

# Syllabus

## COMPUTER SCIENCE AND INFORMATION TECHNOLOGY [CS/IT]

Units	Contents of the Subject
Unit 1	<b>THEORY OF COMPUTATION:</b> Regular languages and finite automata, Context free languages and Push-down automata, Recursively enumerable sets and Turing machines, Undecidability, NP-completeness.
Unit 2.	<b>DIGITAL LOGIC:</b> Logic functions, Minimization, Design and Synthesis of Combinational and Sequential circuits, Number representation and Computer arithmetic (fixed and floating point).
Unit 3.	<b>COMPUTER ORGANIZATION AND ARCHITECTURE:</b> Machine instructions and Addressing modes, ALU and data-path, CPU control design, Memory interface, I/O interface (Interrupt and DMA mode), Instruction pipelining, Cache and main memory, Secondary storage.
Unit 4.	<b>PROGRAMMING:</b> Functions, Recursion, Parameter passing, Scope, Binding, Abstract data types, Arrays.
Unit 5.	<b>DATA STRUCTURES AND ALGORITHMS:</b> Analysis, Asymptotic notation, Notions of space and Time complexity, Worst and Average case analysis, Design, Greedy approach, Dynamic programming, Divide and Conquer, Tree and Graph traversals, Connected components, Spanning trees, Shortest paths, Hashing, Sorting, Searching.
Unit 6.	<b>COMPILER DESIGN:</b> Lexical analysis, Parsing, Syntax directed translation, Runtime environments, Intermediate and target code generation, Basics of code optimization.
Unit 7.	<b>OPERATING SYSTEM:</b> Processes, Threads, Inter-process communication, Concurrency, Synchronization, Deadlock, CPU scheduling, Memory management and Virtual memory, File systems, I/O systems, Protection and Security.
Unit 8.	<b>DATABASE:</b> ER-model, Relational model (relational algebra, tuple calculus), Database design (integrity constraints, normal forms), Query languages (SQL), File structures (sequential files, indexing, B and B+ trees), Transactions and Concurrency control.
Unit 9.	<b>INFORMATION SYSTEMS AND SOFTWARE ENGINEERING:</b> Information gathering, Requirement and Feasibility Analysis, Data flow diagrams, Process specifications, Input/Output design, Process life cycle, Planning and Managing the project, Design, Coding, Testing, Implementation, Maintenance.
Unit 10.	<b>COMPUTER NETWORKS:</b> ISO/OSI stack, LAN technologies (Ethernet, Token ring), Flow and Error control techniques, Routing algorithms, Congestion control, TCP/UDP and Sockets, IP(v4), Application layer protocols (icmp, dns, smtp, pop, ftp, http), Basic concepts of Hubs, Switches, Gateways, and Routers

<b>Unit 11.</b>	<b>WEB TECHNOLOGIES:</b> Proxy, HTML, XML, Basic concepts of cgi-bin programming.
<b>Unit 12.</b>	<p><b>ENGINEERING MATHEMATICS:</b></p> <p><b>Mathematical Logic:</b> Propositional Logic, First Order Logic.</p> <p><b>Probability:</b> Conditional Probability, Mean, Median, Mode and Standard Deviation, Random Variables, Distributions, Uniform, Normal, Exponential, Poisson, Binomial.</p> <p><b>Set Theory &amp; Algebra:</b> Sets, Relations, Functions, Groups, Partial Orders, Lattice, Boolean Algebra.</p> <p><b>Combinatorics:</b> Permutations, Combinations, Counting, Summation, Generating Functions, Recurrence Relations, Asymptotics.</p> <p><b>Graph Theory:</b> Connectivity, Spanning Trees, Cut Vertices &amp; Edges, Covering, Matching, Independent Sets, Colouring, Planarity, Isomorphism.</p> <p><b>Linear Algebra:</b> Algebra of Matrices, Determinants, Systems of Linear Equations, Eigen Values and Eigen Vectors.</p> <p><b>Numerical Methods:</b> LU Decomposition for Systems of Linear Equations, Numerical Solutions of Non-Linear Algebraic Equations by Secant, Bisection and Newton-Raphson Methods, Numerical Integration by Trapezoidal and Simpson's Rules.</p> <p><b>Calculus:</b> Limit, Continuity &amp; Differentiability, Mean Value Theorems, Theorems of Integral Calculus, Evaluation of Definite &amp; Improper Integrals, Partial Derivatives, Total Derivatives, Maxima &amp; Minima.</p>

## **INFORMATION ABOUT GATE**

### **What is GATE ?**

Graduate Aptitude Test in Engineering (GATE) is an all India examination administered and conducted jointly by the Indian Institute of Science and seven Indian Institutes of Technology on behalf of the National Coordination Board - GATE, Department of Higher Education, Ministry of Human Resource Development (MHRD), Government of India.

The GATE committee, which comprises of representatives from the administering institutes, is the sole authority for regulating the examination and declaring the results.

### **Why Should we Take GATE ?**

To pursue M.Tech program in a leading institute of the country. The benefits of M.Tech are:

- More and better companies are coming for Campus Placement in leading Institutes
- Higher salaries are being offered for M.Tech as compared to B.E.
- M.Tech degree leads to specialization and furthering of interest in a certain area which may lead to Ph.D
- M.Tech degree is a must for those wishing to apply for Faculty/Research positions in educational Institutes/R&D centers.
- Scholarship is paid during M. Tech, so no headache to parents for financial requirements.
- The M.Tech program is a IV semester (24 months) program; so get more time to work out career opportunities
- MOST IMPORTANTLY to get to be a part of any Nationally reputed Educational Institute and enjoy learning and research.

## **GATE QUALIFICATION**

Admission to postgraduate programmes with MHRD and some other government scholarships/ assistantships in engineering colleges/institutes is open to those who qualify through GATE. GATE qualified candidates with Bachelor's degree in Engineering/ Technology/ Architecture or Master's degree in any branch of Science/ Mathematics/ Statistics/ Computer Applications are eligible for admission to Master/Doctoral programmes in Engineering/ Technology/ Architecture as well as for Doctoral programmes in relevant branches of Science with MHRD or other government scholarships/ assistantships. To avail the scholarship, the candidate must secure admission to such a postgraduate programme, as per the prevailing procedure of the admitting institution. However, candidates with Master's degree in Engineering/ Technology/ Architecture may seek admission to relevant Ph.D programmes with scholarship/ assistantship without appearing in the GATE examination.

Some institutions specify GATE qualification as mandatory even for admission of self-financing students to postgraduate programmes. GATE qualified candidates are also eligible for the award of Junior Research Fellowship in CSIR Laboratories and CSIR sponsored projects. Top rank holders in some GATE papers are entitled to apply for "Shyama Prasad Mukherjee Fellowship" awarded by CSIR. Some government organizations prescribe GATE qualification as a requirement for applying to the post of a Scientist/ Engineer

## **ELIGIBILITY FOR GATE**

- Bachelor degree holders in Engineering/ Technology/ Architecture/Pharmacy (4 years after 10+2) and those who are in the final or pre-final year of such programmes.
- Master degree holders in any branch of Science/ Mathematics/ Statistics/ Computer Applications or equivalent and those who are in the final or pre-final year of such programmes.
- Candidates in the second or higher year of the Four-year Integrated Master degree programme (Post-B.Sc.) in Engineering/Technology or in the third or higher year of Five-year Integrated Master degree programme and Dual Degree programme in Engineering/Technology.
- Candidates with qualifications obtained through examinations conducted by professional societies recognised by UPSC/AICTE (e.g. AMIE) as equivalent to B.E./B.Tech. Those who have completed section A or equivalent of such professional courses are also eligible.

## **After the GATE Exam, What Next ?**

1. After declaration of GATE results, student must apply to individual institutes to get their application forms.
2. Institutes advertise for M. Tech admissions in leading newspapers from 1st April till the end of July. However some institutes do not advertise and therefore students have to send request for the applications to the institute directly.
3. Admission in the institute is based on GATE percentile.
4. The concerned institute may conduct written test and/or interview for the purpose of admission.

### **Scholarship**

During the pursuit of M.Tech, you are paid a scholarship of Rs. 5000 to Rs. 9000 per month by the Government of India. This amount is enough for living expenses including purchase of books. The scholarship is paid for entire period of M.Tech., which is 24 months.

Students of III<sup>rd</sup> year of their degree can also appear GATE. Since

- (a) The syllabus prescribed in GATE covers mainly upto V<sup>th</sup> Semester.
- (b) It will be a good try! If it is a bad score you can always try again.

## Structure of GATE and GATE Result

### List of GATE Papers and Corresponding Codes

Paper	Code	Paper	Code
Aerospace Engineering	AE	Instrumentation Engineering	IN
Agricultural Engineering	AG	Mathematics	MA
Architecture and Planning	AR	Mechanical Engineering	ME
Biotechnology	BT	Mining Engineering	MN <sup>s</sup>
Civil Engineering	CE	Metallurgical Engineering	MT
Chemical Engineering	CH	Physics	PH
Computer Science and Information Technology	CS	Production and Industrial Engineering	PI
Chemistry	CY	Textile Engineering and Fibre Science	TF\$
Electronics and Communication Engineering	EC	Engineering Sciences	XE*
Electrical Engineering	EE	Life Sciences	XL*
Geology and Geophysics	GG#		

# GG (Geology and Geophysics) paper will consist of two parts: Part A and Part B. Part A will be common for all candidates. Part B will contain two sections: Section 1 (Geology) and Section 2 (Geophysics). Candidates will have to attempt questions in Part A and either Section 1 or Section 2 in Part B.

\$ GATE 2010 examination for Mining Engineering (MN) and Textile Engineering and Fibre Science (TF) papers will be computer based ONLINE examination.

\* XE (Engineering Sciences) and XL (Life sciences) papers are of general nature and will comprise of the following sections

Engineering Sciences (XE) Paper Section	CODE	Life Science (XL) Paper Section	CODE
Engineering Mathematics (Compulsory)	A	Chemistry (Compulsory)	H
Fluid Mechanics	B	Biochemistry	I
Materials Science	C	Botany	J
Solid Mechanics	D	Microbiology	K
Thermodynamics	E	Zoology	L
Polymer Science and Engineering	F		
Food Technology	G		

A candidate appearing in XE or XL paper will be required to answer three sections apart from the General Aptitude (GA) questions. Section A is compulsory in XE paper and Section H is compulsory in XL paper. The candidate can choose any two out of the remaining sections listed against the respective papers.

The choice of the appropriate paper is the responsibility of the candidate. Some guidelines in this respect are suggested below:

Candidate is expected to appear in a paper (one of the listed above) appropriate to the discipline of his/her qualifying degree.

Candidate is, however, free to choose any paper according to his/her admission plan, keeping in mind the eligibility criteria of the institutions in which he/she wishes to seek admission.

## GATE Results

- (a) GATE score is valid for two years.
- (b) The Score Card of the Qualified Candidates will be given Percentile Score for that discipline and the performance Index. The percentile score in each discipline is calculated as follow:

$$P = \frac{N \sum_{c=1}^r n_c}{N} \times 100$$

where P = Percentile score

N = total number of candidates appearing in the discipline

$n_c$  = number of candidates having the same all india rank in the same discipline

r = all India rank

**Performance index (PI)** in a discipline is calculated as follows:

$$\text{Performance Index} = K_1 + K_2 \frac{(m - a)}{s}$$

where m = marks obtained by the candidate in a paper

a = average marks in the paper

s = standard deviation in the paper

$K_1$  and  $K_2$  = constants which are same for all disciplines

- (d) The GATE evaluation process is called out with utmost care. Requests for revaluation of answer scripts and retotaling of marks will not be entertained.
- (e) The GATE results and particulars of the qualified candidates will be made available to interested organizations (educational institutions, R and D laboratories, industries etc.,) from India and abroad based on written request by the organization and on payment. Details can be obtained from GATE Chairman of IITs/IISc.



# Contents

S.No.	Chapter	Page No.
1	Theory of Computation	
	1.1 Finite State Machine .....	1.1.1-1.1.24
	1.2 Regular Expressions and Languages .....	1.2.1-1.2.22
	1.3 Context Free Languages and Pushdown Automata .....	1.3.1-1.3.22
	1.4 Turing Machine .....	1.4.1-1.4.14
	1.5 Analysis of Algorithm and Computational Complexity .....	1.5.1-1.5.12
2.	Digital Logic	
	2.1 Number System.....	2.1.1-2.1.14
	2.2 Logic Functions and Minimization .....	2.2.1-2.2.38
	2.3 Combinational Circuits.....	2.3.1-2.3.28
	2.4 Sequential Circuits .....	2.4.1-2.4.28
3.	Computer Organization and Architecture	
	3.1 Overview of Computer Architecture .....	3.1.1-3.1.12
	3.2 CPU Organization .....	3.2.1-3.2.26
	3.3 Memory Organization.....	3.3.1-3.3.20
	3.4 Input Output .....	3.4.1-3.4.10
4.	Programming	
	4.1 C-Fundamentals .....	4.1.1-4.1.12
	4.2 Control Statements .....	4.2.1-4.2.28
	4.3 Functions and Pointers.....	4.3.1-4.3.32
	4.4 Arrays .....	4.4.1-4.4.18
	4.5 Structures and Union .....	4.5.1-4.5.10
	4.6 OOPs .....	4.6.1-4.6.10
5.	Data Structures and Algorithms	
	5.1 Array and Queue .....	5.1.1-5.1.14
	5.2 Tree and Binary Search Tree .....	5.2.1-5.2.34
	5.3 Sorting and Searching .....	5.3.1-5.3.34
	5.4 Stack and NP complete .....	5.4.1-5.4.10

<b>6.</b>	<b>Compiler Design</b>	
6.1	Introduction to Compiler .....	6.1.1-6.1.10
6.2	Lexical Analysis.....	6.2.1-6.2.8
6.3	Parsing Techniques .....	6.3.1-6.3.18
6.4	Syntax Directed Translation .....	6.4.1-6.4.8
6.5	Code Generation And Optimization .....	6.5.1-6.5.12
<b>7.</b>	<b>Operating System</b>	
7.1	Process, IPC and Deadlock.....	7.1.1-7.1.28
7.2	File System .....	7.2.1-7.2.12
7.3	Memory Management .....	7.3.1-7.3.18
7.4	Input Output and Disk Scheduling .....	7.4.1-7.4.12
7.5	Protection and Security .....	7.5.1-7.5.10
<b>8.</b>	<b>Databases</b>	
8.1	Introduction to DBMS and Relational Model.....	8.1.1-8.1.20
8.2	Normalization .....	8.2.1-8.2.14
8.3	Structured Query Language.....	8.3.1-8.3.16
8.4	File Organization Techniques.....	8.4.1-8.4.10
8.5	Transaction Management .....	8.5.1-8.5.10
<b>9.</b>	<b>Information Systems and Software Engineering</b>	
9.1	Software Process Models.....	9.1.1-9.1.8
9.2	Software Metrics & Project Estimation .....	9.2.1-9.2.8
9.3	Project Planning and Management .....	9.3.1-9.3.10
9.4	Software Design.....	9.4.1-9.4.6
9.5	Software Testing .....	9.5.1-9.5.6
<b>10.</b>	<b>Computer Networks</b>	
10.1	Introduction to ISO/OSI Model & Physical Layer.....	10.1.1-10.1.14
10.2	The Data Link Layer .....	10.2.1-10.2.16
10.3	Network Layer .....	10.3.1-10.3.12
10.4	Transport Layer .....	10.4.1-10.4.16
10.5	Session, Presentation and Application Layer .....	10.5.1-10.5.10

<b>11.</b>	<b>Web Technology</b>	
11.1	HTML .....	11.1.1-11.1.6
11.2	XML .....	11.2.1-11.2.6
11.3	ASP .....	11.3.1-11.3.4
11.4	.NET .....	11.4.1-11.4.4
11.5	Software Process Models .....	11.5.1-11.5.8
<b>12.</b>	<b>Engineering Mathematics</b>	
12.1	Mathematical Logic .....	12.1.1-12.1.18
12.2	Probability .....	12.2.1-12.2.34
12.3	Set Theory And Algebra .....	12.3.1-12.3.40
12.4	Combinatorics .....	12.4.1-12.4.18
12.5	Graph Theory .....	12.5.1-12.5.22
12.6	Linear Algebra .....	12.6.1-12.6.36
12.7	Numerical Methods .....	12.7.1-12.7.18
12.8	Calculus .....	12.8.1-12.8.42
<b>13.</b>	<b>General Aptitude</b>	
13.1	Verbal Ability .....	13.1.1-13.1.16
13.2	Numerical Ability .....	13.2.1-13.2.32
<b>GP</b>	<b>Gate Paper 2010 .....</b>	<b>GP.1-GP.24</b>

		estimated	actual
6.1.10-1.1.11		120.00	121
6.2.11-2.3.11		180.0	171
4.2.10-1.5.11		98.0	91
6.3.11-1.4.12		120.0	111
8.2.11-1.3.12		100.00	91
		estimated	actual
8.1.11-1.1.12		100.00	101
8.2.11-1.2.12		100.00	101
9.2.11-1.3.12		100.00	93
10.2.11-1.4.12		100.00	93
11.2.11-1.5.12		100.00	93
12.2.11-1.6.12		100.00	93
13.2.11-1.7.12		100.00	93
		estimated	actual
8.1.12-1.1.13		100.00	101
8.2.12-1.2.13		100.00	101
9.2.12-1.3.13		100.00	101
10.2.12-1.4.13		100.00	101
11.2.12-1.5.13		100.00	101
12.2.12-1.6.13		100.00	101
13.2.12-1.7.13		100.00	101

# **UNIT-1**

# **THEORY OF**

# **COMPUTATION**

Unit 5

# THEORY OF COMPUTATION



# FINITE STATE MACHINE

**1.1**

## LEVEL-1

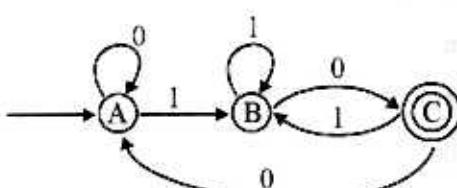
**Q.1** The language accepted by Finite Automata is

- (a) Non-regular languages
- (b) Strings
- (c) Regular languages
- (d) None of these

**Q.2** Select the most appropriate combination

- (a) DFA, NFA, Mealy
- (b) DFA, Moore, Mealy
- (c) DFA, NFA, Moore
- (d) None of these

**Q.3** Following Finite Automata recognizes the languages



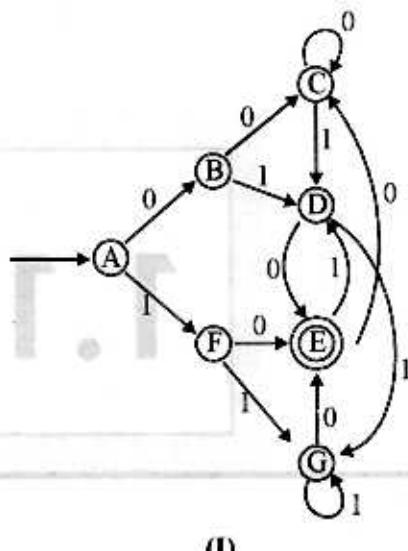
- (a)  $\{0, 1\}^* \{1, 0\}$
- (b)  $0^* 1^* \{1, 0\}$
- (c)  $\{10\}^* 0$
- (d) None of these

**Q.4** Match the following:

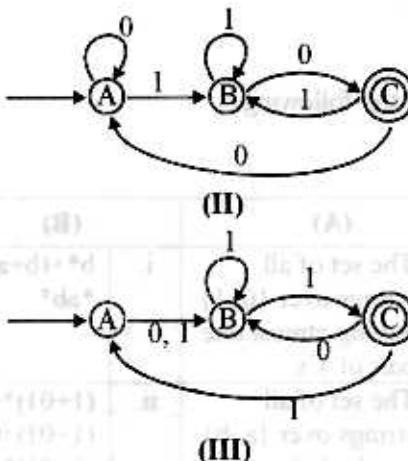
(A)	(B)
1. The set of all strings over $\{0, 1\}$ having atmost one pair of 1's	i. $b^* + (b+abb)^*$ $*ab^*$
2. The set of all strings over $\{a, b\}$ in which there are at least two occurrences of b between any two occurrences of a.	ii. $(1+01)^* +$ $(1+01)^* 00$ $(1+01)^* +$ $(0+10)^* +$ $(0+10)^* 11$ $(0+10)^*$
3. The set of all strings over $\{0, 1\}$ beginning with 00.	iii. $00(0+1)^*$
	iv. $*ab^+ (b+ab)^* + ab$

- (a) 1 - iii, 2 - iv, 3 - i
- (b) 1 - i, 2 - ii, 3 - iii
- (c) 1 - ii, 2 - i, 3 - iii
- (d) None of the above

- Q.5** Which of the following FA recognizes the languages  $\{0, 1\}^* \setminus \{10\}$

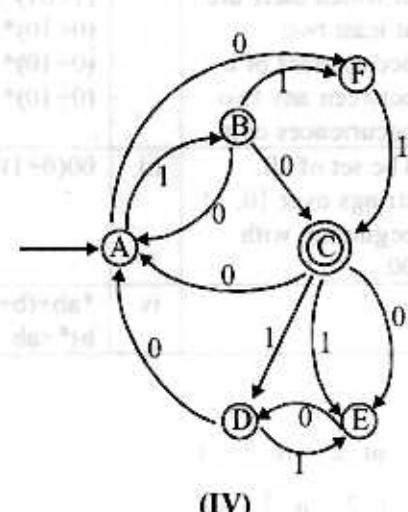


(I)



(II)

(A)



(III)

- (a) I, II  
 (b) I, IV  
 (c) II, III  
 (d) III, IV

- Q.6** Consider the following transition table of an FA.

