers used are :

MAR stand for Memory Address Register: This register holds the memory addresses of data and instructions. This register is used to access data and instructions from memory during the execution phase of an instruction.

Program Counter: The program counter (PC), commonly called the instruction pointer (IP) in Intel x86 microprocessors, and sometimes called the instruction address register, or just part of the instruction sequencer in some computers, is a processor register

Accumulator Register: This Register is used for storing the Results those are produced by the System. When the CPU will generate

Index Register: A hardware element which holds a number that can be added to (or, in some cases, subtracted from) the address portion of a computer instruction to form an effective address. Also known as base register

Memory Buffer Register: MBR stand for Memory Buffer Register. This register holds the contents of data or instruction read from, or written in memory. It means that this register is used to store data/instruction coming from the memory or going to the memory.

Data Register: A register used in microcomputers to temporarily store data being transmitted to or from a peripheral device.

Q.60. For the expression X = (A + B)* (C + D) when evaluated on stack machine how many number of machine instructions an required. (2017)

Ans. Total 8 instructions are required of which 5 are the stack operation instructions.

•	Evaluate X = (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

Q.61. Consider the following program segment used to execute on a hypothetical machine instructors are: Inst 1, Inst2, Inst3, Inst3, Inst5, Inst6, Inst7

Size(in words): 2,1,1,2,1,2,1 respectively.

Assume the word size of the instruction is 32 bit and the program has been loaded into the memory with starting address of 1000 (Decimal onwards). What could be the value present in PC during the execution of Inst6. (Memory byte addressable). (2017)

Ans. 32 bits= 4 bytes

So 1 word= 4 bytes and 2 words = 8 bytes

instr1- location is 1000-1007 as it is 2 words

instr2-	location is 1008-1011,	it is 1 word
instr3-	location is 1012-1015,	it is 1 word
instr4-	location is 1016-1023,	it is 2 words
instr 5 -	location is 1024- 1027,	it is 1 word
instr 6-	location is 1028- 1035 ,	it is 2 words

therefore, the location is 1036

Q.62. Consider a hypothetical processor which supports expand opcode technique. A 32 bit instruction is place in 256MW memory. If there exist 10, one address instruction then how many zero-address instruction are possible. (2017)

Ans.

8	12	12	= 32 Bits	Two address instructions
Upcode	Address 1	Address 2		

$2^8 = 256$ combinations.

256–250 = 6 combinations can be used for one address

Upcode	Address	One address Instructions

6×2^{12}

Maximum number of one address instruction:

$$= 6 \times 2^{17} = 24,576$$

Q.63. What are the advantages of byte addressing mechanism over word addressing mechanism and what are their disadvantages? (2017)

Ans. Byte addressing refers to hardware architectures which support accessing individual bytes of data rather than only larger units called words, which would be word-addressable.

The basic unit of digital storage is called a bit. In most common computer architectures, 8 bits are grouped together to form a byte. Byte addressable memory refers to architectures where data can be accessed 8 bits at a time. In computer architecture, a word is an ordered set of bytes or bits that is the normal unit in which information may be stored, transmitted, or operated on within a given computer. If a computer's memory is word-addressable then each word in memory is assigned its own memory address. That means that the processor is able to address and fetch only complete words from the memory.

As a simple example, consider a computer with a 16–bit address space. The machine would have 65,536 (64K = 216) addressable entities. The maximum memory size would depend on the size of the addressable entity.

Byte Addressable	64 KB
16-bit Word Addressable	128 KB
32-bit Word Addressable	256 KB

For a given address space, the maximum memory size is greater for the larger addressable entities. This may be an advantage for certain applications, although this advantage is reduced by the very large address spaces we now have: 32–bits now and 64–bits soon. The advantages of byte-addressability are clear when we consider applications that process data one byte at a time. Access of a single byte in a byte-addressable system requires only the issuing of a single address. In a 16–bit word addressable system, it is necessary first to compute the address of the word containing the byte, fetch that word, and then extract the byte from the two-byte word. Although the processes for byte extraction are well understood, they are less efficient than directly accessing the byte. For this reason, many modern machines are byte addressable.

Q.64. Apply selective clear and masking operation on contents of registers A = 1010, B = 1100. **(2018)**

Ans. Selective Clear Selective-clear clears to 0 the bits in register A where there is a corresponding 1 in register B:

1010 Content of A before
1100 Content of B (logic operand)
0010 Content of A after

• This is done using the logical-AND Operation and B'.

Mask: Mask clears to 0 the bits in register A where there is a corresponding 0 in register B:

1010 Content of A before
1100 Content of B (logic operand)
1000 Content of A after

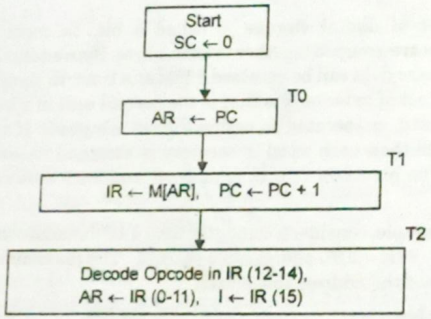
Q.65. What is an effective address? **(2018)**

Ans. Effective address is the address on which the operand is found.

Q.66. Explain fetch and decode operation in instruction cycle. **(2018)**

Ans. Fetch the instruction: The next instruction is fetched from the memory address that is currently stored in the program counter and stored into the instruction register. At the end of the fetch operation, the PC points to the next instruction that will be read at the next cycle.

Decode the instruction: During this cycle the encoded instruction present in the instruction register is interpreted by the decoder.



Q.67. Explain various addressing modes with a numerical example of your choice. **(2018)**

Ans. Addressing modes are nothing but the different ways in which the location of an operand can be specified in an instruction. The number of addressing modes that a processor supports changes according to the instruction set it is based on, however there are a few generic ones that are present in almost all processors and are thus of utmost importance. They are as follows:

i. Immediate mode
ii. Register mode
iii. Absolute mode
iv. Indirect mode
v. Index mode
vi. Base with index
vii. Base with index and offset
viii. Relative
ix. Auto increment
x. Auto decrement

Immediate mode: In this mode, the operand is specified in the instruction itself.

E.g. Move #200,R0

The above instruction places the value 200 in the register R0.Clearly Immediate mode can be used only to specify the source operand.

Figure 1 shows the above concept i.e. the operand is a part of the instruction

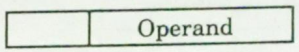


Figure 1: Immediate mode

Register mode: The operand is the contents of a register. We specify the operand in this case by specifying the name of the register in the instruction. Processor registers are often used for intermediate storage during arithmetic operations. This addressing mode is used at that time to access the registers.

E.g. Move R0, R1

Contents of the R0 register are moved to R1 register.

As shown in Figure 2, the instruction names the register that holds the operand.

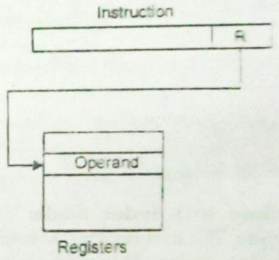


Fig. 2 Register mode

Absolute mode: The operand is in a memory location; the address of the operand is passed explicitly in the instruction. Global variables are represented using this addressing mode.

E.g. Move LOC, R0

Here LOC corresponds to the address from where the contents will be accessed by the processor and placed in R0.

Indirect mode: The effective address (E.A.) of the operands is the contents of a register (see Figure 3(b)) or the memory location whose address appears in the instruction (see Figure 3(a)). The name of the register or the memory address is placed in parentheses to denote indirection or in other words that the contents are addresses of the operands.

E.g. Add (R1), R0 (this mode is often called as register indirect mode) Add (B), R0

This instruction fetches the operand from the address, pointed by the contents of the register R1 or of the memory location 'B' and adds them to R0

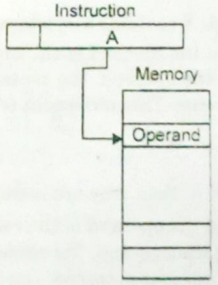


Fig. 3(a)

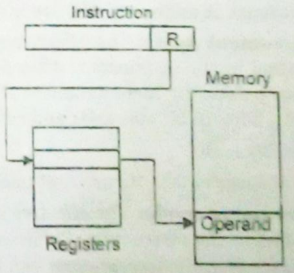


Fig. 3(b)

Index mode: The effective address of the operand is calculated by adding a constant value to the contents of a register, which is clearly shown in Figure 4. The address can be in a register used specially for this purpose or any of the general purpose registers. In either case it is called as an index register.

E.g. Move X (R0), R1

Here, Contents at address X+R0 are moved to R1 .X contains a constant value.

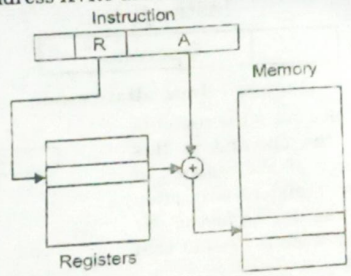


Fig. 4 Index Mode

Base with index mode: The effective address is the sum of contents of two registers. The first register as before is called the index and the second register is called the base register. This mode provides more flexibility since both the components are registers and can thus be changed.

E.g. Move (R0, R1), R2

Here, Contents at address R0+R1 are moved to R2.

Base with index and offset mode: The effective address is the sum of contents of two registers and a constant. The constant value in this case is often called the offset or the displacement.

E.g. Move X (R0, R1), R2

Contents at address X+R0+R1 are moved to R2.

Relative mode: For relative addressing, also called PC-relative addressing, the implicitly referenced register is the program counter (PC). That is, the next instruction address is added to the address field to produce the EA. typically, the address field is treated as a two's complement number for this operation. Thus, the effective address is a displacement relative to the address of the instruction as shown in the example below.

Relative addressing exploits the concept of locality.

E.g. Move X (PC), R1

Here, Contents at address X+PC are moved to R1 .X contains a constant value.

Auto increment mode: The effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are automatically incremented to the next value. This increment is 1 for byte sized operands, 2 for 16 bit operands and so on.

E.g. Add (R2) +, R0

Here are the contents of R2 are first used as an E.A. then they are incremented.

Auto decrement mode: The effective address of the operand is the contents of a register specified in the instruction. Before accessing the operand, the contents of this register are automatically decremented and then the value is accessed.

E.g. Add - (R2), R0

Here are the contents of R2 are first decremented and then used as an E.A. for the operand which is added to the contents of R0. The auto increment addressing mode and the auto decrement addressing mode are widely used for the implementation of data structures like Stack.

Q.68. What are Micro-operations? Explain different types of micro-operations in brief? (2016),(2018),(2019)

Ans. Micro-operations perform basic operations on data stored in one or more registers, including transferring data between registers or between registers and external buses of the central processing unit (CPU), and performing arithmetic or logical operations on registers. The micro-operations in computers are classified into the following categories:

- Register transfer micro-operations: These type of micro operations are used to transfer from one register to another binary information.
- Arithmetic micro-operations : These micro-operations are used to perform on numeric data stored in the registers some arithmetic operations.
- Logic micro-operations: These micro operations are used to perform bit style operations / manipulations on non numeric data.
- Shift micro operations: As their name suggests they are used to perform shift operations in data store in registers.

Arithmetic Micro-Operations Some of the basic micro-operations are addition, subtraction, increment and decrement.

Add Micro-Operation It is defined by the following statement

$R3 \rightarrow R1 + R2$

The above statement instructs the data or contents of register R1 to be added to data or content of register R2 and the sum should be transferred to register R3.

Subtract Micro-Operation Let us again take an example:

$R3 \rightarrow R1 + R2' + 1$

In subtract micro-operation, instead of using minus operator we take **1's compliment** and add 1 to the register which gets subtracted, i.e **R1 - R2** is equivalent to **$R3 \rightarrow R1 + R2' + 1$**

Increment/Decrement Micro-Operation

Increment and decrement micro-operations are generally performed by adding and subtracting 1 to and from the register respectively.

$R1 \rightarrow R1 + 1$

$R1 \rightarrow R1 - 1$

Symbolic Designation	Description
$R3 \leftarrow R1 + R2$	Contents of R1+R2 transferred to R3.
$R3 \leftarrow R1 - R2$	Contents of R1-R2 transferred to R3.
$R2 \leftarrow (R2)'$	Compliment the contents of R2.
$R2 \leftarrow (R2)' + 1$	2's compliment the contents of R2.
$R3 \leftarrow R1 + (R2)' + 1$	R1 + the 2's compliment of R2 (subtraction).
$R1 \leftarrow R1 + 1$	Increment the contents of R1 by 1.
$R1 \leftarrow R1 - 1$	Decrement the contents of R1 by 1.

Logic Micro-Operations These are binary micro-operations performed on the bits stored in the registers. These operations consider each bit separately and treat them as binary variables.

Let us consider the X-OR micro-operation with the contents of two registers R1 and R2.

P: R1 ← R1 X-OR R2

In the above statement we have also included a Control Function.

Shift Micro-Operations These are used for serial transfer of data. That means we can shift the contents of the register to the left or right. In the shift left operation the serial input transfers a bit to the right most position and in shift right operation the serial input transfers a bit to the left most position.

There are three types of shifts as follows:

(a) **Logical Shift** It transfers 0 through the serial input. The symbol **"shl"** is used for logical shift left and "shr" is used for logical shift right.

R1 ← shl R1

R1 ← shr R1

The register symbol must be same on both sides of arrows.

(b) **Circular Shift** This circulates or rotates the bits of register around the two ends without any loss of data or contents. In this, the serial output of the shift register is connected to its serial input. "cil" and "cir" is used for circular shift left and right respectively.

(c) **Arithmetic Shift** This shifts a signed binary number to left or right. An arithmetic shift left multiplies a signed binary number by 2 and shift left divides the number by 2. Arithmetic shift micro-operation leaves the sign bit unchanged because the signed number remains same when it is multiplied or divided by 2.

Q.69. Explain the instruction cycle in detail with diagram. (2018)

Ans. Instruction Cycle

An instruction cycle, also known as fetch-decode-execute cycle is the basic operational process of a computer. This process is repeated continuously by CPU from boot up to shut down of computer.

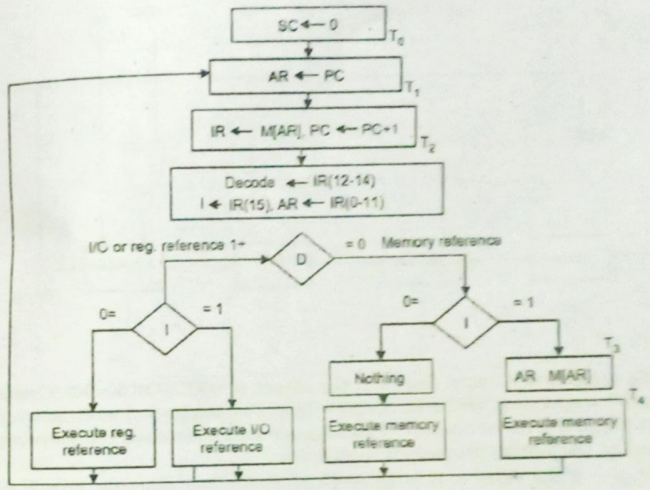
Following are the steps that occur during an instruction cycle:

1. **Fetch the Instruction:** The instruction is fetched from memory address that is stored in PC(Program Counter) and stored in the instruction register IR. At the end of the fetch operation, PC is incremented by 1 and it then points to the next instruction to be executed.

2. **Decode the Instruction:** The instruction in the IR is executed by the decoder.

3. **Read the Effective Address:** If the instruction has an indirect address, the effective address is read from the memory. Otherwise operands are directly read in case of immediate operand instruction.

4. **Execute the Instruction:** The Control Unit passes the information in the form of control signals to the functional unit of CPU. The result generated is stored in main memory or sent to an output device.



Q.70. Apply selective complement operation if content in A is 1010 and B is 1100. (2019)

Ans. The selective-complement operation complements bits in where ther are corresponding 1's in B

1010	A before
1100	B (logic operand)
0110	A after

A ← A ⊕ B

Q.71. Represent the following conditional control statement by two register transfer statements with control functions. (2019)

If (P = 1) then (R1 <-R2) else If (Q=1) then (R1<-R3)

Ans. P : R1 <- R2 ; If (P = 1) then (R1 <- R2)

P'Q : R1 *- R3 ; else if (Q = 1) then (R1 <- R3)

Q.72. Design a 4 bit full adder/subtractor circuit and explain the working.

Ans. A Binary Adder-Subtractor is one which is capable of both addition and Subtraction of binary numbers in one circuit itself. The operation being performed depends upon the binary value the cont.ol signal holds. It is one of the components of the ALU (Arithmetic Logic Unit).

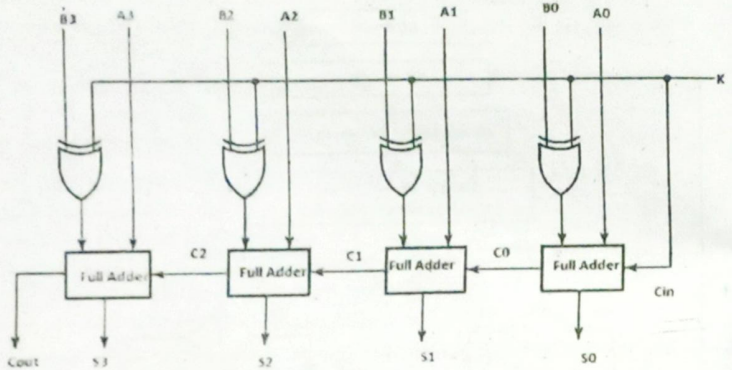
Consider two 4-bit binary numbers A and B as inputs to the Digital Circuit for the operation with digits

A0 A1 A2 A3 for A

B0 B1 B2 B3 for B

The circuit consists of 4 full adders since we are performing operation on 4-bit numbers. There is a control line K that holds a binary value of either 0 or 1 which determines that the operation being carried out is addition or subtraction.

As shown in the figure, the first full adder has control line directly as its input(input carry C0), The input A0 (The least significant bit of A) is directly input in the full adder. The third input is the EX-OR of B0 and K (S in fig But do not confuse it with Sum-S). The two outputs produced are Sum/Difference (S0) and Carry (C1).