$\delta$	a	b
start	$q_1$	$q_0$
$q_0$	$q_1$	$q_0$
$q_1$	$q_2$	$q_1$
$q_2$	$q_3$	$q_2$
$q_3$	$q_4$	$q_3$
$q_4$	$q_4$	$q_4$

What is true for the given FA?

- (a) Accept strings containing even no. of a's and b's.  
 (b) Does not accept strings containing b's  
 (c) Accept strings independent of the no. of b's  
 (d) Both (a) and (b)

- Q.7** Consider the transition table as given below:

$\delta$	0	1	2
(A)	A	B	C
(B)	-	B	C
(C)	-	-	C

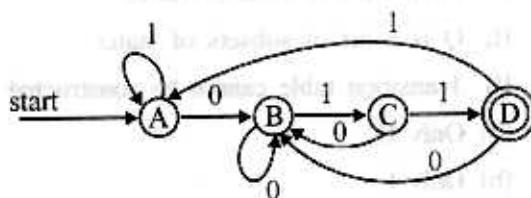
If A becomes an non-accepting state, then how many strings ending with 0 will be accepted. (Maximum length of the string is n)

- (a)  $n + 1$   
 (b) n  
 (c) 1  
 (d) 0

- Q.8** Can DFA simulate NFA?

- (a) Depends on NFA  
 (b) Sometimes  
 (c) No  
 (d) Yes

**Q.9** The figure shown is a DFA, M.



Which of the following regular expression denotes the set of all words accepted by M.

- (a)  $(0|1) 0^* 1^* 0$
- (b)  $1^* 0^* 1 1$
- (c)  $(0|1)^* 011$
- (d)  $1^* 0^* 001$

**Q.10** The number of states of the FSM, required to simulate the behaviour of a computer, with a memory capable of storing 'm' words, each of length 'n' bits is

- (a)  $2^{m+n}$
- (b)  $2^{mn}$
- (c)  $m \times 2^n$
- (d) None of these

**Q.11** An FSM can be considered to be a TM

- (a) of finite tape length, rewinding capability and bidirectional tape movement
- (b) of finite tape length, without rewinding and unidirectional tape movement
- (c) of finite tape length, rewinding capability and unidirectional tape movement
- (d) none of the above

**Q.12** Which of the following statements about push down machines is true.

- (a) The class of FSM's with two pushdown stacks has the same power as the class of post or turing machines.
- (b) For  $n \leq 3$ , the class of FSM's with n push down stacks has the same power as the class of FSM's with two pushdown stacks.
- (c) The class of FSM's with one pushdown stack is more powerful than FSM.
- (d) All of the above

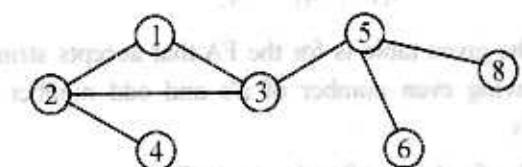
**Q.13** Context sensitive grammar can be recognized by a

- (a) Linearly bounded automaton
- (b) Finite state machine (FSM)
- (c) Non-deterministic push down machine (NDPDM)
- (d) Deterministic push down machine (DPDM)

**Q.14** Which of the following recognizes variable prefixes of the grammar?

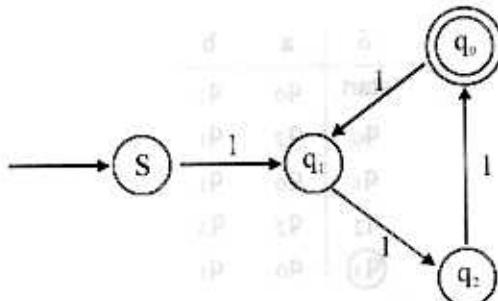
- (a) NFA
- (b) DFA
- (c) Neither NFA nor DFA
- (d) Both NFA and DFA

**Q.15** The number of articulation points of the following graph is



- (a) 3
- (b) 2
- (c) 1
- (d) 0

**Q.16** The transition diagram represents



- (a) FSM to check whether a given unary number is divisible by 3
- (b) FSM to check whether a given binary number is divisible by 3
- (c) both (a) and (b)
- (d) None of the above

**Q.17** Find the odd man out.

- (a) DFA
- (b) Mealy machine
- (c) Moore machine
- (d) NFA

## LEVEL-2

**Q.18** Consider the transition table given below:

$\delta$	a	b
start	$q_2$	$q_1$
$q_0$	$q_2$	$q_1$
$q_1$	$q_3$	$q_0$
$q_2$	$q_0$	$q_3$
$q_3$	$q_1$	$q_2$

The given table is for the FA that accepts strings having even number of a's and odd number of b's.

The final state for the given FA is

- (a)  $q_2, q_3$
- (b)  $q_1$
- (c)  $q_3$
- (d) can't be determined

**Q.19** Consider the following construction of a FA that accepts a string ending with aab for  $\Sigma = \{a, b\}$ .

$\delta$	a	b
start	$q_0$	$q_1$
$q_0$	$q_2$	$q_1$
$q_1$	$q_0$	$q_1$
$q_2$	$q_2$	$q_3$
$q_3$	$q_0$	$q_1$

where  $q_3$  is the final state.

Which two states in combination denotes all string ending in b?

- (a) cannot be determined
- (b) Only  $q_1$  is sufficient
- (c) Only  $q_3$  is sufficient
- (d)  $q_1, q_3$

**Q.20** Which of the following is true for an NFA?

- I. Final state is a set of states
- II. Q is a set of subsets of states
- III. Transition table cannot be constructed
- (a) Only II
- (b) Only I
- (c) Only III
- (d) All of above

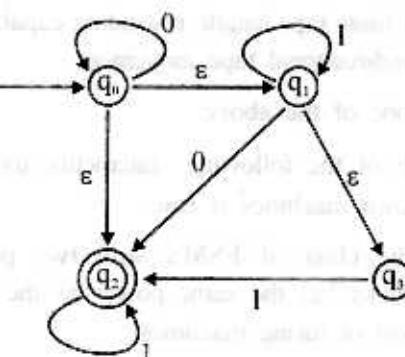
**Q.21** Consider the following transition table for a FA that accepts strings containing substring aba.

$\delta$	a	b
start	$q_0$	$q_1$
$q_0$	$q_0$	$q_2$
$q_1$	$q_0$	A
$q_2$	$q_3$	$q_1$
$q_3$	$q_3$	$q_3$

What is the state that A represents?

- (a)  $q_3, q_1$
- (b)  $q_0$
- (c)  $q_2$
- (d)  $q_1$

**Q.22** After converting following NFA into DFA, the states in the final DFA will be



- (a)  $[q_1, q_2, q_3, q_0], [q_1, q_2, q_3], [q_2], [ ]$
- (b)  $[q_1, q_2, q_3], [q_1, q_2, q_3, q_0], [q_0, q_2, q_3], [ ]$
- (c)  $[q_1, q_2, q_3, q_0], [q_1, q_2, q_3], [ ]$
- (d)  $[q_1, q_2, q_3, q_0], [q_2], [q_1, q_2, q_3]$

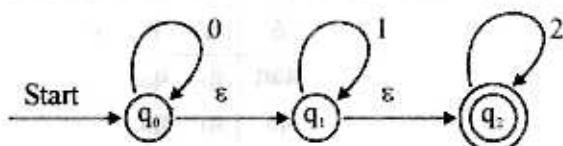
**Q.23** Consider the transition table of a DFA as given below:

$\delta$	a	b
start	$q_0$	$q_4$
$q_0$	$q_0$	$q_1$
$q_1$	$q_0$	$q_2$
$q_2$	$q_0$	$q_2$
$q_3$	$q_3$	$q_4$
$q_4$	$q_3$	$q_4$

Which of the following is the most precise interpretation of state  $q_3$ ?

- (a) Accepts strings starting with a and ending with bb
- (b) Accepts strings starting with b and ending with a
- (c) Accepts strings starting with a and ending with ab
- (d) Accepts strings starting with a and ending with b

**Q.24** Consider the following NFA with  $\epsilon$  moves



Which of the following strings will be accepted by the given NFA?

- I. 001122
- II. 1122
- III. 22
- (a) None of the above
- (b) I, II and III
- (c) Only I
- (d) II and III both

**Q.25** Which of the following are extended transition functions of a DFA and NFA respectively?

I.  $\hat{\delta}(q, \epsilon) = q$

$$\hat{\delta}(q, ya) = \bigcup_{r \in \hat{\delta}(q, y)} \delta(r, a)$$

II.  $\hat{\delta}(q, \epsilon) = q$

$$\hat{\delta}(q, ya) = \delta(\hat{\delta}(q, y), a)$$

III.  $\hat{\delta}(q, \epsilon) = \{q\}$

$$\hat{\delta}(q, ya) = \bigcup_{r \in \hat{\delta}(q, y)} \delta(r, a)$$

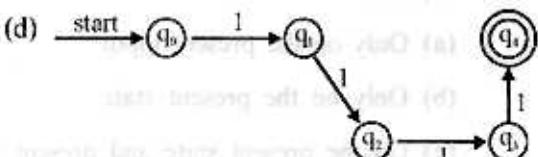
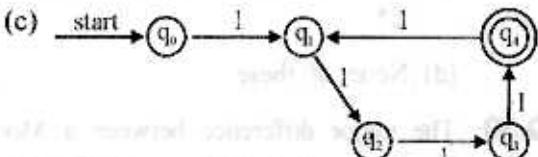
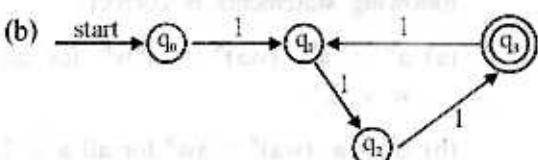
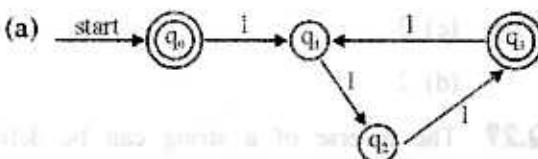
(a) II  $\rightarrow$  NFA, III  $\rightarrow$  DFA

(b) II  $\rightarrow$  DFA, III  $\rightarrow$  NFA

(c) I  $\rightarrow$  NFA, II  $\rightarrow$  DFA

(d) I  $\rightarrow$  DFA, II  $\rightarrow$  NFA

**Q.26** Design a FSM to check whether a given unary number is divisible by 3.



- Q.27** The transition table given below is for the FSM that accepts a string if it ends with 'aa'.

$\delta$	a	b
start	$q_0$	$q_2$
$q_0$	$q_1$	$q_2$
$q_1$	$q_1$	$q_2$
$q_2$	$q_0$	$q_2$

Which is the final state?

- (a)  $q_2$
- (b)  $q_0$
- (c)  $q_1$
- (d) Can't be determined

- Q.28** Let  $\Sigma = \{0, 1\}$ , then an automaton A accepting only those words from  $\Sigma$  having an odd number of 1's requires \_\_\_\_\_ states including the start state.

- (a) 5
- (b) 4
- (c) 3
- (d) 2

- Q.29** The reverse of a string can be defined more precisely by recursive rules. Which of the following statements is correct?

- (a)  $a^R = a^R$ ,  $(wa)^R = a^R w^R$  for all  $a \in \Sigma$ ,  $w \in \Sigma^*$
- (b)  $a^R = a$ ,  $(wa)^R = aw^R$  for all  $a \in \Sigma$ ,  $w \in \Sigma^*$
- (c)  $a^R = a^R$ ,  $(wa)^R = (aw)^R$  for all  $a \in \Sigma$ ,  $w \in \Sigma^*$
- (d) None of these

- Q.30** The major difference between a Moore and a Mealy machine is that output of the former depends

- (a) Only on the present input
- (b) Only on the present state
- (c) On the present state and present input
- (d) None of these

- Q.31** Consider the given FSMs

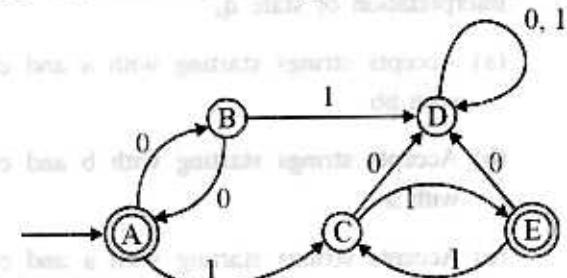


- (a) The second FSM accepts  $\epsilon$  only
- (b) The first FSM accepts nothing
- (c) Both are equivalent
- (d) None of the above

- Q.32** Basic machine will have

- (a) Only input and output states
- (b) Internal states
- (c) Memory
- (d) None of above

- Q.33** The finite automata given below accepts:



- (a)  $\{01\}^* \{10\}^*$
- (b)  $\{00\}^* \{11\}^*$
- (c)  $\{00\}^* \{111\}^*$
- (d)  $\{1\}^* \{0\}^*$

- Q.34** Consider the following transition table of an FA

$\delta$	a	b
start	$q_1$	$q_0$
$q_0$	$q_1$	$q_0$
$q_1$	$q_2$	$q_1$
$q_2$	$q_3$	$q_2$
$q_3$	$q_4$	$q_3$
$q_4$	$q_4$	$q_4$

If the final state is  $q_3$ , which of the following strings will be accepted?

- I. ababab
- II. aaabbb
- III. abbabba

  - (a) I
  - (b) II
  - (c) III
  - (d) All of the above

**Q.35** Consider the transition table as given below:

$\delta$	0	1	2
(A)	A	B	C
(B)	-	B	C
(C)	-	-	C

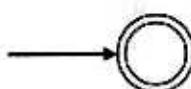
How many strings ending with 0 will be accepted by the given DFA if the maximum possible length of the string is

- (a)  $2n$
- (b)  $n^2$
- (c) 0
- (d)  $n$

**Q.36** Which of the following is not true of a DFA?

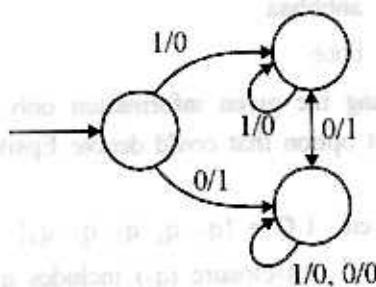
- (a) In any state, it has a transition for each input symbol.
- (b) Initial state and final state can be same.
- (c) There can be more than one accepting state.
- (d) Has a transition from any state, on any input symbol, to exactly one state.

**Q.37** The FSM shown in the figure recognize



- (a)  $\epsilon$ -alone
- (b) no string
- (c) all strings
- (d) none of these

**Q.38** For the machine shown in the figure



- (a) changes the sign bit
- (b) increments a given bit pattern by 1
- (c) finds 2's complement of a given bit pattern
- (d) complements a given bit pattern

**Q.39** Consider the following transition table of an FA.

$\delta$	a	b
start	$q_1$	$q_0$
$q_0$	$q_1$	$q_0$
$q_1$	$q_2$	$q_1$
$q_2$	$q_3$	$q_2$
$q_3$	$q_4$	$q_3$
$q_4$	$q_4$	$q_4$

If the final state is  $q_4$ , then which of the following strings will be accepted?

- I. aaaaaa
- II. aabbaabbba
- III. bbabababbb
- (a) II and III
- (b) III and I
- (c) I and II
- (d) None of these

**Q.40** 6 files F1, F2, F3, F4, F5 and F6 have 100, 200, 50, 80, 120, 150 records respectively.

In what order should they be stored so as to optimize activity? Assume each file is accessed with the same frequency.

- (a) ordering is immaterial as all files are accessed with the same frequency.
- (b) F1, F2, F3, F4, F5, F6
- (c) F2, F6, F5, F1, F4, F3
- (d) F3, F4, F1, F5, F6, F2

**Q.41** 6 files F1, F2, F3, F4, F5 and F6 have 100, 200, 50, 80, 120, 150 records respectively.

The average access time will be

- (a) 210 unit
- (b) 293 units
- (c) 256 units
- (d) 268 units

## LEVEL-3

- Q.42** Let  $\Sigma = \{a, b\}$ . Then an automaton A accepting only those words from  $\Sigma$  having an even number of a's requires \_\_\_\_\_ states including start state.
- 3
  - 15
  - 6
  - 4

**Common Data For Questions 43 & 44:**

Consider the following construction of FA that accepts a string ending with aab for  $\Sigma = \{a, b\}$ .

$\delta$	a	b
start	$q_0$	$q_1$
$q_0$	$q_2$	$q_1$
$q_1$	$q_0$	$q_1$
$q_2$	$q_2$	$q_3$
$q_3$	$q_0$	$q_1$

where  $q_3$  is the final state.