If the value of K (Control line) is 1, the output of B0(EX-OR)K=B0′ (Complement B0). Thus the operation would be A+(B0′). Now 2's complement subtraction for two numbers A and B is given by A+B′. This suggests that when K=1, the operation being performed on the four bit numbers is subtraction.

Similarly, if the Value of K=0, B0 (EX-OR) K=B0. The operation is A+B which is simple binary addition. This suggests that when K=0, the operation being performed on the four bit numbers is addition.

Then C0 is serially passed to the second full adder as one of it's outputs. The sum/difference S0 is recorded as the least significant bit of the sum/difference. A1, A2, A3 are direct inputs to the second, third and fourth full adders. Then the third input is the B1, B2, B3 EX-OR with K to the second, third and fourth full adder respectively. The carry C1, C2 are serially passed to the successive full adder as one of the inputs. C3 becomes the total carry to the sum/difference. S1, S2, S3 are recorded to form the result with S0.

Q.73. A digital computer has a common bus system for 16 registers of 32 bits each. The bus is constructed with multiplexers. (2019)

(i) How many selection inputs are there in each multiplexer. (ii) What size of multiplexers are needed? (iii) How many multiplexers are there in the bus?

Ans. (i) 16 registers = 2^4 = 4 selection inputs

(ii) 16 registers = 16 × 1 line multiplexers

(iii) 32 bits / register = 32 multiplexers

Q.74. Explain Instruction pipelining and pipeline hazards. (2019)

Ans. Instruction pipelining is a technique for implementing instruction-level parallelism within a single processor. Pipelining attempts to keep every part of the processor busy with some instruction by dividing incoming instructions into a series of sequential steps performed by different processor units with different parts of instructions processed in parallel. It allows faster CPU throughput than would otherwise be possible at a given clock rate, but may increase latency due to the added overhead of the pipelining process itself.

A stream of instructions can be executed by overlapping fetch, decode and execute phases of an instruction cycle. This type of technique is used to increase the throughput of the computer system. An instruction pipeline reads instruction from the memory while previous instructions are being executed in other segments of the pipeline. Thus we can execute multiple instructions simultaneously. The pipeline will be more efficient if the instruction cycle is divided into segments of equal duration.

Step:	1	2	3	4	5	6	7	8	9	10	11	12	13
Instruction: 1	FI	DA	FO	EX									
2		FI	DA	FO	EX								
(Branch) 3			FI	DA	FO	EX							
4				FI	--	--	FI	DA	FO	EX			
5					--	--	--	FI	DA	FO	EX		
6									FI	DA	FO	EX	
7										FI	DA	FO	EX

Timing of instruction pipeline

Pipeline hazards : There are some factors that cause the pipeline to deviate its normal performance. Some of these factors are given below:

1. Timing Variations: All stages cannot take same amount of time. This problem generally occurs in instruction processing where different instructions have different operand requirements and thus different processing time.

2. Data Hazards: When several instructions are in partial execution, and if they reference same data then the problem arises. We must ensure that next instruction does not attempt to access data before the current instruction, because this will lead to incorrect results.

3. Branching: In order to fetch and execute the next instruction, we must know what that instruction is. If the present instruction is a conditional branch, and its result will lead us to the next instruction, then the next instruction may not be known until the current one is processed.

4. Interrupts: Interrupts set unwanted instruction into the instruction stream. Interrupts effect the execution of instruction.

5. Data Dependency: It arises when an instruction depends upon the result of a previous instruction but this result is not yet available.

Q.75. Explain Micro programmed Sequencer with help of circuit diagram. (2019)

Ans. A micro-program sequencer attached to a control memory inputs certain bits of the microinstruction, from which it determines the next address for control memory. A typical sequencer provides the following address-sequencing capabilities:

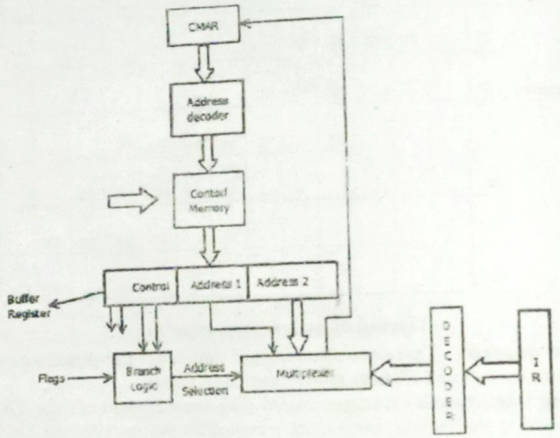
1. Increment the present address for control memory.
2. Branches to an address as specified by the address field of the micro instruction.
3. Brandies to a given address if a specified status bit is equal to 1.
4. Transfer control to a new address as specified by an external source (Instruction Register).
5. Has a facility for subroutine calls and returns.

Depending on the current microinstruction condition flags, and the contents of the instruction register, a control memory address must be generated for the next micro instruction.

There are two general techniques based on the format of the address information in the microinstruction:

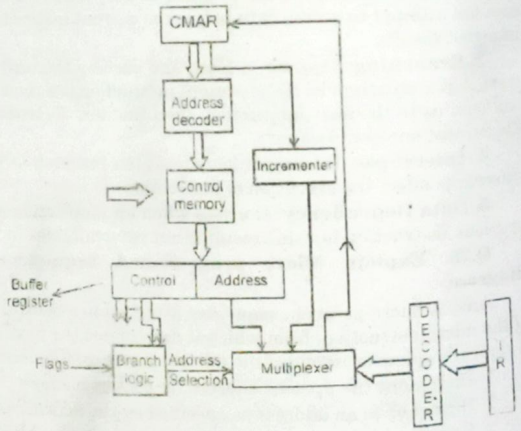
1. Two Address Field. 2. Single Address Field.

Two Address Field: The simplest approach is to provide two address field in each microinstruction and multiplexer is provided to select:



- Address from the second address field.
- Starting address based on the OPcode field in the current instruction.

The address selection signals are provided by a branch logic module whose input consists of control unit flags plus bits from the control partition of the microinstruction.



Single Address Field: Two-address approach is simple but it requires more bits in the microinstruction. With a simpler approach, we can have a single address field in the microinstruction with the following options for the next address.

- Address Field.
- Based on OPcode in instruction register.
- Next Sequential Address.

The address selection signals determine which option is selected. This approach reduces the number of address field to one. In most cases (in case of sequential execution) the address field will not be used. Thus the microinstruction encoding does not efficiently utilize the entire microinstruction.

UNIT –IV

Q.1. How is logical memory address different from physical memory address. **(2015)**

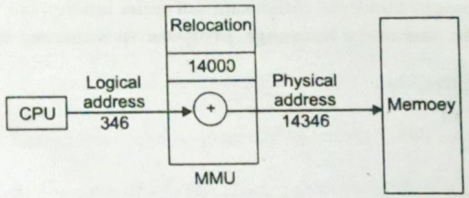
Ans. An address generated by the CPU is commonly referred to as a logical a logical address. The set of all logical addresses generated by a program is known as logical address space. Whereas, an address seen by the memory unit- that is, the one loaded into the memory-address register of the memory- is commonly referred to as physical address. The set of all physical addresses corresponding to the logical addresses is known as physical address space.

(2) The compile-time and load-time address-binding methods generate identical logical and physical addresses. However, in the execution-time address-binding scheme, the logical and physical-address spaces differ.

(3) The user program never sees the physical addresses. The program creates a pointer to a logical address, say 346, stores it in memory, manipulate it, compares it to other logical addresses–all as the number 346. Only when a logical address is used as memory address, it is relocated relative to the base/relocation register. The memory-mapping hardware device called the memory- management unit(MMU) converts logical addresses into physical addresses.

(4) Logical addresses range from 0 to max. User program that generates logical address thinks that the process runs in locations 0 to max. Logical addresses must be mapped to physical addresses before they are used. Physical addresses range from (R+0) to (R + max) for a base/relocation register value R.

(5) Example:



Mapping from logical to physical addresses using memory management unit (MMU) and relocation/base register. The value in relocation/base register is added to every logical address generated by a user process, at the time it is sent to memory, to generate corresponding physical address.

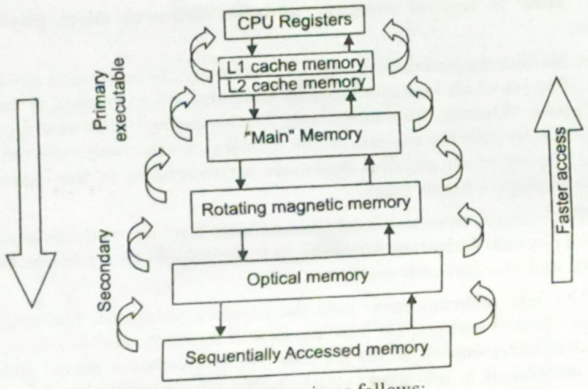
In the above figure, base/ relocation value is 14000, then an attempt by the user to access the location 346 is mapped to 14346.

Q.2. Explain the memory hierarchy with performace? **(2015)**

Ans. The hierarchical arrangement of storage in current computer architectures is called the memory hierarchy. It is designed to take advantage of memory locality in computer programs. Each level of the hierarchy is of higher speed and lower latency, and is of smaller size, than lower levels.

Most modern CPUs are so fast that for most program workloads the locality of reference of memory accesses, and the efficiency of the caching and memory transfer between different levels of the hierarchy, is the practical limitation on processing speed. As a result, the CPU spends much of its time idling, waiting for memory I/O to complete.

Memory Hierarchy



The memory hierarchy in most computers is as follows:

* Processor registers - fastest possible access (usually 1 CPU cycle), only hundreds of bytes in size
* Level 1 (L1) cache - often accessed in just a few cycles, usually tens of kilobytes
* Level 2 (L2) cache - higher latency than L1 by 2× to 10×, often 512KB or more
* Level 3 (L3) cache - (optional) higher latency than L2, often multiple MB's
* Main memory (DRAM) - may take hundreds of cycles, but can be multiple gigabytes
* Disk storage - hundreds of thousands of cycles latency, but very large.

Q.3. Write assembly language program to subtract two 8 bit numbers? (2015)

Ans. ; 2000H = 24
; 2001H = 43
LDA 2000H ;
MOV B,A ;
LDA 2001H ;
SUB B ;
DAA ;
STA 2002H ;
HLT ;

Q.4. The memory unit of a computer has 256K words of 32 bits each. The computer has an instruction format with four fields: an operation code field, a mode field, to specify one of seven addressing modes, a register address field to specify one of 60 processor registers, and a memory address. Specify the instruction format and the number of bits in each field if the instruction is one memory word long. (2015)

Ans. To address 256K we need 18 bits since 256K = 2^18.

To enumerate 7 addressing modes we need 3 bits since 2^3 = 8 what is the first higher number of combinations.

To enumerate 60 registers we need 6 bits since 2^6 = 64 what is the first higher number of combinations.

The rest of bits in the 32-bit word we can use for operation codes (opcodes),

32 - 18 - 3 - 6 = 5. So there can be 2^5 = 32 opcodes.

The instruction format can be:

31...27 - 26...24 - 23...18 - 17...0

opcode - mode - reg_addr - mem_addr

where reg_addr stands for "register address" and mem_addr stands for "memory address"

Q.5. What is cache memory? State the advantages of cache memory? (2015),(2016)

Ans. Cache memory is a small-sized type of volatile computer memory that provides high-speed data access to a processor and stores frequently used computer programs, applications and data. It stores and retains data only until a computer is powered up.

Cache memory is the fastest memory in a computer. It is typically integrated on the motherboard and directly embedded on the processor or main random access memory (RAM).

Cache memory provides faster data storage and access by storing an instance of programs and data routinely accessed by the processor. Thus, when a processor requests data that already has an instance in the cache memory, it does not need to go to the main memory or the hard disk to fetch the data.

Cache memory can be primary or secondary cache memory, where primary cache memory is directly integrated or closest to the processor. In addition to hardware-based cache, cache memory also can be a disk cache, where a reserved portion on a disk stores and provide access to frequently accessed data/applications from the disk.

Advantages: The advantages of cache memory are as follows:

- Cache memory is faster than main memory.
- It consumes less access time as compared to main memory.
- It stores the program that can be executed within a short period of time.
- It stores data for temporary use.

Q.6. A block set associative cache memory consist of 128 blocks divided into 4 block sets.

The main memory consists of 16, 384 blocks and each blocks contains 256 eight bit words.

(i) How many bits are required for addressing the main memory?

(ii) How many bits are needed to represent the TAG SET, WORD fields? (2015)

Ans. (i) The cache is divided into 16 sets of 4 lines each. Therefore, 4 bits are needed to identify the set number.

Main memory consists of 4K = 212 blocks. Therefore, the set plus tag lengths must be 12 bits and therefore the tag length is 8 bits.

Each block contains 128 words. Therefore, 7 bits are needed to specify the word.

Main memory address =	TAG	SET	WORD
	8	4	7

(ii) 8 left most bits = tag 5 middle bits = line number; 3 rightmost bits = byte number.

TAG	Line	WORD
8	5	7

Q.7. How memory management is done. (2015)

Ans. Memory Management: Main Memory refers to a physical memory that is the internal memory to the computer. The word main is used to distinguish it from external mass storage devices such as disk drives. Main memory is also known as RAM. The computer is able to change only data that is in main memory. Therefore, every program we execute and every file we access must be copied from a storage device into main memory.

All the programs are loaded in the main memeory for execution. Sometimes complete program is loaded into the memory, but some times a certain part or routine of the program is loaded into the main memory only when it is called by the program, this mechanism is called **Dynamic Loading,** this enhance the performance.

Also, at times one program is dependent on some other program. In such a case, rather than loading all the dependent programs, CPU links the dependent programs to the main executing program when its required. This mechanism is known as **Dynamic Linking.**

Swapping: A process needs to be in memory for execution. But sometimes there is not enough main memory to hold all the currently active processes in a time sharing system. So, excess process are kept on disk and brought in to run dynamically. Swapping is the process of bringing in each process in main memory, running it for a while and then putting it back to the disk.

Contiguous Memory Allocation: In contiguous memory allocation each process is contained in a single contiguous block of memory. Memory is divided into several fixed size partitions. Each partition contains exactly one process. When a partition is free, a process is selected from the input queue and loaded into it. The free blocks of memory are known as holes. The set of holes is searched to determine which hole is best to allocate.

Memory Protection: Memory protection is a phenomenon by which we control memory access rights on a computer. The main aim of it is to prevent a process from accessing memory that has not been allocated to it. Hence prevents a bug within a process from affecting other processes, or the operating system itself, and instead results in a segmentation fault or storage violation exception being sent to the disturbing process, generally killing of process.

Memory Allocation: Memory allocation is a process by which computer programs are assigned memory or space. It is of three types :

1. **First Fit:** The first hole that is big enough is allocated to program.
2. **Best Fit:** The smallest hole that is big enough is allocated to program.
3. **Worst Fit:** The largest hole that is big enough is allocated to program.

Fragmentation: Fragmentation occurs in a dynamic memory allocation system when most of the free blocks are too small to satisfy any request. It is generally termed as inability to use the available memory.

In such situation processes are loaded and removed from the memory. As a result of this, free holes exists to satisfy a request but is non contiguous i.e. the memory is fragmented into large no. Of small holes. This phenomenon is known as **External Fragmentation.**

Also, at times the physical memory is broken into fixed size blocks and memory is allocated in unit of block sizes. The memory allocated to a space may be slightly larger than the requested memory. The difference between allocated and required memory is known as **Internal fragmentation** i.e. the memory that is internal to a partition but is of no use.

Paging: A solution to fragmentation problem is Paging. Paging is a memory management mechanism that allows the physical address space of a process to be non-contagious. Here physical memory is divided into blocks of equal size called Pages. The pages belonging to a certain process are loaded into available memory frames.

Page Table: A Page Table is the data structure used by a virtual memory system in a computer operating system to store the mapping between virtual address and physical addresses.

Virtual address is also known as Logical address and is generated by the CPU. While Physical address is the address that actually exists on memory.

Segmentation: Segmentation is another memory management scheme that supports the user-view of memory. Segmentation allows breaking of the virtual address space of a single process into segments that may be placed in non-contiguous areas of physical memory.

Segmentation with Paging: Both paging and segmentation have their advantages and disadvantages, it is better to combine these two schemes to improve on each. The combined scheme is known as 'Page the Elements'. Each segment in this scheme is divided into pages and each segment is maintained in a page table. So the logical address is divided into following 3 parts :

• Segment numbers(S) • Page number (P) • The displacement or offset number (D)

Q.8. How many 128x8 RAM chips are required to provide a memory capacity of 2048 bytes? (2015)

Ans. 128 × 8 RAM => 128 Bytes (8bits) / chip => Numbers of chips = 2048 / 128 = 16

Q.9. How many lines of the address be must be used to access 2048 bytes of memory? How many of these lines ssssswill be common to all chips? (2015)

Ans. $2048 = 2^d$, Where d is nbr of address lines => d = 11 We have 16 chips => 4 bits for chip select Number of common address lines is 11 - 4 = 7 Summary: 4 bits (MSB) to select the correct chip and 7 bits (LSB) to select the memory location inside the selected chip.

Q.10. What is strobe control? (2016)

Ans. In computer or memory technology, a strobe is a signal that is sent that validates data or other signals on adjacent parallel lines. In memory technology, the CAS (column address strobe) and RAS (row address strobe) signals are used to tell a dynamic RAM that an address is a column or row address.

Q.11. Compare Direct mapping, Associative mapping and Set associative mapping. (2016)

Ans. In a **direct-mapped** cache, each address in main memory has one and only one place in the cache in which it can be stored. If two addresses in main memory map to the same place in the cache, only one of those two can be resident in cache at the same time. Direct-Mapped Cache is simplier (requires just one comparator and one multiplexer), as a result is cheaper and works faster. Given any address, it is easy to identify the single entry in cache, where it can be. A major drawback when using DM cache is called a conflict miss, when two different addresses correspond to one entry in the cache. Even if the cache is big and contains many stale entries, it can't simply evict those, because the position within cache is predetermined by the address.

In a **set-associative** cache, each address in main memory has some number of places in the cache in which it can reside. Thus there are no two addresses that cannot

reside in cache at the same time. A set-associative scheme is a hybrid between a fully associative cache, and direct mapped cache. It's considered a reasonable compromise between the complex hardware needed for fully associative caches (which requires parallel searches of all slots), and the simplistic direct-mapped scheme, which may cause collisions of addresses to the same slot (similar to collisions in a hash table).