- Q.43** What will be the new entries for rows  $q_2$  and  $q_3$  if we want the given automaton to accept strings containing substring aab?
- $q_2 = (q_2/a, q_3/b)$   
 $q_3 = (q_3/a, q_3/b)$
  - $q_2 = (q_1/b, q_2/b)$   
 $q_3 = (q_3/a, q_3/a)$
  - $q_2 = (q_3/b, q_2/b)$   
 $q_3 = (q_2/a, q_3/a)$
  - No changes required

- Q.44** Which state denotes the string ending in aa
- $q_1$
  - $q_3$
  - $q_2$
  - All of the above

- Q.45** Which of the following represents a Moore machine to determine the residue mod 3 for any binary number i.e.  $\Sigma = \{0, 1\}$ ,  $\Delta = \{0, 1, 2\}$ .

(a) $\delta$	0	1	o/p
start	$q_0$	$q_1$	-
$q_0$	$q_2$	$q_1$	0
$q_1$	$q_2$	$q_0$	1
$q_2$	$q_1$	$q_2$	1

(b) $\delta$	0	1	o/p
start	$q_0$	$q_1$	-
$q_0$	$q_1$	$q_2$	0
$q_1$	$q_2$	$q_0$	1
$q_2$	$q_1$	$q_2$	2

- (c) both (a) and (b)

- (d) none of these

- Q.46** Consider the following transition table for a FA that accepts strings containing substring aba.

$\delta$	a	b
start	$q_0$	$q_1$
$q_0$	$q_0$	$q_2$
$q_1$	$q_0$	A
$q_2$	$q_3$	$q_1$
$q_3$	$q_3$	$q_3$

If we replace A by  $q_0$ , which of the following strings will be accepted?

- (a) abbbbba

- (b) bbbbbba

- (c) abbbbaa

- (d) bbbb

- Q.47** Using the given information only which is the best option that could denote Epsilon-closure of  $q_0$

Given : 1.  $Q = \{q_0, q_1, q_2, q_3, q_4\}$

2.  $\epsilon$ -closure ( $q_2$ ) includes  $q_4$

- $\{q_0, q_1, q_2\}$
- $\{q_0, q_2, q_4\}$
- $\{q_1, q_2, q_2\}$
- $\{q_0, q_2, q_4, q_5\}$

**Q.48** Consider the transition table given below:

$\delta$	a	b
start	$q_2$	$q_1$
$q_0$	$q_2$	$q_1$
$q_1$	$q_3$	$q_0$
$q_2$	$q_0$	$q_3$
$q_3$	$q_1$	$q_2$

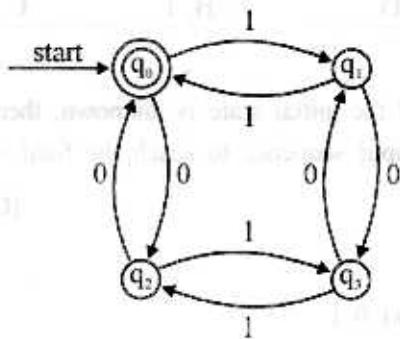
The given table is for FA that accepts strings having

- odd number of a's and even number of b's
- odd number of a's and odd number of b's.

Find the final states for both cases.

- (a)  $q_2, q_3$
- (b)  $q_3, q_1$
- (c)  $q_1, q_2$
- (d)  $q_1, q_3$

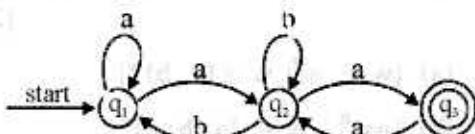
**Q.49** Consider the transition diagram of a DFA as given below



Which is the language of the given DFA?

- (a)  $L = \{w \mid w \text{ has equal no. of ones and zeros}\}$
- (b)  $L = \{\}$
- (c)  $L = \{\epsilon\}$
- (d) None of these

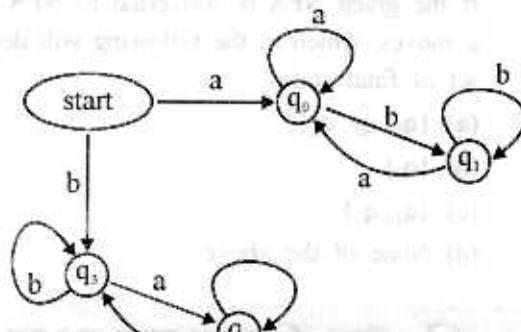
**Q.50** Consider the transition system below, find out the strings recognized by the transition system.



- (a)  $(a+a(b+aa)b)a^*(b+a)a^*$
- (b)  $(a+a(b+aa)^*b)a(b+a^*)^*$
- (c) both (a) & (b)
- (d) None of the above

**Q.51** Consider the transition diagram of an DFA as given below:

Which should be the final state(s) of the DFA if it should accept strings starting with 'a' and ending with 'b'.



- (a)  $q_0, q_1$
- (b)  $q_3$
- (c)  $q_1$
- (d)  $q_0$

**Q.52** I. FSM has the capacity to remember arbitrarily large amount of information

II. FSM can multiply two given arbitrarily long numbers

III. FSM cannot recognize a palindrome

Which of the above statements are not true in case of FSM?

- (a) III
- (b) I, II
- (c) II, III
- (d) I, III

**Q.53** What is the language of the given DFA?

Interpretation	$\delta$	a	b
start end	start	$q_0$	$q_4$
a a	$q_0$	$q_0$	$q_1$
a b	$q_1$	$q_0$	$q_2$
a bb	$q_2$	$q_0$	$q_2$
b a	$q_3$	$q_3$	$q_4$
b b	$q_4$	$q_3$	$q_4$

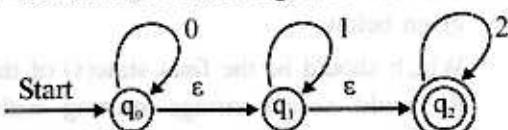
(a)  $L = \{w \mid w \text{ starts with } b \text{ and ends with } a \text{ or } w \text{ starts with } a \text{ and ends with } b\}$

(b)  $L = \{w \mid w \text{ starts with } a \text{ and ends with } bb \text{ or } w \text{ starts with } b \text{ and ends with } a\}$

(c)  $L = \{w \mid w \text{ starts with } b \text{ and ends with } b \text{ or } w \text{ starts with } a \text{ and ends with } a\}$

(d) both (a) and (b)

**Q.54** Consider the following NFA with  $\epsilon$  moves



If the given NFA is converted to DFA without  $\epsilon$ -moves, which of the following will denote the set of final states?

- (a)  $\{q_0, q_1, q_2\}$
- (b)  $\{q_0\}$
- (c)  $\{q_1, q_2\}$
- (d) None of the above

## GATE QUESTIONS

**Q.55** Which one of the following is the strongest correct statement about a finite language over some finite alphabet  $\Sigma$ ? [GATE 1991]

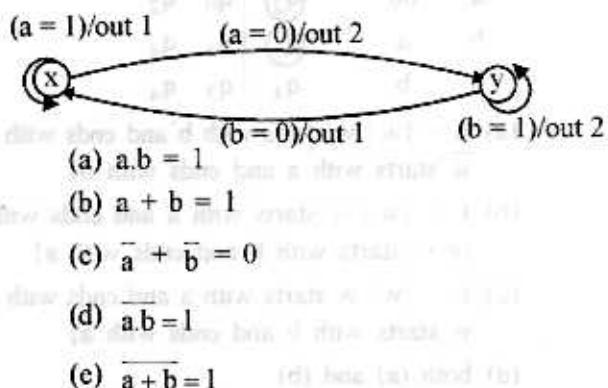
- (a) It could be undecidable
- (b) It is Turing machine recognizable
- (c) It is a context sensitive language
- (d) It is a regular language

**Q.56** In which of the cases stated below is the following statement true? [GATE 1992]

"For every nondeterministic machine  $M_1$ , there exists an equivalent deterministic machine  $M_2$  recognizing the same language."

- (a)  $M_1$  is nondeterministic finite automaton
- (b)  $M_1$  is a nondeterministic PDA
- (c)  $M_1$  is a nondeterministic Turing machine
- (d) For no machine  $M_1$  use the above statement True

**Q.57** If the state machine described in the figure should have a stable state, then restriction on the input is given by : [GATE 1993]



- (a)  $a \cdot b = 1$
- (b)  $a + b = 1$
- (c)  $\bar{a} + \bar{b} = 0$
- (d)  $\bar{a} \cdot \bar{b} = 1$
- (e)  $\bar{a} + \bar{b} = 1$

**Q.58** Which of the following conversion is not possible (algorithmically)? [GATE 1994]

[1-Mark]

- (a) Regular grammar to context-free grammar
- (b) Non-deterministic FSA to deterministic FSA
- (c) Non-deterministic PDA to deterministic PDA
- (d) Non-deterministic Turing machine to deterministic Turing machine

**Q.59** A finite state machine with the following state table has a single input  $x$  and a single output  $z$ .

Present state	next state, $z$
	$x = 1$
A	D, 0
B	B, 1
C	B, 0
D	B, 1

If the initial state is unknown, then the shortest input sequence to reach the final state C is

[GATE 1995]

[2-Marks]

- (a) 0 1
- (b) 1 0
- (c) 1 0 1
- (d) 1 1 0

**Q.60** Which of the following language over  $\{a, b, c\}$  is accepted by a deterministic pushdown automata? [GATE 1997]

[2-Marks]

- (a)  $\{w \subset w^R \mid w \in \{a, b\}^*\}$
- (b)  $\{ww^R \mid w \in \{a, b, c\}^*\}$
- (c)  $\{a^n b^n c^n \mid n \geq 0\}$
- (d)  $\{w \mid w \text{ is a palindrome over } \{a, b, c\}\}$

Note:  $w^R$  is the string obtained by reversing 'w'.

**Q.61** Which of the following set can be recognized by a Deterministic Finite-state Automaton?

[GATE 1998]

[1-Mark]

- (a) The numbers 1, 2, 4, 8, ...,  $2^n$ , ... written in binary
- (b) The numbers 1, 2, 4, ...,  $2^n$ , ... written in unary
- (c) The set of binary string in which the number of zeros is the same as the number of ones
- (d) The set {1, 101, 11011, 1110111, ...}

**Q.62** The string 1 1 0 1 does not belong to the set represented by [GATE 1998]

[1-Mark]

- (a)  $110^* (0 + 1)$
- (b)  $1 (0 + 1)^* 101$
- (c)  $(10)^* (01)^* (00 + 11)^*$
- (d)  $(00 + (11)^* 0)^*$

**Q.63** Let L be the set of all binary strings whose last two symbols are the same. The number of states in the minimum state deterministic finite state automaton accepting L is [GATE 1998]

[2-Marks]

- (a) 2
- (b) 5
- (c) 8
- (d) 3

**Q.64** Regarding the power of recognition of languages, which of the following statements is false?

[GATE 1998]

[1-Mark]

- (a) The non-deterministic finite-state automata are equivalent to deterministic finite-state automata.
- (b) Non-deterministic Push-down automata are equivalent to deterministic Push-down automata.
- (c) Non-deterministic Turing machines are equivalent to deterministic Pushdown automata.
- (d) Non-deterministic Turing machines are equivalent to deterministic Turing machine.
- (e) Multi-tape Turing machines are equivalent to Single-tape Turing machines.

**Q.65** Consider a DFA over  $\Sigma = \{a, b\}$  accepting all strings which have number of a's divisible by 6 and number of b's divisible by 8. What is the minimum number of states that the DFA will have? [GATE 2001]

[2-Marks]

- (a) 8
- (b) 14
- (c) 15
- (d) 48

**Q.66** Given an arbitrary non-deterministic finite automata (NFA) with N states, the maximum number of states in an equivalent minimized DFA is at least. [GATE 2001]

[1-Mark]

- (a)  $N^2$
- (b)  $2^N$
- (c)  $2N$
- (d)  $N!$

**Q.67** The smallest finite automaton which accepts the language {x | length of x is divisible by 3} has

[GATE 2002]

[2-Marks]

- (a) 2 states
- (b) 3 states
- (c) 4 states
- (d) 5 states

**Q.68** The language accepted by a Pushdown Automaton in which the stack is limited to 10 items is best described as [GATE 2002]

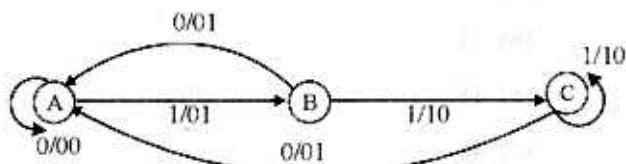
[1-Mark]

- (a) Context free
- (b) Regular
- (c) Deterministic Context free
- (d) Recursive

**Q.69** The Finite state machine described by the following state diagram with A as starting state, where an arc label is  $x/y$  and x stands for 1-bit input and y stands for 2-bit output.

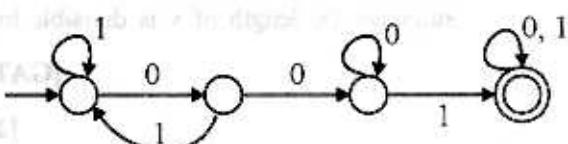
[GATE 2002]

[2-Marks]



- (a) Outputs the sum of the present and the previous bits of the input
- (b) Outputs 01 whenever the input sequence contain 11
- (c) Outputs 00 whenever the input sequence contains 10
- (d) None of the above

**Q.70** Consider the following deterministic finite state automaton M.



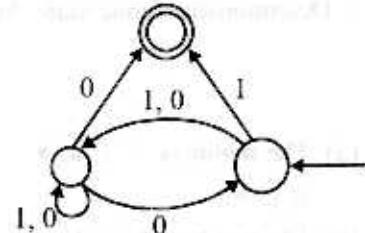
Let S denote the set of seven bit binary strings in which the first, the fourth, and the last bits are 1. The number of strings in S that are accepted by M is

[GATE 2003]

[2-Marks]

- (a) 1
- (b) 5
- (c) 7
- (d) 8

**Q.71** Consider the NFA M shown below:



Let the language accepted by M be L. Let  $L_1$  be the language accepted by the NFA  $M_1$ , obtained by changing the accepting state of M to a non-accepting state and by changing the non-accepting state of M to accepting states. Which of the following statements is true?

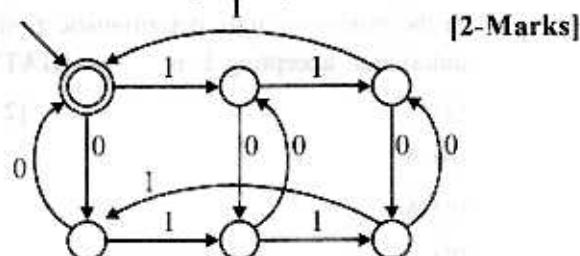
[GATE 2003]

- (a)  $L_1 = \{0, 1\}^* - L$
- (b)  $L_1 = \{0, 1\}^*$
- (c)  $L_1 \subseteq L$
- (d)  $L_1 = L$

**Q.72** The following finite state machine accepts all those binary strings in which the number of 1's and 0's are respectively

[GATE 2004]

[2-Marks]



- (a) divisible by 3 and 2
- (b) odd and even
- (c) even and odd
- (d) divisible by 2 and 3

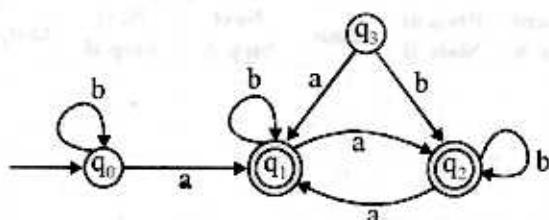
**Q.73** Let  $N_f$  and  $N_p$  denote the classes of languages accepted by non-deterministic finite automata and non-deterministic pushdown automata, respectively. Let  $D_f$  and  $D_p$  denote the classes of languages accepted by deterministic finite automata and deterministic pushdown automata, respectively. Which one of the following is TRUE?

[GATE 2005]

[2-Marks]

- (a)  $D_f \subset N_f$  and  $D_p \subset N_p$
- (b)  $D_f \subset N_f$  and  $D_p = N_p$
- (c)  $D_f = N_f$  and  $D_p = N_p$
- (d)  $D_f = N_f$  and  $D_p \subset N_p$

**Q.74** Consider the following Finite State Automaton:



The minimum state automaton equivalent to the above FSA has the following number of states

[GATE 2007]

[2-Marks]

- (a) 1
- (b) 2
- (c) 3
- (d) 4

**Q.75** A minimum state deterministic finite automaton accepting the language  $L = \{w \mid w \in \{0, 1\}^*, \text{number of } 0's \text{ and } 1's \text{ in } w \text{ are divisible by } 3 \text{ and } 5, \text{ respectively}\}$  has

[GATE 2007]

[2-Marks]

- (a) 15 states
- (b) 11 states
- (c) 10 states
- (d) 9 states

**Q.76** Which of the following statements is false?

[GATE 2008]

[2-Marks]

- (a) Every NFA can be converted to an equivalent DFA
- (b) Every non-deterministic Turing machine can be converted to an equivalent deterministic Turing machine
- (c) Every regular language is also a context-free language
- (d) Every subset of a recursively enumerable set is recursive

**Q.77** Given below are two finite state automata ( $\rightarrow$  indicates the start state and F indicates a final state)

Y:		a	b
	$\rightarrow 1$	1	2
	2(F)	2	1

Z:		a	b
	$\rightarrow 1$	2	2
	2(F)	1	1

Which of the following represents the product automaton  $Z \times Y$ ?

[GATE 2008]

[2-Marks]

		a	b
	$\rightarrow P$	S	R
(a)	Q	R	S
	R(F)	Q	P
	S	Q	P

		a	b
	$\rightarrow P$	S	Q
(b)	Q	R	S
	R(F)	Q	P
	S	P	Q

		a	b
	$\rightarrow P$	Q	S
(c)	Q	R	S
	R(F)	Q	P
	S	Q	P

		a	b
	$\rightarrow P$	S	Q
(d)	Q	S	R
	R(F)	Q	P
	S	Q	P

**Q.78** Which one of the following is FALSE?

[GATE 2009]

[1-Mark]

- (a) There is a unique minimal DFA for every regular language
- (b) Every NFA can be converted to an equivalent PDA
- (c) Complement of every context-free language is recursive
- (d) Every nondeterministic PDA can be converted to an equivalent deterministic PDA

**Q.79** Given the following state table of an FSM with two states A and B, one input and one output:

Present State A	Present State B	Input	Next Step A	Next Step B	Output
0	0	0	0	0	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	1	0	0
0	0	1	0	1	0
0	1	1	0	0	1
1	0	1	0	1	1
1	1	1	0	0	1

If the initial state is A = 0, B = 0, what is the minimum length of an input string which will take the machine to the state A = 0, B = 1 with output = 1?