Q.12. Differentiate between access time and cycle time of a memory. (2016)

Ans. Memory access time is corresponding to the time interval between the read/write request and the availability of the data. The time required by a processor to access data or to write data from and to memory chip is referred as access time. Cache memory has the shortest access time. Access time is also frequently used to describe the speed of disk drives. Disk access times are measured in milliseconds (thousandths of a second), often abbreviated as ms. Fast hard disk drives for personal computers boast access times of about 9 to 15 milliseconds. The access time for disk drives includes the time it actually takes for the read/write head to locate a sector on the disk (called the seek time). This is an average time since it depends on how far away the head is from the desired data.

Cycle time represents the minimum time interval between two successive accesses. The memory cycle time is a measurement of how quickly two back-to-back accesses of a memory chip can be made. Note that a DRAM chip's cycle time is usually much longer than its access time, which measures only a single access. This is because there is a latency between successive memory accesses. Cycle time is the time, usually measured in nanosecond s, between the start of one random access memory access to the time when the next access can be started.

Q.13. Differentiate between Asynchronous Data Transfers and synchronous data transfer. (2016)

Ans. Synchronous: In synchronous data transfers, the sender and receiver take some time to communicate before they make the exchange. This communication outlines the parameters of the data exchange. This usually involves establishing which end, sender or receiver, will be in control of the transfer. Here, the two parties also ensure they are using the same timing; that is, they know when each burst ends and another begins. They also set parameters for resetting their clocks during the transfer to make sure they don't drift away from the agreed-upon timing. A synchronous data transfer is the generation and transferring of data on a Clock with known phase and frequency. The data is synchronized according to the Setup and Hold requirements of the known clock.Synchronous messaging involves a client that waits for the server to respond to a message. Messages are able to flow in both directions, to and from.

Asynchronous: In asynchronous, or "best effort" transfers, sender and receiver do not establish the parameters of the information exchange. Rather, the sender places extra bits of data before and after each burst that indicate when each burst begins and ends. It then sends the information, and it is up to the receiver to determine how to reset its clock to match the timing of the signal. Unlike synchronous transfers, the receiver does not take time to communicate to the sender information about what it received. Whereas, asynchronous data transfer could be the one where a data is generated on, say, CLOCK 1 and it is sampled on any other clock, say, CLOCK 2 (Where clock 1 and clock 2 are asynchronous to each other and has no known phase and frequency relationship). Then the data will arrive asynchronously to CLOCK 2 and it will be an asynchronous data transfer. Asynchronous messaging involves a client that does not wait for a message from the server.

Q.14. Differentiate between Unsigned notation and, signed notation. Find range of 2 byte integer in both cases. (2016)

Ans. Signed variables, such as signed integers will allow you to represent numbers both in the positive and negative ranges. Unsigned variables, such as unsigned integers, will only allow you to represent numbers in the positive. Unsigned and signed variables of the same type (such as int and byte) both have the same range (range of 65,536 and 256 numbers, respectively), but unsigned can represent a larger magnitude number than the corresponding signed variable. Signed variables use one bit to flag whether they are positive or negative. Unsigned variables don't have this bit, so they can store larger numbers in the same space, but only nonnegative numbers, e.g. 0 and higher. Unsigned variables are variables which are internally represented without a mathematical sign (plus or minus) can store 'zero' or positive values only. Let us say the unsigned variable is n bits in size, then it can represent 2^n (2 power n) values - 0 through (2^n –1). A signed variable on the other hand, 'loses' one bit for representing the sign, so it can store values from -(2^(n–1) –1) through (2^(n–1)) including zero. Thus, a signed variable can store positive values, negative values and zero.

Here n= 2*8=16 bits

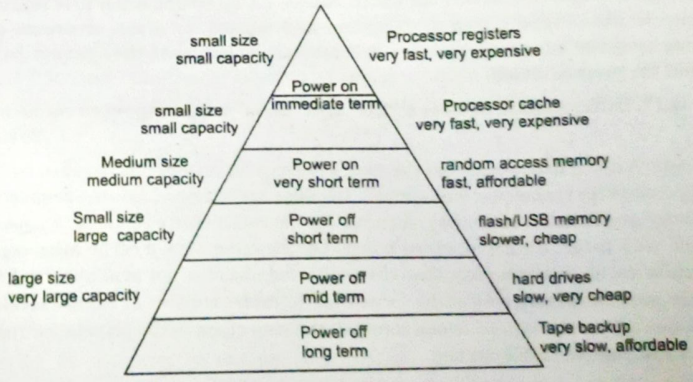
So for unsigned notation range is from – 0 to 65535

while for signed notation range is from – –32768 to + 32767

Q.15. Explain the need of memory hierarchy with the help of a block diagram? What is the reason for not having one large memory unit for storing all information at one place?

Ans. In computer architecture the memory hierarchy is a concept used to discuss performance issues in computer architectural design, algorithm predictions, and lower level programming constructs involving locality of reference. The memory hierarchy in computer storage separates each of its levels based on response time. Since response time, complexity, and capacity are related,the levels may also be distinguished by their performance and controlling technologies. Designing for high performance requires considering the restrictions of the memory hierarchy, i.e. the size and capabilities of each component. Each of the various components can be viewed as part of a hierarchy of memories (m1, m2,...,mn) in which each member mi is typically smaller and faster than the next highest member mi+1 of the hierarchy. To limit waiting by higher levels, a lower level will respond by filling a buffer and then signaling to activate the transfer.

Compare Memory Hierarchy



A large single unit is not used whereas the memory hierarchy is used since it increases performance by saving time based on locality of reference. A "memory hierarchy" in computer storage distinguishes each level in the "hierarchy" by response time. Each level of the hierarchy is of higher speed and lower latency, and is of smaller size, than lower levels.

Q.16. Discuss about Input-Output and Interrupts in detail. **(2016)**

Ans. INPUT-OUTPUT: In computing, input/output or I/O (or, informally, io or IO) is the communication between an information processing system, such as a computer, and the outside world, possibly a human or another information processing system. Inputs are the signals or data received by the system and outputs are the signals or data sent from it. The term can also be used as part of an action; to "perform I/O" is to perform an input or output operation. I/O devices are used by a human (or other system) to communicate with a computer. For instance, a keyboard or computer mouse is an input device for a computer, while monitors and printers are output devices.

Devices for communication between computers, such as modems and network cards, typically perform both input and output operations. The designation of a device as either input or output depends on perspective. Mouse and keyboards take physical movements that the human user outputs and convert them into input signals that a computer can understand; the output from these devices is the computer's input. Similarly, printers and monitors take signals that a computer outputs as input, and they convert these signals into a representation that human users can understand. From the human user's perspective, the process of reading or seeing these representations is receiving input; this type of interaction between computers and humans is studied in the field of human–computer interaction.

INTERRUPTS: In system programming, an interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. An interrupt alerts the processor to a high-priority condition requiring the interruption of the current code the processor is executing. Hardware interrupts are used by devices to communicate that they require attention from the operating system.

Internally, hardware interrupts are implemented using electronic alerting signals that are sent to the processor from an external device, which is either a part of the computer itself, such as a disk controller, or an external peripheral. A software interrupt is caused either by an exceptional condition in the processor itself, or a special instruction in the instruction set which causes an interrupt when it is executed. The former is often called a trap or exception and is used for errors or events occurring during program execution that are exceptional enough that they cannot be handled within the program itself.

Q.17. Differentiate between "hit" and "miss" with respect to cache memory. **(2015),(2017)**

Ans. A cache **hit** means that the user's request for an object is satisfied by the cache entity (computer cache/http web cache/CDN edge server etc.), say, the requested object is exactly presented in the entity. As a result, the cache entity sends back object data to the end user to facilitate a quick response. On the other side, a cache **miss** regarding a particular cache server means that the requested object is not available in the storage , so the request message must be forwarded to cache entities at higher levels. In the worst case , the user request is delivered all the way along to the original storage server (such as an remote HTTP server.

Q.18. How interrupts are handled? Explain. **(2017)**

Ans. If there is an interrupt present then it will trigger the interrupt handler, the handler will stop the present instruction which is processing and save its configuration in a register and load the program counter of the interrupt from a location which is given by the interrupt vector table. After processing the interrupt by the processor interrupt handler will load the instruction and its configuration from the saved register, process will start its processing where it's left. This saving the old instruction processing configuration and loading the new interrupt configuration is also called as context switching. The interrupt handler is also called as Interrupt service routine (ISR). There are different types of interrupt handler which will handle different interrupts.

Q.19. Explain Virtual memory. Also state "Locality of Reference" principle. **(2017)**

Ans. In computing, virtual memory is a memory management technique that provides an "idealized abstraction of the storage resources that are actually available on a given machine" which "creates the illusion to users of a very large memory.

In computer science, **locality of reference,** also known as the principle of locality, is a term for the phenomenon in which the same values, or related storage locations, are frequently accessed, depending on the memory access pattern. There are two basic types of reference locality – temporal and spatial **locality**.

Q.20. What are the different kinds of operation used in CPU design? **(2017)**

Ans. The micro-operations in computers are classified into the following categories:

- **Register transfer micro-operations:** These type of micro operations are used to transfer from one register to another binary information.
- **Arithmetic micro-operations :** These micro-operations are used to perform on numeric data stored in the registers some arithmetic operations.
- Logic micro-operations: These micro operations are used to perform bit style operations / manipulations on non numeric data.
- **Shift micro operations:** As their name suggests they are used to perform shift operations in data store in registers

Q.21. Define DMA. Explain its need. Explain DMA transfer in detail with the help of suitable diagram. **(2017),(2018)**

Ans. Stands for "Direct Memory Access." DMA is a method of transferring data from the computer's RAM to another part of the computer without processing it using the CPU. While most data that is input or output from your computer is processed by the CPU, some data does not require processing, or can be processed by another device. In these situations, DMA can save processing time and is a more efficient way to move data from the computer's memory to other devices.

For example, a sound card may need to access data stored in the computer's RAM, but since it can process the data itself, it may use DMA to bypass the CPU. Video cards that support DMA can also access the system memory and process graphics without needing the CPU. Ultra DMA hard drives use DMA to transfer data faster than previous hard drives that required the data to first be run through the CPU.

DMA Transfer Types

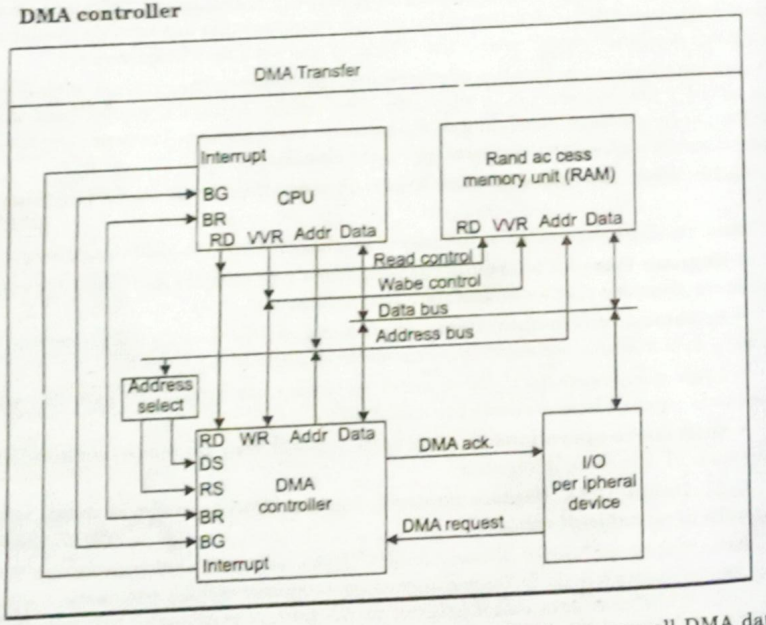
Memory To Memory Transfer: In this mode block of data from one memory address is moved to another memory address. In this mode current address register of

channel 0 is used to point the source address and the current address register of channel is used to point the destination address in the first transfer cycle, data byte from the source address is loaded in the temporary register of the DMA controller and in the next transfer cycle the data from the temporary register is stored in the memory pointed by destination address.

Auto initialize: In this mode, during the initialization the base address and word count registers are loaded simultaneously with the current address and word count registers by the microprocessor. The address and the count in the base registers remain unchanged throughout the DMA service.

After the first block transfer i.e. after the activation of the EOP signal, the original values of the current address and current word count registers are automatically restored from the base address and base word count register of that channel.

DMA controller



The controller is integrated into the processor board and manages all DMA data transfers. Transferring data between system memory and an 110 device requires two steps. Data goes from the sending device to the DMA controller and then to the receiving device. The microprocessor gives the DMA controller the location, destination, and amount of data that is to be transferred. Then the DMA controller transfers the data, allowing the microprocessor to continue with other processing tasks. When a device needs to use the Micro Channel bus to send or receive data, it competes with all the other devices that are trying to gain control of the bus. This process is known as arbitration. The DMA controller does not arbitrate for control of the BUS instead; the I/O device that is sending or receiving data (the DMA slave) participates in arbitration.

Q.22. Main memory size is 128 KB, cache size is 16 KB, block size is 256B. Using direct bit mapping what is the no of tags bits in physical address and what is the size of tag directory. Assume memory is byte addressable. (2017)

Ans.

tag	index	block

Tag bits = address bit length – exponent of index- exponent of offset

Index bits = log(cache size)= log $(16*2^{10})$ = 14

Block offset = 8

Tag bits= 32- 14- 8 = 10

Tag directory= no. of lines * no. of tag bits = 14* 10 = 140

Q.23. Types of interrupt and Memory hierarchy (2017)

Ans. Interrupt is a signal which has highest priority from hardware or software which processor should process its signal immediately.

Types of Interrupts:

Although interrupts have highest priority than other signals, there are many type of interrupts but basic type of interrupts are

1.Hardware Interrupts: If the signal for the processor is from external device or hardware is called hardware interrupts. Example: from keyboard we will press the key to do some action this pressing of key in keyboard will generate a signal which is given to the processor to do action, such interrupts are called hardware interrupts. Hardware interrupts can be classified into two types they are

- **Maskable Interrupt:** The hardware interrupts which can be delayed when a much highest priority interrupt has occurred to the processor.
- **Non Maskable Interrupt:** The hardware which cannot be delayed and should process by the processor immediately.

2. Software Interrupts: Softwaref interrupt can also divided in to two types. They are

- **Normal Interrupts:** the interrupts which are caused by the software instructions are called software instructions.
- **Exception:** unplanned interrupts while executing a program is called Exception. For example: while executing a program if we got a value which should be divided by zero is called a exception.

In computer architecture, the **memory hierarchy** separates computer storage into a hierarchy based on response time. Since response time, complexity, and capacity are related, the levels may also be distinguished by their performance and controlling technologies Memory hierarchy affects performance in computer architectural design, algorithm predictions, and lower level programming constructs involving locality of reference.

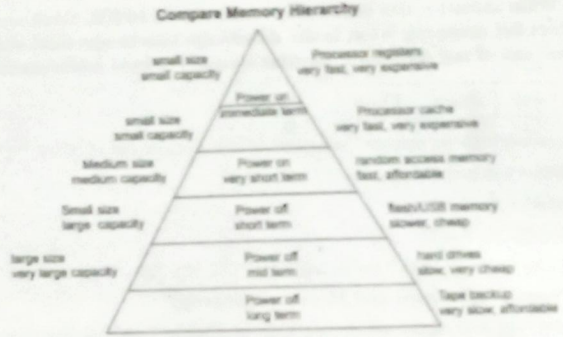
There are four major storage levels.

1.Internal – Processor registers and cache.

2.Main – the system RAM and controller cards.

3.On-line mass storage – Secondary storage.

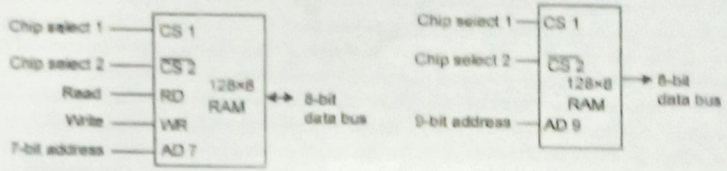
4.Off-line bulk storage – Tertiary and Off-line storage.



Q.24. Differentiate between RAM and ROM chips? **(2018)**

Ans.

	RAM	ROM
Accessibility	-The information stored in RAM is easily accessed because it s communicated directly with the processor	- The processor cannot directly access the information. Hence, the information will be transferred into the RAM and then it gets executed by the process to access the ROM information
Volatility	-Volatile in nature. - Data s stored as long as the power supply as the power supply is switched on. - Data with eased if the computer crashes or is turned off.	- Non-volatile in nature. - Data is stored even the power supply is switched off. - Data is retain even if the computer crashes or it turned off.
Storage	- Data is temporary - It is only there as long as the computer is on and it can be changed	- Data is permanent - It can never be changed - Contents are remain same
Speed	- The accessing speed of RAM is faster, it assist the processor to boost up the speed	- Speed is slower compared to RAM, ROM cannot boost up the processor speed
Data Preserving	- Electricity supply is needed in RAM to flow to preserving information	- Electricity supply is not needed in ROM to flow to preserving information
Structure	- The RAM is an chip, which is in the rectangle form and is inserted over the mother board of the computer	- ROMs are generally the optical drivers, which are made of magnetic tapes.
Cost	- The price of RAMs are comparatively high	- The price of ROMs are comparatively low
Chip size	- Physically size of RAM chip is larger than ROM chip	- Physically size of ROM chip is smaller than RAM chip
Category	-Read write memory - Data can be written to or read from.	- Read-only memory - Data can only be read - User cannot make any changes to the information.



Q.25. A computer uses a memory unit with 256K words of 32 bits each. A binary instruction code is stored in one word of memory.

The instruction has four parts: an indirect bit, an operation code, a register code the register code part to specify one of 64 registers and an address part.

(i) How many bits are there in the operation code, the register code part, and the address part?

(ii) Draw the instruction word format and indicate the number of bits in each part.

(iii) How many bits are there in the data and address inputs if the memory?