[GATE 2009]

- (a) 3
- (b) 4
- (c) 5
- (d) 6

## ANSWER KEY

1	c	2	b	3	a	4	c	5	a
6	c	7	d	8	d	9	c	10	b
11	b	12	d	13	a	14	c	15	a
16	a	17	d	18	b	19	d	20	b
21	d	22	a	23	b	24	b	25	b
26	a	27	c	28	c	29	b	30	b
31	d	32	a	33	b	34	d	35	d
36	a	37	a	38	b	39	c	40	d
41	d	42	a	43	a	44	c	45	d
46	b	47	b	48	a	49	d	50	d
51	c	52	b	53	d	54	a	55	d
56	a, c	57	b	58	c	59	b	60	a, c
61	d	62	c, d	63	b	64	b	65	a
66	b	67	b	68	c	69	a	70	c
71	b	72	a	73	d	74	b	75	a
76	d	77	a	78	d	79	a		

## SOLUTIONS

**S.1 (c)**

A language is said to be regular if it is accepted by a finite automaton. That is, **the language accepted by a finite automaton is a regular language.**

**S.2 (b)**

Here, NFA is an odd term as all other have definite transition to a unique state on a given input symbol.

**S.3 (a)**

The transition diagram for an Finite Automata recognizes languages  $\{0, 1\}^* \setminus \{10\}$ .

**S.4 (c)**

2. Let  $W$  be given set. If  $W$  has  $n$  a's then it is in a set represented by  $(b)^*$ . If  $W$  has only one a then it is in a set represented by  $b^*ab^*$ . If  $W$  has more than one a, write  $W = W_1aW_2$ , where  $W$  does not contain a. Then  $W$ , is in a set represented by  $(b+abb)^*$ . So the given set is represented by the regular expression  $b^* + (b+abb)^*ab^*$ .
3. The set of all strings over  $\{0, 1\}$  is represented by  $(0+1)^*$  and when begining with 00 it becomes  $00(0+1)^*$ .

**S.5 (a)**

The language  $\{0, 1\}^* \setminus \{01\}$  represents the string which ends always with substring 01. Both FA I and II represents the above string.

**S.6 (c)**

We can see that, in any state, if the input is b, the machine remains in the same state. That is, the occurrence of b does not affect the current status of machine. Hence, whichever may be final state, the machine accepts the strings independent of the number of b's, though there may be constraints on a's.

**S.7 (d)**

As seen in the transition diagram a string ending with 0 should be a string containing only zeros. But since A is an non-accepting state, no string ending with zero will be accepted.

**S.8 (d)**

For every NFA, an equivalent DFA can be constructed. Hence, a DFA can simulate NFA.

Note : Power of NFA and DFA are same.

**S.9 (c)**

Minimum string accepted is 011. At beginning any number of combinations of 1s and 0s are possible.

$$\therefore (0|1)^*011$$

**S.10 (b)**

Totally there are  $mn$  bits. Each bit will be in one of the two possible states 1 or 0. So the entire memory made up of  $mn$  bits will be in one of the possible  $2^{mn}$  states.

**S.12 (d)**

Above all statements are related to the power of pushdown machine.

**S.13 (a)**

Type -1 grammars (context sensitive grammars) generate the context sensitive languages. The languages described by these grammars are exactly all languages that can be recognized by a linear bounded automaton (a non-deterministic turing machine whose tape is bounded by a constant times the length of the input).

**S.14 (c)**

Because NFA and DFA don't have storage capability.

**S.15 (a)**

The articulation points are nodes 2, 3 and 5, on removing any of these nodes the graph is not connected.

**S.16 (a)**

Since input is only 1, it is not FSM to check a given binary number divisible by 3, as in a binary number input is either 0 or 1.

It is FSM to check whether a given unary number is divisible by 3.

Transition table:

States	Input
$q_0$	$q_1$
$q_1$	$q_2$
$q_2$	$q_0$

**S.17 (d)**

All the other three have definite transition to a unique state on a given input symbol. But NFA may have transitions to more than one state on an input symbol.

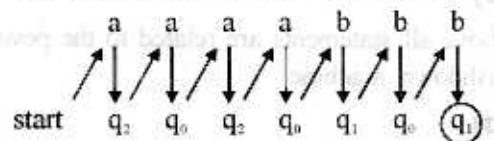
**S.18 (b)**

The best method to solve this problem is to take one string (only one will be sufficient, though more than one can be used for cross-checking) which is valid and check in which state it resides.

E.g. 'b' is a valid string.

But on 'b' in the start state we go to  $q_1$  state. So  $q_1$  is the final state. But we will consider one more case for confirmation.

Take aaaabbb which is a valid string.



Here  $q_1$  is the final state.

**S.19 (d)**

We can see from table that on b, the transition is made either on  $q_1$  or  $q_3$  (see column for input symbol b). Therefore,  $q_1$  and  $q_3$  will denote all strings ending in b.  $q_1$  denotes all strings ending in b except strings ending with aab. But  $q_3$  denotes all strings ending with aab.

**S.20 (b)**

For NFA, like DFA Q is a finite non-empty set of states. And definitely transition table can be constructed, though entries in the table may be sets of states.

**S.21 (d)**

That idea will become clear if we know what each state represents. Then we redraw the given transition table as given below:

	$\delta$	a	b
String ending with	Start	$q_0$	$q_1$
a	$q_0$	$q_0$	$q_2$
b	$q_1$	$q_0$	A
ab	$q_2$	$q_3$	$q_1$
aba	( $q_3$ )	$q_3$	$q_3$

We can see that,  $q_1$  denotes strings ending with b and that b is not preceded by a; As strings ending with b preceded by a is denoted by  $q_2$ . So while in  $q_1$ , if we get b, we will be in  $q_1$  itself. Hence A denotes  $q_1$ .

**S.22 (a)**

The above NFA is NFA with  $\epsilon$  moves

$$\epsilon\text{-closure } (q_0) = \{q_0, q_1, q_2, q_3\}$$

$$\epsilon\text{-closure } (q_1) = \{q_1, q_3\}$$

$$\epsilon\text{-closure } (q_2) = \{q_2\}$$

$$\epsilon\text{-closure } (q_3) = \{q_3\}$$

Transition Table

State	Input	
	0	1
$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$	$[q_1, q_2, q_3]$
$[q_1, q_2, q_3]$	$[q_2]$	$[q_1, q_2, q_3]$
$[q_2]$	$[]$	$[q_2]$

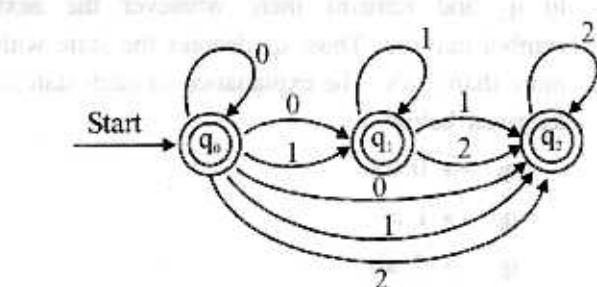
**S.23 (b)**

From the transition diagram we can see that  $q_3$  accepts only those strings which starts with b and ends in a.

Interpretation	$\delta$	a	b
start	end	start	$q_0$
a	a	$q_0$	$q_1$
a	b	$q_1$	$q_0$
a	bb	( $q_2$ )	$q_0$
b	a	( $q_3$ )	$q_4$
b	b	$q_4$	$q_3$

**S.24 (b)**

The given machine accepts the language that consists of strings containing any no. of 0's followed by any number of 1's followed by any number of 2's. Hence, all the 3 given strings will be accepted. The idea will be clear from the equivalence NFA without  $\epsilon$ -moves as given below.

**S.25 (b)**

From definition:

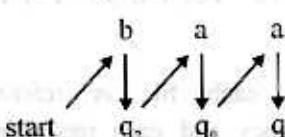
Also we know that entries in transition table for NFA is subsets of states And only III gives that output as a set of  $\delta(q, \epsilon)$ . Hence II should be for DFA. Only one option i.e. (b) relates III to NFA.

**S.26 (a)**

The unary number has a single symbol and the length of the string is its value. On first symbol of number (i.e. 1), we will move to remainder 1 state; when next input is received we move to remainder 2 state and on next we move to 0 remainder state i.e.  $q_3$  as we are checking for divisibility by 3. Also, zero is divisible by 3, therefore,  $q_0$  is also an accepting state.

**S.27 (c)**

Simulation:



Hence,  $q_3$  is the final state.

**S.28 (c)**

Transition table for the given automaton A is as shown below

	$\delta$	0	1
start $\rightarrow$	$q_0$	$q_2$	$q_1$
Odd no. of 1's	$q_1$	$q_1$	$q_2$
Even no. of 1's	$q_2$	$q_2$	$q_1$

So, the no. of states required is 3.

**S.29 (b)**

$$a^R = a$$

$$(wa)^R = aw^R \text{ for all } a \in \Sigma, w \in \Sigma^*$$

**S.30 (b)**

Output of Moore machine depends only on the present state.

Output of Mealy machine depends on the present state and the present input. (b) 28.2

**S.31 (d)**

Both are not equivalent, since first FSM has same start and accepting state which is not in case with second FSM. Also no transition is given in second FSM so (a) is false.

Also (b) is false as there is an implicit transition from any state to itself. So  $\epsilon$  will be accepted.

**S.32 (a)**

A basic machine recognizes an input set I and produces a output set O, where I and O are finite. It has neither memory nor internal states

**S.33 (b)**

From the given FSM we can see that the accepted language is even no. of zero's followed by even no. of ones.

**S.34 (d)**

If  $q_3$  becomes the final state, then the given FA accepts the strings containing exactly 3a's. We can see from the table that once the machine enters in  $q_3$ , it remains in  $q_3$  for any further entries of b. But if an additional 'a' comes it goes to  $q_4$  and remains there whatever the next symbol may be. Thus,  $q_4$  denotes the state with more than 3 a's. The explanation of each state is as given below:

$$q_0 \rightarrow 0 \text{ a's}$$

$$q_1 \rightarrow 1 \text{ a}$$

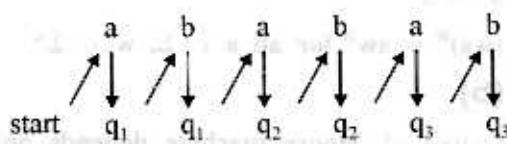
$$q_2 \rightarrow 2 \text{ a's}$$

$$q_3 \rightarrow 3 \text{ a's}$$

$$q_4 \rightarrow \text{more than 3 a's}$$

If too many words are confusing you, don't worry. Go by simulation.

Simulation:



∴ string ends in  $q_3$ . It will be accepted.  
Similarly for others.

### S.35 (d)

For the given DFA, the string ending with 0 should be a string containing only zeros. Hence, total no. of string possible is  $n$ , starting with string of length 1, length 2, ..., length  $n$  and all of them contain only zeros.

### S.36 (a)

Though from any state, on any input symbol, there is a transition to exactly one state but it is not necessary that there will be a transition for each and every input symbol. Sometimes we design the DFA to "die" in situations where we know it is impossible for any extension of the input sequence to be accepted.

### S.37 (a)

Here the final state and the start state are one and the same. No transition is there. But by definition, there is an (implicit)  $\epsilon$ -transition from any state of itself. So, the only string that could be accepted is  $\epsilon$ .

### S.38 (b)

Let 011011 be the input to the FSM and let it be fed from the right (i.e. least significant digit first). If we add 1 to 011011 we should get 011100. Whenever we add 1 to a 1, we make it 0 and carry 1 to the next stage (state) and repeat the process. If we add 1 to a 0, then first make it 1 and all the more significant digits will remain the same, i.e., a 0 will be 0 and a 1 will be 1. That's what the given machine does.

### S.39 (c)

If  $q_3$  becomes the final state, then the given FA accepts the strings containing exactly 3 a's. We can see from the table that once the machine enters in  $q_3$ , it remains in  $q_3$  for any further entries of 'a'. But if an additional 'b' comes it goes

to  $q_4$  and remains there whatever the next symbol may be. Thus,  $q_4$  denotes the state with more than 3 a's. The explanation of each state is as given below:

$q_0 \rightarrow 0$  a's

$q_1 \rightarrow 1$  a

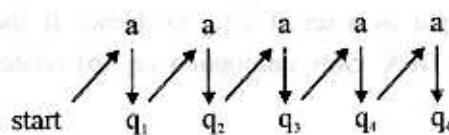
$q_2 \rightarrow 2$  a's

$q_3 \rightarrow 3$  a's

$q_4 \rightarrow$  more than 3 a's

If too many words are confusing you, don't worry. Go by simulation.

Simulation:



Since at the end of the string the machine resides in  $q_4$  it will be accepted if  $q_4$  is the final state.

Similarly II will also make the machine reside in  $q_4$ . Hence, it will also be accepted. But this will not be the case for III.

### S.40 (d)

Since the access is sequential, greater the distance, greater will be the access time. Since all the files are referenced with equal frequency, overall access time can be reduced by arranging them i.e., F3, F4, F1, F5, F6, F2.

### S.41 (d)

Since each file is referenced with equal frequency and each record in a particular file can be referenced with equal frequency, average access time will be  $(25 + (50 + 40) + (50 + 80 + 50) + \dots)/6 = 268$  (approximately).

### S.42 (a)

Transition table for the required automaton A is as given below:

Transition table:

$\delta$	a	b
start $q_0$	$q_1$	$q_2$
$q_1$	$q_2$	$q_1$
$q_2$	$q_1$	$q_2$

where,

- $q_0 \Rightarrow$  initial state
- $q_1 \Rightarrow$  odd no. of a's
- $q_2 \Rightarrow$  even no. of a's (final state)

### S.43 (a)

The new automaton accepts strings containing the substring aab. It doesn't matter whether the string ends with aab or not. This means that, once we get 'aab', we will be in the final (accepting) state whatever the symbols coming after that. Therefore, once we enter  $q_3$  state, we will be in  $q_3$  itself for any further inputs. There will be no changes in  $q_2$ .

$$\therefore q_2 = (q_2/a, q_3/b) \text{ (same as given)}$$

i.e.  $q_2$  on a makes transition to itself and  $q_2$  on b makes transition to  $q_3$ .

$$\text{And } q_3 = (q_3/a, q_3/b)$$

i.e.  $q_3$  on a or b makes transition to itself.

### S.44 (c)

Since  $q_3$  is the final state i.e. accepting strings ending in aab, therefore the state that denotes the string ending in aa will be the state that on b goes to final state  $q_3$ . From the table we can see that the required state is  $q_2$ .

### S.45 (d)

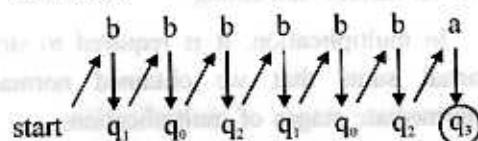
The correct machine is

Remainder	start	$\delta$		o/p
		0	1	
0	$q_0$	$q_0$	$q_2$	0
1	$q_1$	$q_2$	$q_0$	1
$2(10)_2$	$q_2$	$q_1$	$q_2$	2

which is not in options. Hence, **none of the these.**

### S.46 (b)

Simulation:



whereas the other strings don't end in the final state

### S.47 (b)

Here, since no information is given by which we can get the correct answer, hence we will go by method of elimination. Now again there is no option like 'can't be determined'. So we have to eliminate three options that are definitely false using the given information.

Eliminate option (a). Since  $\epsilon$ -closure of  $q_0$  includes  $q_4$  (given), then if  $\epsilon$ -closure  $q_0$  contains  $q_2$ , it should also contain  $q_4$ .

Eliminate option (c). Since  $\epsilon$ -closure of any state include itself, here  $q_0$  is missing.

Eliminate option (d). Since  $q_5$  is not the state of given machine.

It is given that  $Q = \{q_0, q_1, q_2, q_3, q_4\}$

### S.48 (a)

Again this can be solved by taking two strings that are according to the given constraints. The final picture is shown below.

		$\delta$	a	b
a	b	start	$q_2$	$q_1$
even	even	$q_0$	$q_2$	$q_1$
even	odd	$q_1$	$q_3$	$q_0$
odd	even	$q_2$	$q_0$	$q_3$
odd	odd	$q_3$	$q_1$	$q_2$

### S.49 (d)

The given DFA accepts the language:

$L = \{w \mid w \text{ has both even no. of 0's and even no. of 1's}\}$

The job of the states of this DFA is to count, both the number of 0's and number of 1's, but count them modulo 2. That is, the state is used to remember whether the number of 0's seen so far is even or odd, and also to remember whether the number 1's seen so far is even or odd. There are thus four states, which can be given the following interpretations.

$q_0$  : Both the number of 0's seen so far and the number of 1's seen so far are even.

$q_1$  : The number of 0's seen so far is even, but the number of 1's seen so far is odd.

q<sub>2</sub>: The number of 1's seen so far is even, but the number of 0's seen so far is odd.

q<sub>3</sub>: Both the number of 0's seen so far and the number of 1's seen so far are odd.

**Note:** State q<sub>0</sub> is both the start state and the alone accepting state. It is so because before reading any inputs, the numbers of 0's and 1's seen so far are both zero, and zero is even.

### S.50 (d)

The graph does not contain any  $\epsilon$ -move and there is only one initial state.

The three equations for q<sub>1</sub>, q<sub>2</sub> and q<sub>3</sub> can be written as

$$\begin{aligned} q_1 &= q_1a + q_2b + \Lambda, \\ q_2 &= q_1a + q_2b + q_3a, \\ q_3 &= q_2a \end{aligned}$$

It is necessary to reduce the number of unknowns by repeated substitution. By substituting q<sub>3</sub> in q<sub>2</sub> – equation, we get

$$\begin{aligned} q_2 &= q_1a + q_2b + q_2aa \\ &= q_1a + q_2(b + aa) \\ &= q_1a(b + aa)^* \end{aligned}$$

Substituting q<sub>2</sub> in q<sub>1</sub>, we get

$$\begin{aligned} q_1 &= q_1a + q_1a(b+aa)^*b + \Lambda \\ &= q_1(a + a(b + aa)^*b) + \Lambda \end{aligned}$$

Hence q<sub>1</sub> =  $\Lambda(a + a(b + aa)^*b)^*$

$$q_2 = (a + a(b + aa)^*b)^*a(b + aa)^*$$

$$q_3 = (a + a(b + aa)^*b)^*a(b + aa)^*$$

Since q<sub>3</sub> is the final state, the set of strings recognized by the graph is given by

$$(a+a(b+aa)^*b)^*a(b+aa)^*$$

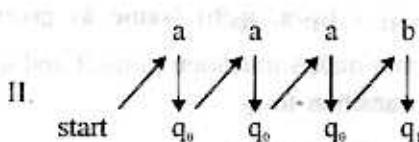
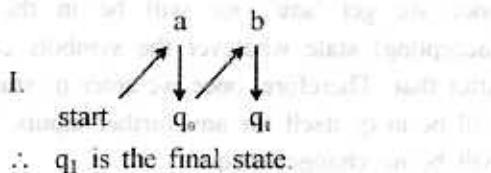
### S.51 (c)

First of all q<sub>3</sub> cannot be the final state as to enter into q<sub>3</sub>, starting letter should be b. Now if q<sub>0</sub> is a final state then it should not accept string ending with a. But it accepts strings containing any number of a's without a single b. Therefore, q<sub>0</sub> also cannot be a final state. Therefore, eliminate all options that either contain q<sub>0</sub> or q<sub>3</sub> or both. So option (a), (b) and (d) are eliminated. Hence (c). This method is widely used for objective type questions and is known as 'Method of Elimination'.