Ans. $256\ K = 2^3 \times 2^{10} = 2^{18}$

$64 = 2^6$

(i) Address : 18 bits $64 = 2^6$

Register code : 6 bits

Indirect bit: $\frac{1}{25}$ bits

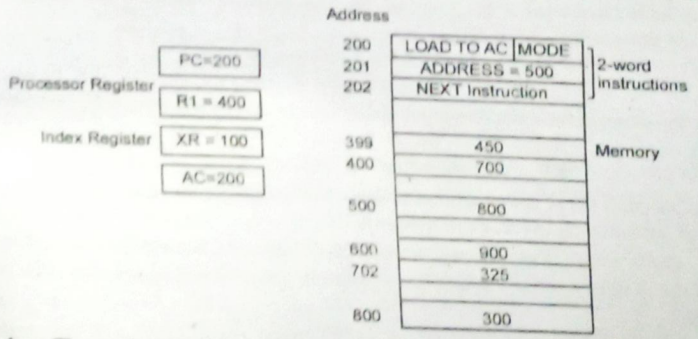
$32 - 25 = 7$ bits for opcode

(ii)

1	7	6	18	
I	opcode	Register	Address	= 32 bits

(iii) Data: 32 bits; address: 18 bits

Q.26. For each possible addressing mode calculate the effective address and content of accumulator. **(2019)**



Ans. The mode field of the instruction can specify any one of a number of modes. For each possible mode we calculate the effective address and the operand that must be loaded into AC. In the direct address mode the effective address is the address part

of the instruction 500 and the operand to be loaded into AC is 800. In the immediate mode the second word of the instruction is taken as the operand rather than an address, so 500 is loaded into AC. (The effective address in this case is 201.) In the indirect mode the effective address is stored in memory at address 500. Therefore, the effective address is 800 and the operand is 300. In the relative mode the effective address is 500 + 202 = 702 and the operand is 325. (Note that the value in PC after the fetch phase and during the execute phase is 202) In the index mode the effective address is XR + 500 = 100 + 500 = 600 and the operand is 900. In the register mode the operand is in R1 and 400 is loaded into AC. (There is no effective address in this case.) In the register indirect mode the effective address is 400, equal to the content of R1 and the operand loaded into AC is 700. The autoincrement mode is the same as the register indirect mode except that R1 is incremented to 401 after the execution of the instruction. The autodecrement mode decrements R1 to 399 prior to the execution of the instruction. The operand loaded into AC is now 450. Table lists the values of the effective address and the operand loaded into AC for the nine addressing modes.

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

Q.27. What is speed Up ratio? Explain. **(2019)**

Ans. Consider a 'k' segment pipeline with clock cycle time as 'Tp'. Let there be 'n' tasks to be completed in the pipelined processor. Now, the first instruction is going to take 'k' cycles to come out of the pipeline but the other 'n –1' instructions will take only '1' cycle each, i.e, a total of 'n – 1' cycles. So, time taken to execute 'n' instructions in a pipelined processor:

$$ET_{pipeline} = k + n - 1 \text{ cycles}$$
$$= (k + n - 1)Tp$$

In the same case, for a non-pipelined processor, execution time of 'n' instructions will be:

$$ET_{non\text{-}pipeline} = n * k * Tp$$

So, speedup (S) of the pipelined processor over non-pipelined processor, when 'n' tasks are executed on the same processor is:

S = Performance of pipelined processor/Performance of Non-pipelined processor

As the performance of a processor is inversely proportional to the execution time, we have,

$$S = ET_{non\text{-}pipeline}/ET_{pipeline}$$
$$\Rightarrow \quad S = [n * k * Tp]/[(k + n - 1) * Tp]$$
$$S = [n * k]/[k + n - 1]$$

When the number of tasks n are significantly larger than k, that is, n >> k

$$S = n * k/n$$
$$S = k$$

where k are the number of stages in the pipeline.

Q.28. Difference between Isolated I/O and Memory Mapped I/O. **(2019)**

Ans. Differences between memory mapped I/O and isolated I/O –

ISOLATED I/O	MEMORY MAPPED I/O
Memory and I/O have separate address space	Both have same address space
All address can be used by the memory	Due to addition of I/O addressable memory become less for memory
Separate instruction control read and write operation in I/O and Memory	Same instructions can control both I/O and Memory
In this I/O address are called ports.	Normal memory address are for both
More efficient due to separate buses	Lesser efficient
Larger in size due to more buses	Smaller in size
It is complex due to separate separate logic is used to control both.	Simpler logic is used as I/O is also treated as memory only.

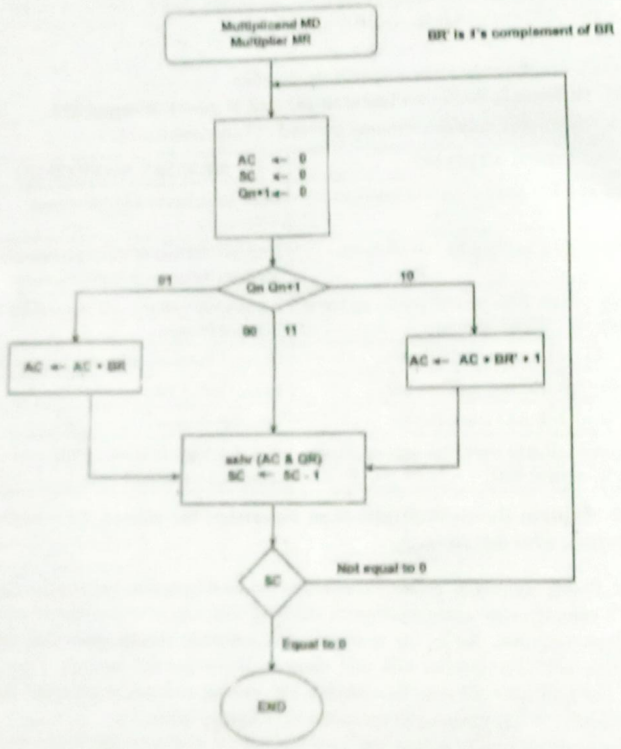
Q.29. Explain Booth Multiplication algorithm for signed 2's complement presentation with example. **(2019)**

Ans. Booth algorithm gives a procedure for **multiplying binary integers** in signed 2's complement representation in efficient way, i.e., less number of additions/ subtractions required. As in all multiplication schemes, booth algorithm requires examination of the multiplier bits and shifting of the partial product. Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to following rules:

1. The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier
2. The multiplicand is added to the partial product upon encountering the first 0 (provided that there was a previous '1') in a string of 0's in the multiplier.
3. The partial product does not change when the multiplier bit is identical to the previous multiplier bit.

Booth's Algorithm Flowchart: We name the register as A, B and Q, AC, BR and QR respectively. Q_n designates the least significant bit of multiplier in the register QR. An extra flip-flop Q_{n+1} is appended to QR to facilitate a double inspection of the multiplier. The flowchart for the booth algorithm is shown below.

AC and the appended bit Q_{n+1} are initially cleared to 0 and the sequence SC is set to a number n equal to the number of bits in the multiplier. The two bits of the multiplier in Q_n and Q_{n+1} are inspected. If the two bits are equal to 10, it means that the first 1 in a string has been encountered. This requires subtraction of the multiplicand from the partial product in AC. If the 2 bits are equal to 01, it means that the first 0 in a string of 0's has been encountered. This requires the addition of the multiplicand to the partial product in AC.



When the two bits are equal, the partial product does not change. An overflow cannot occur because the addition and subtraction of the multiplicand follow each other. As a consequence, the 2 numbers that are added always have a opposite signs, a condition that excludes an overflow. The next step is to shift right the partial product and the multiplier (including Q_{n+1}). This is an arithmetic shift right (ashr) operation which AC and QR to the right and leaves the sign bit in AC unchanged. The sequence counter is decremented and the computational loop is repeated n times.

Example - A numerical example of booth's algorithm is shown below for n = 4. It shows the step by step multiplication of -5 and -7.

$MD = -5 = 1011$, $MD = 1011$, $MD's\ 1 = 0101$

$MR = -7 = 1001$

The explanation of first step is as follows: Q_{n+1}

$AC = 0000$, $MR = 1001$, $Q_{n+1} = 0$, $SC = 4$

$Q_n\ Q_{n+1} = 10$

So, we do $AC = (MD's + 1$, which gives $AC = 0101$

On right shifting AC and MR, we get

$AC = 0010$, $MR = 1100$ and $Q_{n+1} = 1$

OPERATION	AC	MR	Q_{n+1}	SC
	0000	1001	0	4
AC + MD's1	0101	1001	0	
ASHR	0010	1100	1	3
AC + MR	1101	1100	1	
ASHR	1110	1110	0	2
ASHR	1111	0111	0	1
AC + MD' + 1	0010	0011	1	0

Product is calculated as follows:

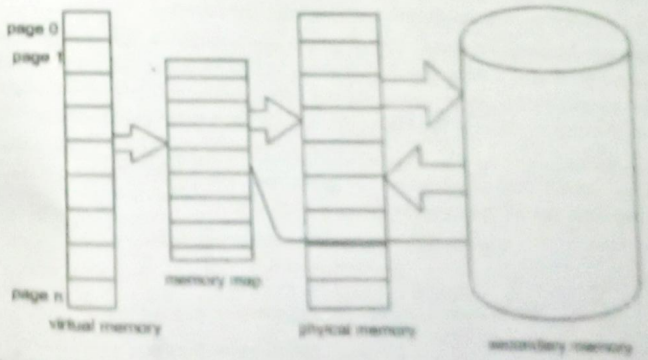
Product = AC MR

Product = 0010 0011 = 35

Q.30. Virtual Memory (2015),(2019)

Ans. Virtual Memory: Virtual memory is the separation of logical memory from physical memory. This separation provides large virtual memory for programmers when only small physical memory is available.