### Or

We can also solve this problem by simulation. Here we take two valid strings and check in which state they are accepted. Here sometimes choosing of proper trial data (i.e. two valid strings in this case) becomes difficult if options have combination of entries. For e.g. option (a) q<sub>0</sub>, q<sub>1</sub>.

**Simulation:**



Here too q<sub>1</sub> comes out to be a final state.

**Hint for choosing proper trial data:**

- Choose the minimum valid string possible. It can be
  - empty string.
  - String not containing some of the input symbols.
- Choose the minimum valid string that contains all the input symbols possible as per the given constraints.

### S.52 (b)

It is the limitation of FSM that, it does not have the capacity to remember arbitrarily large amount of information, as it has only a fixed number of states and these sets a limit to the length of sequence it can remember.

Head can never move in reverse direction.

∴ FSM cannot retrieve what it has read previously, before coming to current position on the tape. As it cannot retrieve, we cannot say it can remember something.

In multiplication, it is required to store the partial sums that we obtained normally at intermediate stages of multiplication.

∴ FSM cannot multiply two given arbitrarily long numbers.

## S.53 (d)

Here, (a) and (b) both represent the same language.

Interpretation	$\delta$	a	b
start end	start	$q_0$	$q_4$
a a	$q_0$	$q_0$	$q_1$
a b	$q_1$	$q_0$	$q_2$
a bb	$q_2$	$q_0$	$q_2$
b a	$q_3$	$q_3$	$q_4$
b b	$q_4$	$q_3$	$q_4$

## S.54 (a)

We know that if we convert a NFA with  $\epsilon$ -moves to NFA without  $\epsilon$ -moves, all the states whose  $\epsilon$ -closure includes final state(s) are marked as final states.

$$\epsilon\text{-closure } (q_0) = \{q_0, q_1, q_2\}$$

$$\epsilon\text{-closure } (q_1) = \{q_1, q_2\}$$

$$\epsilon\text{-closure } (q_2) = \{q_2\}$$

We can see that epsilon closures of  $q_0$ ,  $q_1$  and  $q_2$  include the final state  $q_2$ . Hence, all of them will be marked as final states.

## S.55 (d)

Set of all strings which will be accepted by an FA is a language, such language is known as **Regular set**.

## S.56 (a, c)

$M_1$  is nondeterministic finite automata. For every Nondeterministic machine  $M_1$  there exists an equivalent deterministic machine  $M_2$  recognizes the same language.

Also, If  $M_1$  is a non-deterministic turing machine, there is a deterministic turing machine  $M'$ , such that  $T(M_1) = T(M')$ .

## S.58 (c)

For every non-deterministic PDA there is no equivalent deterministic PDA.

## S.59 (b)

Present state	Initial Input Sequence	Next state, z
D	x = 1	B, 1
B	x = 0	$\downarrow$ C, 1

C is the final state

$$\therefore x = 1 \quad 0$$

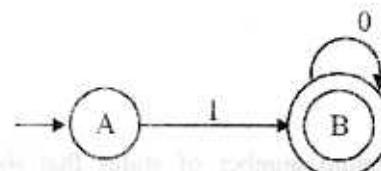
## S.60 (a, c)

$\{w \in w^R \mid w \in \{a, b\}^*\}$  and  $\{a^n b^n c^n \mid n \geq 0\}$  is accepted by deterministic pushdown automata.

## S.61 (d)

The numbers 1, 2, 4, 8, ...,  $2^n$  written in binary can be recognized by

Deterministic finite state automata.



The numbers 1, 2, 4, ...,  $2^n$  written in unary is nothing but 1, 11, 1111, 11111111 etc. cannot be recognized by deterministic Finite-state automata. The sets of binary string in which number of zeroes is same as the number of one is not recognized by Finite automata. Because finite automata does not have external memory to store zero and one and then check for the equivalence.... same way the set {1, 10, 11011, 1110111,} is set in which string is symmetric about one so it cannot be recognized by Deterministic finite automaton.

## S.62 (c, d)

$$\text{Let } r = 110 * (0 + 1)$$

$$L(r) = \{110, 111, 1100, 1101, \dots\}$$

$L(r)$  contains string 1101.

$$\text{Let } S = 1(0 + 1)^* 101$$

$$L(S) = \{1101, 10101, 11101, \dots\}$$

$L(S)$  contains 1101 as string.

$$\text{Let } t = (00 + (11)^* 0)^*$$

$$L(t) = \{\epsilon, 00, 0, 110, \dots\}$$

L(t) does not contain 1101 as string

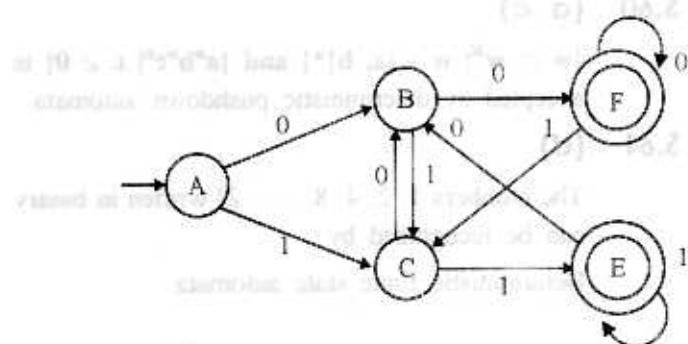
Let  $u = (10)^* (00 + 11)^*$

$$L(u) = \{ \epsilon, 10, 00, 11, 1000, 1011 \dots \}$$

L(u) does not contain string 1101.

S.63 (b)

Minimum states are 5



S.65 (a)

Minimum number of states that the DFA will have is 8.

For all strings which have number of a's divisible by 6 is having 6 states

(i.e. remainder 0, 1, 2, 3, 4, 5)

for all strings which have number of b's divisible by 8 is having 8 states

(i.e. remainder 0, 1, 2, 3, 4, 5, 6, 7)

i.e. minimum number of states that the DFA will have in which all strings will have number of a's divisible by 6 and number of b's divisible by 8 is 8.

S.66 (b)

If the NFA has  $N$  states, the resulting DFA can have upto  $2^N$  states, exponentially more, which sometimes makes the construction impractical for large NFA's.

S.67 (b)

$\{x \mid \text{length of } x \text{ is divisible by 3}\}$  has 3 states.

Because length of 'x's divisible by 3 has remainder as 0, 1, or 2.

S.68 (c)

The language accepted by a pushdown automaton in which stack is limited to 10 items is best described as Deterministic context free.

S.69 (a)

Outputs the sum of the present and previous bits of the input.

S.70 (c)

The given bit pattern can be represented as:

$$1 - - 1 - - 1$$

The four blanks can be filled in  $2^4 = 16$  ways. Therefore there are 16 such strings in this pattern. Not all of these are accepted by the machine. The strings and its acceptance is given below:

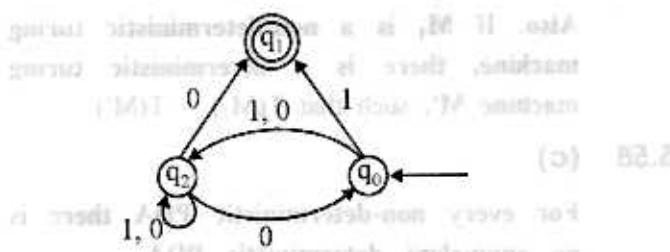
	accepted						
1 0 0 1	0 0 1	✓					
1 0 0 1	0 1 1	✓					
1 0 0 1	1 0 1	✓					
1 0 0 1	1 1 1	✓					
1 0 1 1	0 0 1	✓					
1 1 0 1	0 0 1	✓					
1 1 1 1	0 0 1	✓					

Only these seven strings given above are accepted. All other strings in this pattern are rejected.

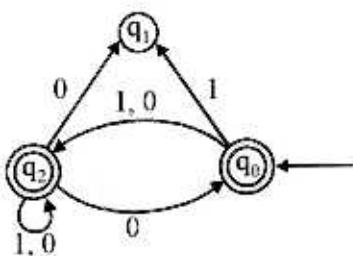
∴ Correct answer is (c).

S.71 (b)

The given machine M is



Now the complementary machine  $\bar{M}$  is



In the case of DFA,  $L(\bar{M}) = \overline{L(M)}$  but in the case of NFA this is not true. In fact  $L(\bar{M})$  and  $L(M)$  have no connection.

$\therefore$  to find  $L_1 = L(\bar{M})$  we have to look at  $\bar{M}$  and directly find its language.

Clearly  $\lambda \in L(\bar{M})$ , since  $q_0$  is accepting it

$(0+1)(0+1)^* \in L(\bar{M})$ , since  $q_2$  is accepting it.

$$\therefore L(\bar{M}) = L_1 = 1 + (0+1)(0+1)^*$$

$$\therefore L_1 = (0+1)^* = \{0, 1\}^*$$

### S.72 (a)

The given finite state machine accepts any strings  $w \in \{0, 1\}^*$  in which the number of 1's is multiple of 3 and the number of 0's is multiple of 2.

### S.73 (d)

Nf : Non-Deterministic finite automata.

Np : Non-Deterministic pushdown automata.

Df : Deterministic finite automata.

Dp : Deterministic push down automata.

According to "Subset Construction" theorem every language accepted by Non-deterministic finite automata (Nf) is also accepted by some Deterministic Finite automata (Df) so  $Df = Nf$ .

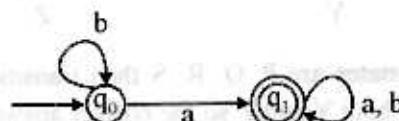
Deterministic pushdown automata (Dp) recognizes a proper subset of the language of context-free languages and the non-deterministic

push down automata recognizes the context-free languages.

So  $Dp \subset Np$

### S.74 (b)

The minimum state FSA is given below for the regular expression  $b^*a(a+b)^*$

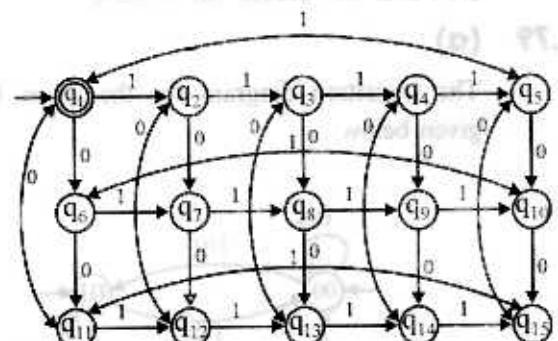


So FSA contains minimum two states  $q_0$  and  $q_1$ .

### S.75 (a)

$L = \{w \mid s \in (0, 1)^* \text{ number of } 0's \text{ and } 1's \text{ in } w \text{ are divisible by } 3 \text{ and } 5 \text{ respectively}\}$

The minimum state deterministic finite automaton accepting the language  $L$  has  $3 \times 5 = 15$  states.



### S.76 (d)

(a) True

(b) True

(c) True

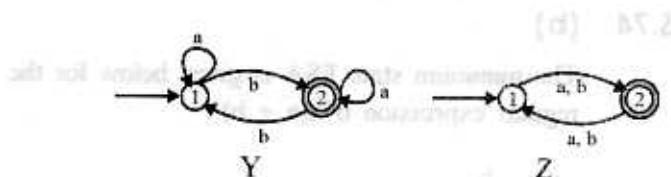
(d) Every subset of a recursively enumerable set is recursive, is false, since a set is a subset of itself and there are RE languages, which are not REC.

### S.77 (a)

The transition table for Z and Y is as follows

	a	b		a	b
$\rightarrow 1$	1	2	$\rightarrow 1$	2	2
$2(F)$	2	1	$2(F)$	1	1

Z contains 2 states and Y contains 2 states so the product automaton  $Z^*Y$  contains  $2 \times 2 = 4$  states.



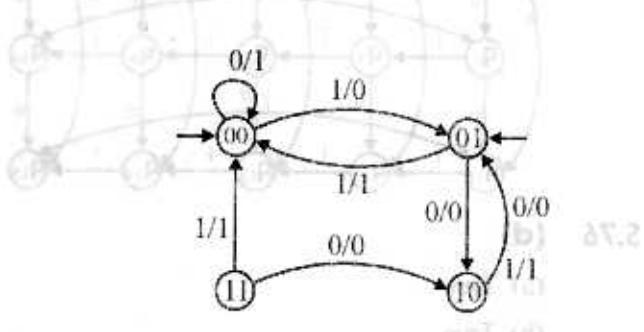
If the states are P, Q, R, S then transition table satisfy both Y and Z so the correct answer is (a).

### S.78 (d)

Since it is known that the set of DCFL's (Deterministic Context Free Language) are a proper sub class of the set of CFL's, NPDA's (**Non Deterministic Push-down Automata**) which corresponds in general to class of CFL's cannot in general be converted to DPDA's (**Deterministic Push-down Automata**), which correspond to the class of DCFL's. So, choice (d) is false.

### S.79 (a)

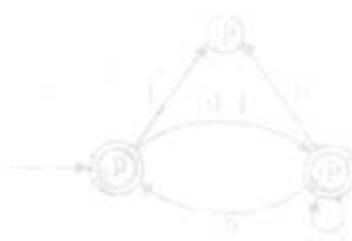
The transition diagram for the given FSM is given below:



As can be seen from above diagram the minimum length of the input string that will take machine from 00 to 01 with output 1, is three.

In fact the minimum length input string is "101". Which will give an output of 1, while reaching state 01.

(b) 55.2



switch on a / bus X not sides not same left

d	e	d	e
5	5	1	1



# REGULAR EXPRESSIONS AND LANGUAGES

**1.2**

## LEVEL-1

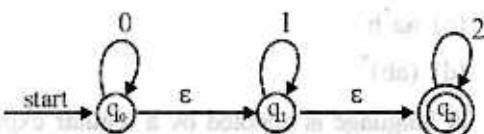
**Q.1**  $(a + b)^*a(a + b)^*$

- (a) All strings containing atleast one 'a'.
- (b) All strings of odd length
- (c) All strings of even length
- (d) Strings of odd length having 'a' and 'b'

**Q.2** Strings having at the most one occurrence of ab or ba but not both

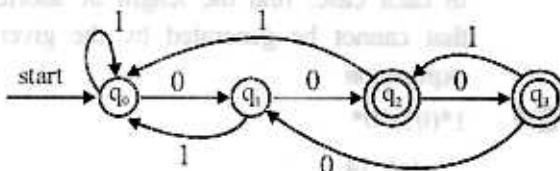
- (a)  $a^*b^* + b^*a^*$
- (b)  $a^*ba^* + b^*a^*$
- (c)  $b^*ab + a^*ba$
- (d) None of these

**Q.3** Give the regular expression described by the following NFA.



- (a)  $(0^* + 1^* + 2^*)^*$
- (b)  $0^*1^*2^*$
- (c)  $(0 + 1 + 2)^*$
- (d) None of these

**Q.4**



Which of the following string is accepted by above FA.

- (a) 0 1 0 1 1 0 0 1 0 0 0 1
- (b) 0 1 0 0 1 0 0 0 1 1
- (c) 1 0 1 0 0 1 0 0 1 0
- (d) 0 1 0 0 1 0 1 0 1 0

**Q.5**

Strings containing all possible combinations of 0's and 1's but not two consecutive zeros.  $\Sigma = \{0, 1\}$

- (a)  $(1 + 01)^*$
- (b)  $(1 + 10)^*$
- (c) None of the above
- (d) Both (a) & (b)

**Q.6**

If  $L(r) = \{a, c, ab, cb, abb, cbb, \dots\}$ . Find r ?

- (a)  $r = ab^* + cb^*$
- (b)  $r = ab^* + bc^* + ca^*$
- (c)  $r = ab^* + c^*b$
- (d)  $r = a^*b^*c^*$

**Q.7** Convert the following R.E. in simple English

$$r = (aa)^* + (bb)^*$$

- (a) String containing 0 or more a's OR strings containing one or more b's.
- (b) Strings of even length containing only a's and strings of odd length containing only b's.
- (c) Strings containing only a's or only b's
- (d) Strings containing 'aa' as a substring or 'bb' as a substring

**Q.8** If  $L = \{100, 10, 111\}$  and  $M = \{\epsilon, 100\}$  then  $LM$  is,

- (a)  $\{100, 10, 111, 100100, 10100, 111100\}$
- (b)  $\{\epsilon, 100100, 10100, 111100\}$
- (c) Both (a) & (b)
- (d) None of the above

### Common Data For Questions 9 & 10:

In each case, find the length of shortest string that cannot be generated by the given regular expression

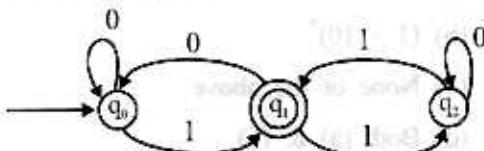
**Q.9**  $1^*(01)^*0^*$

- (a) 14
- (b) 10
- (c) 3
- (d) 12

**Q.10**  $(0^* + 1^*)(0^* + 1^*)(0^* + 1^*)$

- (a) 15
- (b) 5
- (c) 4
- (d) 2

**Q.11** Consider the following diagram and select the correct statement



- (a) The graph represents  $M = (\{q_0, q_1\}, \{1, 0\}, q_0, \{q_2\})$
- (b) The graph represents  $M = (\{q_0, q_2\}, \{0, 1\}, q_0, \{q_1\})$
- (c) The graph represents  $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$
- (d) None of these

**Q.12** Using the diagram from Q.16, find which of the following is accepted by it.

- |                            |      |       |
|----------------------------|------|-------|
| (i) 101                    | (ii) | 0111  |
| (iii) 1100                 | (iv) | 11001 |
| (a) (i), (ii), (iii), (iv) |      |       |
| (b) (i), (ii), (iv)        |      |       |
| (c) (i), (iv)              |      |       |
| (d) (i), (iii)             |      |       |

**Q.13** A language L is called regular \_\_\_\_\_

- (a) Iff there exist some non-deterministic finite accepter M such that  $L = L(M)$
- (b) Iff there exist some non-deterministic finite accepter M such that  $L \neq L(M)$
- (c) Iff there exist some deterministic finite accepter M such that  $L = L(M)$
- (d) Iff there exist some deterministic finite accepter M such that  $L \neq L(M)$

**Q.14** The logic of pumping lemma is a good example of

- (a) iteration
- (b) recursion
- (c) the divide and conquer technique
- (d) the pigeon hole principle

**Q.15** Consider the production grammar,

$$S \rightarrow AB/AS$$

$$A \rightarrow a/aA$$

$$B \rightarrow b$$

Which of the following regular expressions corresponds to the given grammar?

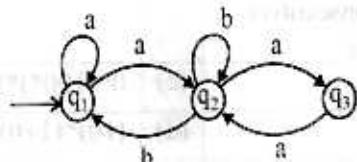
- (a)  $aa^*b$
- (b)  $a(ab)^*b$
- (c)  $aa^*b^+$
- (d)  $(ab)^*$

**Q.16** A language is denoted by a regular expression  $L = (x)^*(x + yx)$ . Which of the following is not a legal string in L?

- (a) xyxyx
- (b) xxyx
- (c) xxx
- (d) yx

- Q.17** The regular expression  $(a + b)(a + b)$  denotes the set

- {a, b, aa, bb, ab, ba}
- {a, b, aa, bb}
- {aa, bb, ab, ba}
- {a, b}

**Q.18**

Which Regular Expression denotes the language of the given machine

- $a^*a(a + bb)^*(a + (b + aa)^*)^*$
- $a^*a(b + aa)^*ba(b + aa)^*a$
- $a^*(a(b + aa)^*b)^*(b+bb)^*a$
- can't be determined

- Q.19** Which of the following statements is true?

- The CFL's are not closed under intersection
- The context free languages are closed under substitution
- Context-free languages are closed under union, concatenation and kleen closure.
- All of the above

- Q.20** Strings containing at most 2a's is given by

- $b^* + b^*ab^* + b^*ab^*ab^*$
- $b^*(a + \epsilon)b^*(a + \epsilon)$
- $b^*ab^*a$
- none of these

- Q.21** The regular expression  $(a | b)^*$  denotes the set of all strings

- equal to regular expression  $(a^*b^*)^*$
- with one or more instances of a or b
- with zero or more instances of a or b
- both (a) and (c)

- Q.22** The regular sets are closed under

- kleens closure
- concatenation
- union
- All of these

## LEVEL-2

- Q.23** Which of the following statements are false?

- Pumping Lemma can show a language is regular
  - Pumping Lemma can show a language is not regular
  - Pumping Lemma cannot show a language is regular
  - Pumping Lemma cannot show a language is not regular
- Only I and IV
  - Only II
  - Only III
  - Only IV

- Q.24** Give a regular expression for the set of strings containing all b's or an a followed by b's or just null strings

- $bb^* + ab^*$
- $(\epsilon + a)b^*$
- $(\epsilon + a)b^* + \epsilon$
- $b^* + ab^* + \epsilon$

- Q.25** The regular expression  $[(a + b)(a + b)]^*$  represents a language of \_\_\_\_\_

- All strings of even length
- All strings of odd length
- All strings
- None of above

- Q.26** An identifier in the C programming languages is any string of length 1 or more that contains only letters, digits and underscore (\_) and begins with a letter or an underscore. Write a regular expression for the language of all C identifiers. Use  $\ell$  (for letter), 'd' (for digit) and '\_' (for underscore).

- $(\ell + \underline{ } + \epsilon)(\ell + d + \underline{ })^*$
- $(\ell + \underline{ })(\ell + d + \underline{ })^*$
- $(\ell + \underline{ })(\ell + d + \underline{ })$
- None of these

**Q.27** The length of string should be a multiple of 3

- (a)  $[(a^* + b^*)(aba) + (ab^*)(a^*a)]$
- (b)  $(a+b)^* (a+b)^* (a+b)^*$
- (c)  $[(a+b)(a+b)(a+b)]^*$
- (d) None of the above

**Q.28** Consider the regular expression,

$$r = 0^*(10^*10^*)^*10^*$$

Which of the following are not valid strings?

I. 0101010011

II. 1111001111001

III. 0011001100111

- (a) cannot be determined
- (b) Only I
- (c) Only II
- (d) None of them

**Q.29** Strings in which every group of three symbols should contain atleast one a.

- (a)  $[(a+b)b^* (a+b)a]$
- (b)  $[(\epsilon + b + bb)a]^* [\epsilon + b + bb]$
- (c)  $(abb)^*(bab)^*(bba)^*$
- (d)  $[(a+b)(a+b)a]^*$

**Q.30** Explain the regular expression  $(ab + ba)^*$  in simple English

- (a) Strings of even length and no two a's are consecutive.
- (b) Strings of even length and two consecutive symbols are same.
- (c) Strings of even length
- (d) None of these

**Q.31** Strings in which every occurrence of 'a' should be odd consecutive (i.e. a symbol occurring odd number of times).

- (a)  $b^*[(aa)^*abb^*]^*[(aa)^*a + \epsilon]$
- (b)  $(a+b)aa^*(a+b)$
- (c)  $[(a+b)a]^* (a+b + \epsilon)$
- (d)  $(aba)^*(bba)^*(abb)^*$

**Q.32** Match the following:

	Group 1	Group 2
(i)	Binary strings that are multiple of 2	(a) $(0+1).10$
(ii)	Strings of 1's and 0's with no consecutive 1's	(b) $(0+1)^*.0$
		(c) $0^*(100^*)^*(1+\epsilon)$
		(d) $(10)^*(1+0)^*$

- (a) (i)  $\rightarrow c$ , (ii)  $\rightarrow d$
- (b) (i)  $\rightarrow d$ , (ii)  $\rightarrow c$
- (c) (i)  $\rightarrow b$ , (ii)  $\rightarrow c$
- (d) (i)  $\rightarrow a$ , (ii)  $\rightarrow c$

**Q.33** Let  $r$  and  $s$  be the symbols in the alphabet  $\Sigma$ . Find the simplest regular expression corresponding to the same language as of the given R.E.

$$(r + s + rs + sr)^*$$

- (a)  $(r+s)^*$
- (b)  $(r+rs+sr)$
- (c)  $(s+sr)^*$
- (d)  $(s+rs+s)^*$

**Q.34** Explain the given regular expression in simple English:  $b^*ab^*ab^*$

- (a) string containing more b's than a's and atleast two a's
- (b) string containing atmost two a's
- (c) strings containing atleast two a's
- (d) none of these

**Q.35** Which of the following strings given below belong to the language determined by the R.E.,  $(00^*1)^*$ ?

- I. 000011
- II. 00011
- III. 0010010011
- IV. 0001000100011

- (a) I, II, III and IV
- (b) Only I, II and III
- (c) Only I and II
- (d) Only I

• Deterministic Finite Automata (DFA)

• Non-Deterministic Finite Automata (NFA)

• Regular Expressions

• Regular Languages

• Context-Free Grammars

• Context-Free Languages

• Pushdown Automata

• Turing Machine

• Undecidable Problems

# CONTEXT FREE LANGUAGES AND PUSHDOWN AUTOMATA

**1.3**

## LEVEL-1

**Q.1** Consider the grammar G as given below

$$S \rightarrow aSa \mid bSb \mid \epsilon$$

Which of the following strings cannot be generated using the grammar G?

- (a) aabbaabb
- (b) bbaabaab
- (c) aaabbbaaaa
- (d) None of these

**Q.2** Consider the grammar G as given below

$$S \rightarrow aSa \mid bSb \mid \epsilon$$

If we add the productions  $S \rightarrow a \mid b$  to G, then which of the following strings can be generated?

- I. aababaa
- II. aaabbbaaaa
- III. abababab
- IV. aabbaabbaa

- (a) Only I, II and IV
- (b) Only I
- (c) Only III
- (d) Only I and III

**Q.3** What does the given CFG indicate? (X is the start symbol)

$$X \rightarrow XS \mid SY \mid YSY \mid Y$$

$$Y \rightarrow aY \mid a$$

$$S \rightarrow aB \mid bA$$

$$A \rightarrow a \mid aS \mid bAA$$

$$B \rightarrow b \mid bS \mid aBB$$

(a) Strings that do not contain 'aa'

(b) Strings with half the number of b's than a's

(c) Strings with more number of a's than b's

(d) Strings that do not contain 'bbb'

**Q.4** Write the equivalent R.E. for the following:

String should not contain consecutive a's or consecutive b's.

$$(a + b)^* a(a + b)^+ b(a + b)^*$$

$$(\epsilon + b) (ab)^* (a + \epsilon)$$

$$a^* + (ab)^* + (ba)^* + b^*$$

$$(ab)^* + (ba)^*$$

- Q.5** Every context-free grammar (CFG) can be converted into an equivalent  
 (a) Greibach Normal Form (GNF)  
 (b) Chomsky Normal Form (CNF)  
 (c) Both (a) and (b)  
 (d) None of above
- Q.6** Which of the following CFG's can't be simulated by an FSM?  
 (a)  $S \rightarrow aSb \mid ab$   
 (b)  $S \rightarrow abX, X \rightarrow cY, Y \rightarrow d \mid aX$   
 (c)  $S \rightarrow Sa \mid S$   
 (d) None of these
- Q.7** Consider the grammar G, given below:  
 $S \rightarrow aB \mid bA$   
 $A \rightarrow a \mid aS \mid bAA$   
 $B \rightarrow b \mid bS \mid bBB$
- Which of the following strings will be accepted by the given grammar?  
 (a) aabbbaa  
 (b) bababab  
 (c) Neither (a) nor (b)  
 (d) both (a) & (b)
- Q.8** The grammar  $S \rightarrow aaSbb \mid ab$  can generate the set  
 (a)  $\{a^{2n+1}b^{2n+1} \mid n = 1, 2, 3, \dots\}$   
 (b)  $\{a^n b^n \mid n = 1, 2, 3, \dots\}$   
 (c)  $\{a^{2n+1}b^{2n+1} \mid n = 0, 1, 2, 3, \dots\}$   
 (d)  $\{a^{2n-1}b^{2n-1} \mid n = 0, 1, 2, 3, \dots\}$
- Linked Questions 9 & 10:**
- Q.9** Select the correct statement  
 I. A regular grammar is one that is either right linear or left linear  
 II. Let  $G_1 = (V, T, S, P)$  be a right linear grammar, then  $L(G)$  is a regular language  
 (a) I  
 (b) II  
 (c) Neither I nor II  
 (d) I & II
- Q.10** Consider the following grammar  $G_1$  and  $G_2$ .  
 $G_1 = (\{S\}, \{a, b\}, S, P_1)$ , with  $P_1$  given as  
 $S \rightarrow abS \mid a$   
 $G_2 = (\{S_0, S_1, S_2\}, \{a, b\}, S, P_2)$  with productions  
 $S_0 \rightarrow S_1 ab$   
 $S_1 \rightarrow S_1 ab \mid S_2$   
 $S_2 \rightarrow a$
- Then by using the result in question (9) select the correct statement  
 (a)  $G_1$  and  $G_2$  are regular grammars  
 (b)  $G_1$  is right linear whereas  $G_2$  is left linear grammar  
 (c) Both (a) & (b)  
 (d) None of the above
- Q.11** The language of PDA is  
 (a) Context sensitive language  
 (b) Context free language  
 (c) Regular language  
 (d) none of these
- Q.12** Let L denote the language generated by the grammar  $S \rightarrow 0S0 \mid 00$ , then  
 (a)  $L = (0, 0)^*$   
 (b)  $L = 0^+ 0^+$   
 (c)  $L = (0, 0)^*$   
 (d) all of these
- Q.13** Consider a grammar G  
 $A \rightarrow AB$   
 $AB \rightarrow BC$   
 $BC \rightarrow CD$   
 $CD \rightarrow a$
- The language L generated by G is most accurately said to be  
 (a) Chomsky type 3  
 (b) Chomsky type 2  
 (c) Chomsky type 1  
 (d) Chomsky type 0

**Q.14** The given CFG indicates:

$$\begin{aligned} S &\rightarrow aS \mid bX \mid a \mid b \\ X &\rightarrow aS \mid a \end{aligned}$$

- (a) Strings that do not contain 'aaa'
- (b) Strings with aa or bb but not both
- (c) Strings that has no consecutive b's but, 'a' can be consecutive
- (d) Strings that do not contain 'bbb'

**Q.15** The language accepted by a Pushdown Automata is \_\_\_\_\_.

- (a) Context free language
- (b) Regular Language
- (c) Neither (a) nor (b)
- (d) Both (a) and (b)

**Q.16** Context free grammar is not closed under

- (a) kleen star
- (b) complementation
- (c) union
- (d) product

**Q.17** Context-free grammar can be recognized by

- (a) push down automata
- (b) 2-way linear bounded automata
- (c) finite state automaton
- (d) both (a) and (b)

**Q.18** Context free language are closed under

- (a) complement, kleen closure
- (b) intersection, complement
- (c) union, kleen closure
- (d) union, intersection

## LEVEL-2

**Q.19** Let  $L_D$  be the set of all languages accepted by a PDA by final state and  $L_E$  be the set of all languages accepted by empty stack, then

- (a)  $L_E \subset L_D$
- (b)  $L_D \subset L_E$
- (c)  $L_D = L_E$
- (d) None of these

**Q.20** Write a grammar for strings containing more a's than b's.

$$(a) S \rightarrow aAb \mid aAa$$

$$A \rightarrow aA \mid bA \mid \epsilon$$

with S as starting symbol

$$(b) S \rightarrow aB \mid bA$$

$$A \rightarrow a \mid aS \mid bAA$$

$$B \rightarrow b \mid bS \mid aBB$$

with S as starting symbol

$$(c) X \rightarrow YS \mid SY \mid YSY \mid Y$$

$$S \rightarrow aB \mid bA$$

$$A \rightarrow a \mid aS \mid bAA$$

$$B \rightarrow b \mid bS \mid aBB$$

$$Y \rightarrow aY \mid a$$

with X as starting symbol

- (d) None of the above

**Q.21** Consider the productions of grammar G as given below:

$$S \rightarrow aAb \mid bAa \mid aSa \mid bSb$$

$$A \rightarrow \epsilon \mid Aa \mid Ab$$

Which of the following strings won't be accepted by the language generated by the given grammar G?

- (a) aabbaaa
- (b) aaabbbaaa
- (c) aaaabbbbaaaaa
- (d) All of above

**Q.22** Consider the productions of grammar G as given below:

$$S \rightarrow aAb \mid bAa \mid aSa \mid bSb$$

$$A \rightarrow \epsilon \mid Aa \mid Ab$$

Which of the following string can be generated by the given grammar?

- (a) abbaaba
- (b) aaabbbaaa
- (c) aabbaaa
- (d) ababab

**Q.23** Consider the following grammar

$$G = (N, T, P, S) \text{ where,}$$

$$N = \{S, A, B\}, \quad T = \{a, b\}$$

$$\begin{aligned} P = \{ & (S \rightarrow AB), \\ & (A \rightarrow aa), \\ & (A \rightarrow a), \\ & (B \rightarrow ab), \\ & (B \rightarrow b)\} \end{aligned}$$

Then what is true about the grammar

- I. It produces strings without 'aa'
  - II. It is ambiguous
  - III. It is unambiguous
  - IV. It produces strings of a's followed by b.
- (a) III, IV  
 (b) II, IV  
 (c) I, III  
 (d) I, II

**Q.24** The reduced grammar equivalent to the grammar whose productions are

$$S \rightarrow AB \mid CA$$

$$B \rightarrow BC \mid AB$$

$$A \rightarrow a$$

$$C \rightarrow aB \mid b$$

$$(a) S \rightarrow CA$$

$$A \rightarrow a$$

$$C \rightarrow b$$

$$(b) S \rightarrow CA \mid B$$

$$B \rightarrow BC \mid B$$

$$A \rightarrow a$$

$$C \rightarrow aB \mid b$$

$$(c) S \rightarrow A(B \mid C)$$

$$B \rightarrow B(C \mid A)$$

$$A \rightarrow a$$

$$C \rightarrow aB \mid b$$

$$(d) \text{None of these}$$

**Q.25** Write the grammar generating all string consisting of a's and b's with at least two a's

$$(a) S \rightarrow AaAaA$$

$$A \rightarrow bA \mid a \mid \epsilon$$

$$(b) S \rightarrow aAaA$$

$$A \rightarrow aA \mid bA \mid \epsilon$$

$$(c) S \rightarrow AaAaA$$

$$A \rightarrow aA \mid bA \mid \epsilon$$

$$(d) S \rightarrow A aa A$$

$$A \rightarrow bA \mid a \mid \epsilon$$

**Q.26** Write the grammar for the regular expression  $a^*b^*$

$$(a) S \rightarrow ab \mid \epsilon$$

$$A \rightarrow aA \mid \epsilon$$

$$B \rightarrow bB \mid \epsilon$$

$$(b) S \rightarrow AB$$

$$A \rightarrow aA \mid \epsilon$$

$$B \rightarrow bB \mid \epsilon$$

$$(c) S \rightarrow AB$$

$$A \rightarrow aA \mid bB \mid \epsilon$$

$$B \rightarrow bB \mid aA \mid \epsilon$$

$$(d) \text{Both (a) and (b)}$$

**Q.27** If  $G = \{ \{S\}, \{a\}, S, \{S \rightarrow SS\} \}$ , find the language generated by  $G$

$$(a) L(G) = a^*$$

$$(b) L(G) = a^+$$

$$(c) L(G) = \emptyset$$

$$(d) \text{Both (a) and (b)}$$

**Q.28** Consider the grammar  $G$

$$S \rightarrow AB$$

$$A \rightarrow aAA \mid \epsilon$$

$$B \rightarrow bBB \mid \epsilon$$

Find the nullable symbol in the given grammar

$$(a) A, B and S$$

$$(b) A and B$$

$$(c) B$$

$$(d) A$$

**Q.29** Which of the following is true?

- If the grammar  $G_1$  is constructed from  $G$  by eliminating  $\epsilon$ -productions, then  $L(G_1) = L(G)$
  - If the grammar  $G_1$  is constructed from grammar  $G$  by eliminating unit productions, then  $L(G_1) = L(G) - \{\epsilon\}$
- Only I
  - Only II
  - Both I and II
  - Neither I nor II

**Q.30** Find the odd man out (with respect to CFLs).

Union, Concatenation, Closure, Intersection

- Union
- Intersection
- Closure
- Concatenation

**Q.31**  $V = \{A, B, C\}$

$T = \{a, b, c\}$

$G = (V, T, P, A)$

where  $P$  consists of

- $$\begin{aligned} A &\rightarrow AB \\ AB &\rightarrow BC \\ B &\rightarrow a \end{aligned}$$

The above grammar is called a

- Regular grammar
- Context free grammar
- Context-sensitive grammar
- None of the above

**Q.32** The set  $\{a^n b^n \mid n = 1, 2, \dots\}$  can be generated by which of the following CFGs.

- $S \rightarrow ab \mid aSb$
  - $S \rightarrow aaSbb \mid ab$
  - $S \rightarrow ab \mid aSb \mid \epsilon$
  - $S \rightarrow aaSbb \mid ab \mid aabb$
- I, IV
  - II, IV
  - II
  - III

**Q.33** Explain the given regular expression in simple English.

$$a(a + bb)^*$$

- All strings starting with a and having an even number of b's
- All strings starting with a and having every occurrence of b as even consecutive (i.e., even number of b's appear in succession)
- None of them
- Both (a) and (b)

**Q.34** The language  $\{a^m b^n c^{m+n} \mid m, n \geq 1\}$  is

- regular
- context-free but not regular
- context sensitive but not context free
- type-0 but not context sensitive

**Q.35** Consider a grammar  $G = (\{x, y\}, \{s, x, y\}, P, s)$  where the elements of parse:

$$\begin{aligned} S &\rightarrow xY \\ S &\rightarrow yX \\ S &\rightarrow xZ \\ X &\rightarrow x \\ Y &\rightarrow y \\ Z &\rightarrow z \end{aligned}$$

The language  $L$  generated by  $G$  most accurately is said to be

- Chomsky type 3
- Chomsky type 2
- Chomsky type 1
- Chomsky type 0

**Q.36** Let  $L_1 = \{x \mid x \text{ is a palindrome in } (0+1)^*\}$

$$\begin{aligned} L_2 &= \{\text{letter (letter + digit)}^*\}, \\ L_3 &= \{0^n 1^n 2^n \mid n \geq 1\} \\ L_4 &= \{a^m b^n a^{m+n} \mid m, n \geq 1\} \end{aligned}$$

Mark the incorrect statement:

- Both  $L_3$  and  $L_4$  are context sensitive language
- Both  $L_1$  and  $L_2$  are regular sets
- $L_2$  is a regular set and  $L_4$  is not a context free language
- $L_1$  is context free language and  $L_3$  is context sensitive language

## LEVEL-3

**Q.37** Consider the grammar

$$S \rightarrow SS$$

$$S \rightarrow (S)$$

$$S \rightarrow \epsilon$$

How many steps will be required to derive ((0 0) 0)?