Virtual memory is used to give programmers the illusion that they have a very large memory even though the computer has a small main memory. It makes the task of programming easier because the programmer no longer needs to worry about the amount of physical memory available.



Basically, virtual memory provides an illusion to the users that the PC has enough primary memory left to run the programs. Though the size of program i.e. to be executed may sometimes very bigger than the size of primary memory left, the user never feels that he needs a bigger primary storage to run that program. When the RAM is full, but program execution needs more space in RAM, then the operating system occupies a portion of the hard disk and uses it as a RAM. In that part of the secondary storage, the part of the program which not currently being executed is stored and all the parts of the program that are eventually executed are first brought into the main memory.

Q.31. Modes of Transfer for I/O devices

Ans. Data transfer to and from the peripherals may be done in any of the three possible way:

1. Programmed I/O: It is due to the result of the I/O instructions that are written in the computer program. Each data item transfer is initiated by an instruction in the program. Usually the transfer is from a CPU register and memory. In this case it requires constant monitoring by the CPU of the peripheral devices.

Example of Programmed I/O: In this case, the I/O device does not have direct access to the memory unit. A transfer from I/O device to memory requires the execution of several instructions by the CPU, including an input instruction to transfer the data from device to the CPU and store instruction to transfer the data from CPU to memory, In programmed I/O, the CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer. This is a time consuming process since it needlessly keeps the CPU busy. This situation can be avoided by using an interrupt facility.

2. Interrupt- initiated I/O: Since in the above case we saw the CPU is kept busy unecessarily. This situation can very well be avoided by using an interrupt driven method for data transfer. By using interrupt facility and special commands to inform the interface to issue an interrupt request signal whenever data is available from any device. In the meantime the CPU can proceed for any other program execution. The interface meanwhile keeps monitoring the device. Whenever it is determined that the device is ready for data transfer it initiates an interrupt request signal to the computer. Upon detection of an external interrupt signal the CPU stops momentarily the task that it was already performing, branches to the service program to process the I/O transfer, and then return to the task it was originally performing.

Both the methods programmed I/O and Interrupt-driven I/O require the active intervention of the processor to transfer data between memory and the I/O module, and any data transfer must transverses path through the processor.

3. Direct Memory Access: The data transfer between a fast storage media such as magnetic disk and memory unit is limited by the speed of the CPU. Thus we can allow the peripherals directly communicate with each other using the memory buses, removing the intervention of the CPU. This type of data transfer technique is known as DMA or direct memory access. During DMA the CPU is idle and it has no control over the memory buses. The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit.

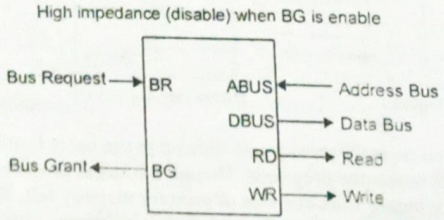


Fig. CPU Bus Signals for DMA Transfer

Bus Request: It is used by the DMA controller to request the CPU to relinquish the control of the buses.

Bus Grant: It is activated by the CPU to inform the external DMA controller that the buses are in high impedance state and the requesting DMA can take control of the buses. Once the DMA has taken the control of the buses it transfers the data. This transfer can take place in many ways.

Types of DMA transfer using DMA controller:

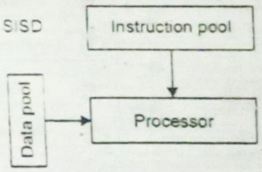
Burst Transfer: DMA returns the bus after complete data transfer. A register is used as a byte count, being decremented for each byte transfer, and upon the byte count reaching zero, the DMAC will release the bus. When the DMAC operates in burst mode, the CPU is halted for the duration of the data transfer.

Cyclic Stealing: In this DMA controller transfers one word at a time after which it must return the control of the buses to the CPU. The CPU merely delays its operation for memory cycle to allow the direct memory I/O transfer to "steal" one memory cycle.

Q.32. Flynn's classification? (2019)

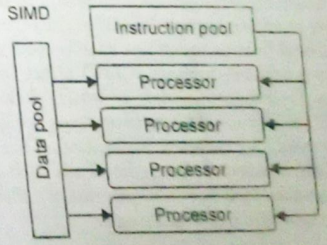
Ans. Flynn's classification -

1. Single-instruction, single-data (SISD) systems: An SISD computing system is a uniprocessor machine which is capable of executing a single instruction, operating on a single data stream. In SISD, machine instructions are processed in a sequential manner and computers adopting this model are popularly called sequential computers. Most conventional computers have SISD architecture. All the instructions and data to be processed have to be stored in primary memory.



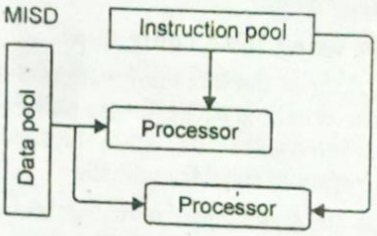
The speed of the processing element in the SISD model is limited(dependent) by the rate at which the computer can transfer information internally. Dominant representative SISD systems are IBM PC, workstations.

2. Single-instruction, multiple-data (SIMD) systems - An SIMD system is a multiprocessor machine capable of executing the same instruction on all the CPUs but operating on different data streams. Machines based on an SIMD model are well suited to scientific computing since they involve lots of vector and matrix operations. So that the information can be passed to all the processing elements (PEs) organized data elements of vectors can be divided into multiple sets (N-sets for N PE systems) and each PE can process one data set.



Dominant representative SIMD systems is Cray's vector processing machine.

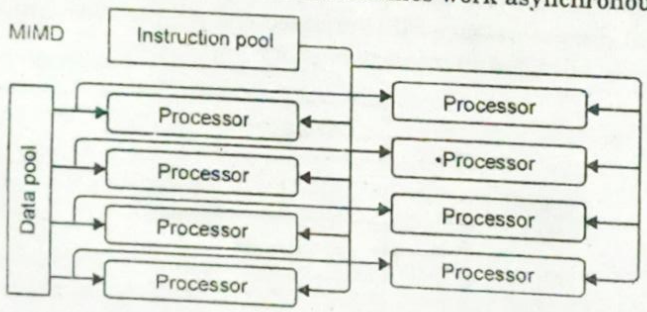
3. Multiple-instruction, single-data (MISD) systems: An MISD computing system is a multiprocessor machine capable of executing different instructions on different PEs but all of them operating on the same dataset.



Example $Z = \sin(x) + \cos(x) + \tan(x)$

The system performs different operations on the same data set. Machines built using the MISD model are not useful in most of the application, a few machines are built, but none of them are available commercially.

4. Multiple-instruction, multiple-data (MIMD) systems: An MIMD system is a multiprocessor machine which is capable of executing multiple instructions on multiple data sets. Each PE in the MIMD model has separate instruction and data streams; therefore machines built using this model are capable to any kind of application. Unlike SIMD and MISD machines, PEs in MIMD machines work asynchronously.



MIMD machines are broadly categorized into **shared-memory MIMD and distributed-memory MIMD** based on the way PEs are coupled to the main memory.

In the **shared memory MIMD** model (tightly coupled multiprocessor systems), all the PEs are connected to a single global memory and they all have access to it. The communication between PEs in this model takes place through the shared memory, modification of the-data stored in the global memory by one PE is visible to all other PEs. Dominant representative shared memory MIMD systems are Silicon Graphics machines and Sun/IBM's SMP (Symmetric Multi-Processing).

In **Distributed memory MIMD** machines (loosely coupled multiprocessor systems) all PEs have a local memory. The communication between PEs in this model takes place through the interconnection network (the inter process communication channel, or IPC). The network connecting PEs can be configured to tree, mesh or in accordance with the requirement. The shared-memory MIMD architecture is easier to program but is less tolerant to failures and harder to extend with respect to the distributed memory MIMD model. Failures in a shared-memory MIMD affect the entire system, whereas this is not the case of the distributed model, in which each of the PEs can be easily isolated. Moreover, shared memory MIMD architectures are less likely to scale because the addition of more PEs leads to memory contention. This is a situation that does not happen in the case of distributed memory, in which each PE has its own memory. As a result of practical outcomes and user's requirement , distributed memory MIMD architecture is superior to the other existing models.

Our Useful Solutions For GGSIPU

- SOLVED PAPERS FOR B.TECH
- SOLVED PAPERS FOR BBA
- SOLVED PAPER FOR BCA

Upcoming Solutions For GGSIPU

- SOLVED PAPERS FOR MCA
- SOLVED PAPERS FOR MBA

AKASH BOOKS

Sales Office: H.No.4278, Gali No. 3,
Ansari Road, Darya Ganj, New Delhi-110002
Ph: 011-23281171,0-9818257423,9971086337
E-mail: akashbooks@rediffmail.com