- (a) 16
- (b) 10
- (c) 12
- (d) 15

**Q.38** The CFA equivalent to regular expression  $(011 + 1)^*(01)^*$  is:

(a)  $S \rightarrow AB \mid BC$

$B \rightarrow AB \mid A$

$C \rightarrow CD \mid D$

$A \rightarrow 011 \mid 1$

$D \rightarrow 01$

(b)  $S \rightarrow AB \mid BD$

$A \rightarrow 011 \mid 1$

$B \rightarrow BA \mid A$

$D \rightarrow 01$

(c)  $S \rightarrow BC \mid AC$

$B \rightarrow BA \mid A$

$A \rightarrow 011 \mid 0$

$C \rightarrow CD \mid D$

$D \rightarrow 01$

(d)  $S \rightarrow BC$

$B \rightarrow AB \mid A$

$A \rightarrow 011 \mid 1$

$C \rightarrow DC \mid D$

$D \rightarrow 01$

**Q.39** The CNF for given CFG is:

$$S \rightarrow AACD$$

$$A \rightarrow aAb \mid \epsilon$$

$$C \rightarrow aC \mid a$$

$$D \rightarrow aDa \mid bDb \mid \epsilon$$

(a)  $S \rightarrow CD \mid AC \mid CX_a \mid AT_1 \mid AU_1$

$$A \rightarrow W_1X_a \quad A \rightarrow X_bX_a$$

$$C \rightarrow C \ X_a \quad X_a \rightarrow a$$

$$X_b \rightarrow b \quad U_1 \rightarrow CD$$

$$V_1 \rightarrow AC \quad D \rightarrow X_aY_1 \mid X_bZ_1$$

$$T_1 \rightarrow AT_2, \quad T_2 \rightarrow CD$$

(b)  $S \rightarrow CD \mid CA \mid CX_a \mid b$

$$A \rightarrow W_1X_a \quad W_1 \rightarrow X_bA$$

$$A \rightarrow X_5X_a \quad C \rightarrow CX_a$$

$$D \rightarrow X_aX_b \mid X_bX_c$$

$$X_a \rightarrow a \quad X_b \rightarrow b$$

(c)  $S \rightarrow CD \mid AC \mid X_aC \mid a$

$$A \rightarrow X_aW_1 \quad W_1 \rightarrow AX_b$$

$$A \rightarrow X_aX_b \quad C \rightarrow X_aC \mid a$$

$$S \rightarrow AT_1 \mid AU_1 \mid AV_1$$

$$T_1 \rightarrow AT_2 \quad U_1 \rightarrow CD$$

$$V_1 \rightarrow AC \quad T_2 \rightarrow CD$$

$$D \rightarrow X_aY_1 \mid X_bZ_1$$

$$Y_1 \rightarrow DX_a \quad Z_1 \rightarrow DX_b$$

$$D \rightarrow X_aX_b \mid X_bX_c$$

$$X_a \rightarrow a \quad X_b \rightarrow b$$

(d)  $S \rightarrow DC \mid CA \mid CX_a \mid a$

$$A \rightarrow W_1X_a \quad W_1 \rightarrow AX_aX_b$$

$$A \rightarrow X_aX_b \quad C \rightarrow X_aC$$

$$D \rightarrow X_aY_1 \mid X_bZ_1$$

$$Y_1 \rightarrow DX_a \mid X_bD$$

$$D \rightarrow X_aX_b \mid X_aX_b$$

$$X_a \rightarrow a \quad X_b \rightarrow b$$

**Q.40** The CFG equivalent to the following PDA is:

$$M = \{ \{q_0, q_1\}, \{0\}, \{R\}, \delta, q_0, R, \phi \}$$

$$\text{where } \delta(q_0, 0, R) = \{(q_1, R)\}$$

$$\delta(q_1, 0, R) = \{(q_1, R)\}$$

$$\delta(q_1, \epsilon, R) = \{(q_1, \epsilon)\}$$

(a)  $S \rightarrow \{q_1, R, q_0\}$

$$[q_1, R, q_0] \rightarrow 0[q_1, R, q_1]$$

$$[q_1, R, q_1] \rightarrow 0[q_1, R, q_1] | \epsilon$$

(b)  $S \rightarrow \{q_0, R, q_1\}$

$$[q_0, R, q_1] \rightarrow 0[q_1, R, q_1]$$

$$[q_1, R, q_1] \rightarrow 0[q_1, R, q_1] | \epsilon$$

(c)  $S \rightarrow [q_0, R, q_1][q_1, R, q_0]$

$$[q_0, R, q_1] \rightarrow 0[q_1, R, q_1]$$

$$[q_0, R, q_1] \rightarrow 0[q_1, R, q_1]$$

$$[q_1, R, q_1] \rightarrow \epsilon | 0$$

$$[q_1, R, q_0] \rightarrow 0[q_1, R, q_1]$$

(d)  $S \rightarrow \{q_1, R, q_0\}$

$$[q_1, R, q_0] \rightarrow 0[q_1, R, q_1]$$

$$[q_1, R, q_1] \rightarrow 0[q_1, R, q_1]$$

**Q.41** Any string of terminals that can be generated by the following CFG satisfy which of the given choices?

$$S \rightarrow XY$$

$$X \rightarrow aX \mid bX \mid a$$

$$Y \rightarrow Ya \mid Yb \mid a$$

(a) Has no consecutive a's or b's

(b) Has atleast two a's

(c) Has atleast one 'b'

(d) Should end in 'a'

**Q.42** Consider the grammar

$$S \rightarrow PQ \mid SQ \mid PS$$

$$P \rightarrow x$$

$$Q \rightarrow y$$

To get string of n terminals, the number of productions (i.e. steps in derivation) to be used is

(a)  $n^2$

(b)  $2n$

(c)  $2n + 1$

(d)  $2n - 1$

**Q.43** Consider the grammar G

$$S \rightarrow AB$$

$$A \rightarrow aAA \mid \epsilon$$

$$B \rightarrow bBB \mid \epsilon$$

If  $G_1$  is constructed from G after eliminating  $\epsilon$ -productions, then  $G_1$  is given by

(a)  $S \rightarrow AB$

$$A \rightarrow aAA \mid aA \mid a$$

$$B \rightarrow bBB \mid bB \mid b$$

(b)  $S \rightarrow AB \mid A \mid B$

$$A \rightarrow aAA \mid aA \mid a$$

$$B \rightarrow bBB \mid bB \mid b$$

(c)  $S \rightarrow AB \mid A \mid B$

$$A \rightarrow aAA \mid aA$$

$$B \rightarrow bBB \mid bB$$

(d)  $S \rightarrow AB$

$$A \rightarrow aAA \mid aA$$

$$B \rightarrow bBB \mid bB$$

**Q.44** The following context free grammar

$$S \rightarrow aB \mid bA$$

$$A \rightarrow a \mid aS \mid bAA$$

$$B \rightarrow b \mid bS \mid aBB$$

generates strings of terminals that have

(a) odd number of a's and even number of a's

(b) even number of a's and even number of b's

(c) odd number of a's and odd number of b's

(d) equal number of a's and b's

## GATE QUESTIONS

**Q.45** FORTRAN is a

[GATE 1987]

(a) Regular language

(b) Context-free language

(c) context-sensitive language

(d) None of the above

**Q.46** A context-free grammar is ambiguous if [GATE 1987]

- (a) the grammar contains useless non-terminals
- (b) it produces more than one parse tree for some sentence
- (c) Some production has two non-terminals side by side on the right-hand side.
- (d) None of the above

**Q.47** Context-free languages and regular languages are both closed under the operation(s) of :

[GATE 1989]

- (a) Union
- (b) Intersection
- (c) Concatenation
- (d) Complementation

**Q.48** Which of the following problems are undecidable? [GATE 1989]

- (a) Membership problem in context-free languages
- (b) Whether a given context-free language is regular
- (c) Whether a finite state automaton halts on all inputs
- (d) Membership problem for type 0 languages

**Q.49** Context-free languages are [GATE 1992]

- (a) closed under union
- (b) closed under complementation
- (c) closed under intersection
- (d) closed under Kleen closure

**Q.50** For a context-free grammar, FOLLOW (A) is a set of terminals that can appear immediately to the right of non-terminal A in some "sentential" form. We define two sets LFOLLOW (A) and RFOLLOW(A) by replacing the word "sentential" by "left most sentential" and "right most sentential" respectively in the definition of FOLLOW (A). [GATE 1992]

Which of the following statements is/are true?

- (a) FOLLOW (A) and FOLLOW (A) may be different
- (b) FOLLOW (A) and RFOLLOW (A) are always the same
- (c) All the three sets are identical
- (d) All the three sets are different

**Q.51** If G is a context-free grammar and w is a string of length  $\ell$  in L(G), how long is a derivative of w in G, if G is Chomsky normal form?

[GATE 1992]

- (a)  $2\ell$
- (b)  $2\ell+1$
- (c)  $2\ell - 1$
- (d)  $1\ell$

**Q.52** Which of the following features cannot be captured by context-free grammars?

[GATE 1994]

- (a) Syntax of if-then-else statements
- (b) Syntax of recursive procedures
- (c) Whether a variable has been declared before its use
- (d) Variable names of arbitrary length

**Q.53** Consider a grammar with the following productions :

$$S \rightarrow a \alpha b \mid b \alpha c \mid aB$$

$$S \rightarrow \alpha S \mid b$$

$$S \rightarrow \alpha bb \mid ab$$

$$S \alpha \rightarrow bdb \mid b$$

The above grammar is

- (a) Context free
- (b) Regular
- (c) Context sensitive
- (d) LR(k)

**Q.54** Define a context free language  $L \subseteq \{0, 1\}^*$  init ( $L$ ) = { $u/vv \in L$  for some  $v$  in  $\{0, 1\}^*$ } (in other words, init ( $L$ ) is the set of prefixes of  $L$ ). Let  $L$  { $w/w$  is nonempty and has an equal number of 0's and 1's}. Then init ( $L$ ) is

[GATE 1996]

- (a) The set of all binary strings with unequal number of 0's and 1's
- (b) The set of all binary string including the null string
- (c) The set of all binary strings with exactly one more 0 than the number of 1's or one more 1 than the number of 0's
- (d) None of the above

**Q.55** Which of the following statements is false?

[GATE 1996]

- (a) The Halting problem of Turing machines is undecidable
- (b) Determining whether a context-free grammar is ambiguous is undecidable
- (c) Given two arbitrary context-free grammar  $G_1$  and  $G_2$  it is undecidable whether  $L(G_1) = L(G_2)$
- (d) Given two regular grammars  $G_1$  and  $G_2$  it is undecidable whether  $L(G_1) = L(G_2)$

**Q.56** In the following grammar

$$X ::= X \oplus \frac{Y}{Y}$$

$$Y ::= Z * \frac{Y}{Z}$$

$$Z ::= id$$

Which of the following is true? [GATE 1997]

[1-Mark]

- (a) ' $\oplus$ ' is left associative while ' $*$ ' is right associative
- (b) Both ' $\oplus$ ' and ' $*$ ' are left associative
- (c) ' $\oplus$ ' is right associative while ' $*$ ' is left associative
- (d) None of the above

**Q.57** A grammar that is both left and right recursive for a non-terminal, is

[GATE 1999]

[2-Marks]

- (a) Ambiguous
- (b) Unambiguous
- (c) Information is not sufficient to decide whether it is ambiguous or unambiguous
- (d) None of the above

**Q.58** If  $L_1$  is context free language and  $L_2$  is a regular language which of the following is/are false?

[GATE 1999]

- [2-Marks]

- (a)  $L_1 - L_2$  is not context free
- (b)  $L_1 \cap L_2$  is context free
- (c)  $\sim L_1$  is context free
- (d)  $\sim L_2$  is regular

**Q.59** Given the following expression grammar :

$$E \rightarrow E * F \mid F + E \mid F$$

$$F \rightarrow F - F \mid id$$

Which of the following is true? [GATE 2000]

[2-Marks]

- (a) '\*' has higher precedence than '+'
- (b) '-' has higher precedence than '\*'
- (c) '+' and '-' have same precedence
- (d) '+' has higher precedence than '\*'

**Q.60** Let  $L$  denotes the language generated by the grammar  $S \rightarrow 0S0/00$ .

Which of the following is true? [GATE 2000]

[1-Mark]

- (a)  $L = 0^+$
- (b)  $L$  is regular but not  $0^+$
- (c)  $L$  is context free but not regular
- (d)  $L$  is not context free

**Q.61** Which of the following statements is true?

[GATE 2001]

[1-Mark]

- (a) If a language is context free it can always be accepted by a deterministic pushdown automaton.
- (b) The union of two context free languages is context free
- (c) The intersection of two context free languages is context free
- (d) The complement of a context free language is context free

**Q.62** Which of the following is true? [GATE 2002]

[2-Marks]

- (a) The complement of a recursive language is recursive
- (b) The complement of a recursively enumerable language is recursively enumerable
- (c) The complement of a recursive language is either recursive or recursively enumerable
- (d) The complement of a context free language is context free

- Q.63** The C language is [GATE 2002] [2-Marks]
- A context free language
  - A context sensitive language
  - A regular language
  - Parsable fully only by a Turing machine
- Q.64** Let  $G = (\{S\}, \{a, b\}, R, S)$  be a context free grammar where the rule set R is
- $$S \rightarrow aSb \mid SS \mid \epsilon$$
- Which of the following statements is true? [GATE 2003] [2-Marks]
- $G$  is not ambiguous
  - There exist  $x, y \in L(G)$  such that  $xy \notin L(G)$
  - There is a deterministic pushdown automaton that accepts  $L(G)$
  - We can find a deterministic finite state automaton that accepts  $L(G)$
- Q.65** Which one of the following statements is FALSE? [IT-GATE 2004] [1 Mark]
- There exist context free languages such that all the context free grammars generating them are ambiguous
  - An unambiguous context free grammar always has a unique parse tree for each string of the language generated by it.
  - Both deterministic and non-deterministic pushdown automata always accept the same set of languages.
  - A finite set of strings from one alphabet is always a regular language.
- Q.66** Let  $M = (K, \Sigma, \Gamma, \Delta, s, F)$  be a pushdown automaton, where  $K = \{s, f\}$ ,  $F = \{f\}$ ,  $\Sigma = \{a, b\}$ ,  $\Gamma = \{a\}$  and  $\Delta = \{((s, a, \epsilon), (s, a)), ((s, b, \epsilon), (s, a)), ((s, a, \epsilon), (f, \epsilon)), ((f, a, a), (f, \epsilon)), ((f, b, a), (f, \epsilon)))\}$ .
- Which one of the following strings is not a number of  $L(M)$ ? [IT-GATE 2004] [2-Marks]
- aaa
  - aabab
  - baaba
  - bab
- Q.67** The language  $\{0^n 1^n 2^n \mid 1 \leq n \leq 10^6\}$  is: [IT-GATE 2005] [1 Mark]
- regular
  - context free but not regular
  - context free but its complement is not context free
  - not context free
- Q.68** Let  $P$  be a non-deterministic pushdown automaton (NPDA) with exactly one state,  $q$ , and exactly one symbol,  $Z$ , in its stack alphabet. State  $q$  is both the starting as well as the accepting state of the PDA. The stack is initialized with one  $Z$  before the start of the operation of the PDA. Let the input alphabet of the PDA be  $\Sigma$ . Let  $L(P)$  be the language accepted by the PDA by reading a string and reaching its accepting state. Let  $N(P)$  be the language accepted by the PDA by reading a string and emptying its stack.
- Which of the following statements is TRUE? [IT-GATE 2005] [2-Marks]
- $L(P)$  is necessarily  $\Sigma^*$  but  $N(P)$  is not necessarily  $\Sigma^*$
  - $N(P)$  is necessarily  $\Sigma^*$  but  $L(P)$  is not necessarily  $\Sigma^*$
  - Both  $L(P)$  and  $N(P)$  is necessarily  $\Sigma^*$
  - Neither  $L(P)$  nor  $N(P)$  are necessarily  $\Sigma^*$
- Statement for Linked Answer Questions 69(A) and 69(B):**
- Consider the context-free grammar
- $$\begin{aligned} E &\rightarrow E + E \\ E &\rightarrow (E * E) \\ E &\rightarrow id \end{aligned}$$
- Where  $E$  is starting symbol, the set of terminals is  $\{id, (+, *)\}$ , and the set of non-terminals is  $\{E\}$ .
- Q.69** (A) Which of the following terminal strings has more than one parse tree when parsed according to the above grammar? [IT-GATE 2005] [2-Marks]
- $id + id + id + id$
  - $id + (id * (id * id))$
  - $(id * (id * id)) + id$
  - $((id * id + id) * id)$

**Q.69** (B) For the terminal string with more than one parse tree obtained as solution to Question 83a, how many parse trees are possible?

[IT-GATE 2005]

[2-Marks]

- (a) 5
- (b) 4
- (c) 3
- (d) 2

**Q.70** Consider the languages

$$L_1 = \{a^n b^n c^m \mid n, m > 0\} \text{ and } L_2 = \{a^n b^m c^n \mid n, m > 0\}$$

Which of the following statements is FALSE?

[GATE 2005]

[2-Marks]

- (a)  $L_1 \cap L_2$  is a context-free language
- (b)  $L_1 \cup L_2$  is a context-free language
- (c)  $L_1$  and  $L_2$  are context-free language
- (d)  $L_1 \cap L_2$  is a context sensitive language

**Q.71** Consider the languages

$$L_1 = \{ww^R \mid w \in \{0, 1\}^*\}$$

$$L_2 = \{w \# w^R \mid w \in \{0, 1\}^*\}, \text{ where } \# \text{ is a special symbol}$$

$$L_3 = \{ww \mid w \in \{0, 1\}^*\}$$

Which one of the following is TRUE?

[GATE 2005]

[2-Marks]

- (a)  $L_1$  is a deterministic CFL
- (b)  $L_2$  is a deterministic CFL
- (c)  $L_3$  is a CFL, but not a deterministic CFL
- (d)  $L_3$  is a deterministic CFL

**Q.72** Let  $L_1 = \{0^{n+m} 1^n 0^m \mid n, m \geq 0\}$ ,  $L_2 = \{0^{n+m} 1^n 0^m \mid n, m \geq 0\}$  and  $L_3 = \{0^{n+m} 1^n 0^{n+m} 0^{n+m} \mid n, m \geq 0\}$ . Which of these languages are NOT context free?

[GATE 2006]

- (a)  $L_1$  only
- (b)  $L_3$  only
- (c)  $L_1$  and  $L_2$
- (d)  $L_2$  and  $L_3$

[1-Mark]

**Q.73** Consider the following statements about the context-free grammar,

$$G = \{S \rightarrow SS, S \rightarrow ab, S \rightarrow ba, S \rightarrow \epsilon\}$$

1.  $G$  is ambiguous.
2.  $G$  produces all strings with equal number of a's and b's.
3.  $G$  can be accepted by a deterministic PDA.

Which combination below expresses all the true statements about  $G$ ? [GATE 2006]

[2-Marks]

- (a) 1 only
- (b) 1 and 3 only
- (c) 2 and 3 only
- (d) 1, 2 and 3

**Q.74** For  $s \in (0 + 1)^*$  let  $d(s)$  denote the decimal value of  $s$  (e.g.  $d(101) = 5$ ).

Let  $L = \{s \in (0 + 1)^* \mid d(s) \bmod 5 = 2 \text{ and } d(s) \bmod 7 \neq 4\}$

Which one of the following statements is true?

[GATE 2006]

[2-Marks]

- (a)  $L$  is recursively enumerable, but not recursive
- (b)  $L$  is recursive, but not context-free
- (c)  $L$  is context-free, but not regular
- (d)  $L$  is regular

**Q.75** Let  $L_1$  be regular language,  $L_2$  be a deterministic context-free language and  $L_3$  a recursively enumerable, but not recursive, language. Which one of the following statements is false?

[GATE 2006]

[2-Marks]

- (a)  $L_1 \cap L_2$  is a deterministic CFL
- (b)  $L_3 \cap L_1$  is recursive
- (c)  $L_1 \cup L_2$  is context free
- (d)  $L_1 \cap L_2 \cap L_3$  is recursively enumerable

**Q.76** Which of the following problems is undecidable?  
**[GATE 2007]**

**[1-Mark]**

- (a) Membership problem for CFGs
- (b) Ambiguity problem for CFGs
- (c) Finiteness problem for FSAs
- (d) Equivalence problem for FSAs

**Q.77** The language  $L = \{0^i 2^i \mid i \geq 0\}$  over the alphabet  $\{0, 1, 2\}$  is  
**[GATE 2007]**

**[2-Marks]**

- (a) not recursive
- (b) is recursive and is a deterministic CFL
- (c) is a regular language
- (d) is not a deterministic CFL but a CFL

**Q.78** Match the List-I with List-II and select the correct answer using the codes given below the lists:  
**[GATE 2008]**

**[2-Marks]**

List I	List II
A. Checking that identifiers are declared before their use	1. $L = \{a^n b^m c^n d^m \mid n \geq 1, m \geq 1\}$
B. Number of formal parameters in the declaration of a function agrees with the number of actual parameters in use of that function	2. $X \rightarrow X b X \mid X c X \mid d X f \mid g$
C. Arithmetic expressions with matched pairs of parentheses	3. $L = \{wcw \mid w \in (a \mid b)^*\}$
D. Palindrome	4. $X \rightarrow b X b \mid c X c \mid \varepsilon$

**Codes:**

- | <b>A</b> | <b>B</b> | <b>C</b> | <b>D</b> |
|----------|----------|----------|----------|
|----------|----------|----------|----------|
- (a) 1 3 2 4
  - (b) 3 1 4 2
  - (c) 3 1 2 4
  - (d) 1 3 4 2

**Q.79** If  $L$  and  $\bar{L}$  are recursively enumerable then  $L$  is  
**[GATE 2008]**

- (a) regular
- (b) context-free
- (c) context-sensitive
- (d) recursive

**Q.80** Which of the following are decidable?  
**[GATE 2008]**

**[1-Mark]**

1. Whether the intersection of two regular languages is infinite
2. Whether a given context-free language is regular
3. Whether two push-down automata accept the same language
4. Whether a given grammar is context-free
  - (a) 1 and 2
  - (b) 1 and 4
  - (c) 2 and 3
  - (d) 2 and 4

**Q.81** Which of the following statements are true?  
**[GATE 2008]**

**[2-Marks]**

1. Every left-recursive grammar can be converted to a right-recursive grammar and vice-versa
2. All  $\varepsilon$ -production can be removed from any context-free grammar by suitable transformations
3. The language generated by a context-free grammar all of whose productions are of the form  $X \rightarrow w$  or  $X \rightarrow wY$  (where,  $w$  is a string of terminals and  $Y$  is a non-terminal), is always regular
4. The derivation trees of strings generated by a context-free grammar in Chomsky Normal Form are always binary trees
  - (a) 1, 2, 3 and 4
  - (b) 2, 3 and 4 only
  - (c) 1, 3 and 4 only
  - (d) 1, 2 and 4 only

- Q.82** Let  $L = L_1 \cap L_2$ , where  $L_1$  and  $L_2$  are languages as defined below:

$$L_1 = \{a^m b^m c a^n b^n \mid m, n \geq 0\}$$

$$L_2 = \{a^i b^j c^k \mid i, j, k \geq 0\} \quad [\text{GATE } 2009]$$

Then  $L$  is: [2-Marks]

- (a) not recursive
- (b) regular
- (c) context-free but not regular
- (d) recursively enumerable but not context-free

## ANSWER KEY

1	d	2	a	3	c	4	b	5	d
6	a	7	c	8	c	9	d	10	c
11	b	12	a	13	d	14	d	15	d
16	b	17	d	18	c	19	c	20	c
21	d	22	a	23	b	24	a	25	c
26	b	27	c	28	a	29	d	30	b
31	d	32	a	33	b	34	b	35	a
36	d	37	b	38	d	39	c	40	b
41	b	42	d	43	b	44	d	45	b
46	b	47	a, c	48	b	49	a, d	50	d
51	b	52	a	53	d	54	b	55	d
56	a	57	c	58	a, c	59	a	60	c
61	b	62	b	63	a	64	c	65	c
66	c	67	a	68	d	69(a,b)	a,a	70	a
71	b	72	d	73	b	74	d	75	b
76	b	77	b	78	c	79	d	80	b
81	c	82	c						

## SOLUTIONS

### S.1 (d)

The given grammar is for palindromes of even length.

Option (a) and (b) are false as they are not palindromes whereas (c) is false as it is an odd length palindrome.

### S.2 (a)

The new production has made the grammar to accept palindromes of odd length. Hence I, II and IV will be accepted.

$S \rightarrow a|b$  allows odd length palindromes.

### S.3 (c)

It can be easily predicted by seeing the given productions.

Option (a) and (d) are eliminated because the given production can generate strings containing 'a' and 'bbb'.

Option (b) can be eliminated as it is not true for every strings.

#### S.4 (b)

Option (a), (c) & (d) are eliminated as they cannot generate 'aba' which is a valid string and contains consecutive a's and b's.

#### S.5 (d)

Not every language can be converted to GNF or CNF. The very first thing, that the language must not contain is the null string ' $\epsilon$ ' to convert into CNF or GNF.

#### S.6 (a)

Grammar in (a) generates the set  $\{a^n b^n, n = 1, 2, 3, \dots\}$  which is not regular.

(b) & (c) being left linear and right linear respectively, should have equivalent regular expressions.

#### S.7 (c)

The given grammar is for the language containing equal number of a's and b's. So a and b will not be accepted.

#### S.8 (c)

The given grammar is for the string of the type  $a^p b^p$  where p is an odd number i.e. = 1, 3, 5, 7.

#### S.10 (c)

By definition of right linear & left linear grammar.

#### S.12 (a)

The correct answer is,

$$L = (0.0)^+ \rightarrow \text{concatenation.}$$

Here,  $\epsilon$  is not in the language and zeroes occurs in even numbers.

#### S.13 (d)

By seeing the productions of the given grammar we can find that the grammar is type-0.

Hence the language generated by this grammar will also be type-0 language as mentioned in the definition of type-0 grammar.

#### S.14 (d)

By the property of CFG

#### S.16 (b)

Context free grammar is not closed under complement, intersection or difference.

#### S.18 (c)

Context free language are closed under union, kleen closure.

#### S.19 (c)

Theorem 1: If  $L = N(P_N)$  for some PDA  $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$  then there is a PDA  $P_F$  such that  $L = L(P_F)$ .

Theorem 2: Let  $L$  be  $L(P_F)$  for some PDA  $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$ . Then there is a PDA  $P_N$  such that  $L = N(P_N)$ .

From these two theorems, we can say that,

$$L_D = L_E$$

#### S.20 (c)

The grammar given in (c) is for generating strings containing more a's than b's. It can be checked by generating valid and invalid strings.

#### S.21 (d)

Consider the productions of grammar G as given below:

$$S \rightarrow aAb \mid bAa \mid aSa \mid bSb$$

$$A \rightarrow \epsilon \mid Aa \mid Ab$$

(Hence production ab, ba...) so not correct.

Actually the given grammar is for generating non-palindromes. Hence, the strings in option (a), (b) & (c) won't be accepted as they are palindrome.

#### S.22 (a)

A derivation of abbaaba would look like,

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abbAaba \Rightarrow abbAaaba \Rightarrow abbaaba$$

It can be checked similarly that other strings cannot be derived.

#### S.23 (b)

$$\begin{array}{ll} S \rightarrow AB, & S \rightarrow AB \\ \quad \rightarrow aab & \quad \rightarrow aaab \end{array}$$

$\therefore$  IV is true.

Also aab can be derived in 2 ways,

$$\begin{array}{ll} S \rightarrow AB & S \rightarrow AB \\ \rightarrow aaB & \rightarrow aB \\ \rightarrow aab & \rightarrow aab \end{array}$$

$\therefore$  II is true.

#### S.24 (a)

B is a useless symbol. Since B is not deriving a string of terminals. After deleting all productions in which B is present we get (a).

#### S.25 (c)

First we will derive the R.E. for the given language

$$R = (a + b)^* a (a + b)^* a (a + b)^*$$

Now,  $(a + b)^*$  can be represented using the grammar,

$$A \rightarrow aA \mid bA \mid \epsilon$$

Hence, the required grammar is

$$S \rightarrow AaAaA$$

$$A \rightarrow aA \mid bA \mid \epsilon$$

#### S.26 (b)

The given regular expression is for language containing string with any numbers of a's followed by any number b's. After 'b' we cannot again generate b. This condition is satisfied only by the grammar given in (b).

#### S.27 (c)

The given grammar has the start symbol S which is the only variable.  $\Sigma = \{a\}$ .

There is only one production

$$i.e. S \rightarrow SS$$

i.e. we cannot generate any string.

I is false as it produces strings with aa.

#### S.28 (a)

A and B are directly nullable because they have productions with  $\epsilon$  as the body. Then, we find S is nullable, because the production  $S \rightarrow AB$  has a body consisting of nullable symbols only. Thus, all the three variables are nullable.

#### S.29 (d)

Both the given statements are false. The correct statements are

I. If the grammar  $G_1$  is constructed from G by eliminating  $\epsilon$ -productions, then  $L(G_1) = L(G) - \{\epsilon\}$

II. If the grammar  $G_2$  is constructed from grammar G by eliminating unit productions, then  $L(G_1) = L(G)$

#### S.30 (b)

The CFLs are closed under Union, Concatenation and Closure.

The CFLs are not closed under intersection.

#### S.31 (d)

The grammar is Semi-Thur grammar.

Semi-Thur grammar is nothing but Unrestricted grammar of Type 0 grammar.

There are no restrictions on the productions of grammar of this type.

This type of grammar permits productions of the form

$$\alpha \rightarrow \beta \text{ with } \alpha \neq \epsilon$$

where  $\alpha$  &  $\beta$  are sentential form, in the above grammar there are no restrictions on the productions therefore it is called as Semi-Thur or Type 0 or Unrestricted grammar.

#### S.32 (a)

II is wrong  $\because$  it can generate aabb

III is wrong  $\because$  it can't generate  $\epsilon$  also

I, IV are right as they can generate the desired string.

#### S.33 (b)

Though option (a) also seems to be correct, but it also includes strings like abab which is invalid as per the given regular expression, so option (b) correct.

#### S.34 (b)

$$L = \{a^m b^n c^{m+n} \mid m, n \geq 1\}$$

A DPDA can accept this language. When a's and b's are in input, these are pushed into the stack and when c's are finished, the stack is empty, this means  $n_c(w) = n_a(w) + n_b(w)$  and the string is accepted. Else it is rejected. Now since a DPDA exists, the language is context free.

Clearly the language is not regular, since we must count and compare c's with a's and b's which cannot be done by any FA.

### S.35 (a)

The given grammar is most accurately said to be **chomsky type 3** as from the definition of type 3 grammar, the given productions are of the form—

$$A \rightarrow a$$

$$A \rightarrow aB \quad \text{where } A, B \in V_N \text{ and } a \in \Sigma$$

And also there are no null ( $\epsilon$ ) production.

### S.36 (d)

II generates strings like xxxyy, which are not supposed to be. III generates strings like xyy, which are not supposed to be. I can be verified to generate all the strings in L and only those.

### S.37 (b)

- Rules :
1.  $S \rightarrow SS$
  2.  $S \rightarrow (S)$
  3.  $S \rightarrow \epsilon$

The derivation is as given below

1.  $S \rightarrow S$  ... Rule (2)
2.  $S \rightarrow (SS)$  ... Rule (1)
3.  $S \rightarrow ((S)S)$  ... Rule (2)
4.  $S \rightarrow (((SS)S))$  ... Rule (1)
5.  $S \rightarrow (((S)S)S)$  ... Rule (2)
6.  $S \rightarrow ((( ))S)$  ... Rule (3)
7.  $S \rightarrow ((( )(S)))$  ... Rule (2)
8.  $S \rightarrow ((( )( ))S)$  ... Rule (3)
9.  $S \rightarrow ((( )( ))(S))$  ... Rule (2)
10.  $S \rightarrow ((( )( ))( ))$  ... Rule (3)

Hence, no. of step is required is 10.

### S.38 (d)

Let L be the language corresponding to the regular expression

$$(011 + 1)^*(01)^*$$

The productions

$$A \rightarrow 011 \mid 1$$

generate the language  $\{011, 1\}$ .

Thus,  $B \rightarrow AB \mid A$

$$A \rightarrow 011 \mid 1$$

with B as the start symbol to generate the language  $\{011, 1\}^*$ . Similarly, we can use

$$C \rightarrow DC \mid D$$

$$D \rightarrow 01$$

to derive  $\{01\}^*$  from the start symbol C. Finally we generate the concatenation of the two languages by adding the productions  $S \rightarrow BC$ . The final grammar has start symbol S, auxiliary variables A, B, C and D and productions

$$S \rightarrow BC$$

$$B \rightarrow AB \mid A$$

$$A \rightarrow 011 \mid 1$$

$$C \rightarrow DC \mid D$$

$$D \rightarrow 01$$

### S.39 (c)

Eliminating  $\epsilon$ -productions.

- $$S \rightarrow AACD \mid ACD \mid AAC \mid CD \mid AC \mid C$$
- $$A \rightarrow aAb \mid ab$$
- $$C \rightarrow aC \mid a$$
- $$D \rightarrow aDa \mid bDb \mid aa \mid bb$$

Eliminating unit productions

$$S \rightarrow aC \mid a$$

and delete  $S \rightarrow C$

Restricting the right sides of productions to single terminals or strings of two or more variables.

- $$S \rightarrow AACD \mid ACD \mid AAC \mid CD \mid X_aC \mid a$$
- $$A \rightarrow X_aAX_b \mid X_aX_b$$
- $$C \rightarrow X_aC \mid a$$
- $$D \rightarrow X_aDX_a \mid X_bDX_b \mid X_aX_b \mid X_bX_b$$
- $$X_a \rightarrow a$$
- $$X_b \rightarrow b$$

### S.40 (b)

Let  $G = (V, T, P, S)$  be the required grammar. here,

I.  $T = \{0\}$

II.  $V :$

Let S be the start symbol remaining variables:

$$[q_0, R, q_0], [q_0, R, q_1]$$

$$[q_1, R, q_0], [q_1, R, q_1]$$

## III. Productions P :

## S-productions

$$S \rightarrow [q_0, R, q_0][q_0, R, q_1]$$

$$[q_0, R, q_0] \rightarrow 0[q_1, R, q_0] \text{ using } \delta(q_0, 0, R) \\ = \{(q_1, R)\}$$

$$[q_0, R, q_1] \rightarrow 0[q_1, R, q_1] \text{ using } \delta(q_0, 0, R) \\ = \{(q_1, R)\}$$

$$[q_1, R, q_0] \rightarrow 0[q_1, R, q_0] \text{ using } \delta(q_1, 0, R) \\ = \{(q_1, R)\}$$

We cannot create a production for the variable  $[q_1, R, q_0]$  using  $\delta(q_1, \epsilon, R) = \{(q_1, \epsilon)\}$

$$[q_1, R, q_1] \rightarrow 0[q_1, R, q_1] \text{ using } \delta(q_1, 0, R) \\ = \{(q_1, R)\}$$

$[q_1, R, q_1] \rightarrow \epsilon$  using  $\delta(q_1, \epsilon, R) = \{(q_1, \epsilon)\}$   $[q_1, R, q_0]$  is a useless symbol. After deleting all the productions in which  $[q_1, R, q_0]$  is present  $[q_0, R, q_0]$  will be useless. So the reduced grammar is as follows

$$S \rightarrow [q_0, R, q_1]$$

$$[q_0, R, q_1] \rightarrow 0[q_1, R, q_1]$$

$$[q_1, R, q_1] \rightarrow 0[q_1, R, q_1] \mid \epsilon$$

## S.41 (b)

Consider the string aa,

$$S \rightarrow XY$$

$$S \rightarrow aY$$

$$S \rightarrow aa$$

i.e. 'aa' can be generated from the given grammar. This eliminates option (a) & (c).

Also,  $S \rightarrow XY$

$$S \rightarrow aY \quad (\because X \rightarrow a)$$

$$S \rightarrow aYb \quad (\because Y \rightarrow Yb)$$

$$S \rightarrow aab \quad (\because Y \rightarrow a)$$

Eliminate option (d) as string ends in b.

## S.42 (d)

It can be directly inferred from the grammar that the no. of productions required is  $2n - 1$ . But we can also find the answer by method of elimination.

Lets start with S and develop some string.

$$1. S \rightarrow PQ \quad \text{using } S \rightarrow PQ$$

$$2. S \rightarrow xQ \quad \text{using } P \rightarrow x$$

$$3. S \rightarrow xy \quad \text{using } Q \rightarrow y$$

Here,  $n = 2$  and no. of productions required is 3

Now check,  $n^2$  i.e.  $2^2 \neq 3$   $\therefore$  Eliminate (a)

$2n = 2 \cdot 2 = 4 \neq 3$   $\therefore$  Eliminate (b)

$2n + 1 = 5 \neq 3$   $\therefore$  Eliminate (c)

$\therefore$  Now remaining option is (d).

$$2n - 1 = 4 - 1 = 3.$$

## S.43 (b)

Let us construct the product of grammar  $G_1$ . First consider  $S \rightarrow AB$ . All the symbols of the body are nullable, so there are four ways we could choose present or absent for A and B, independently. However, we are not allowed to choose to make all symbols absent, so there are only three productions

$$S \rightarrow AB \mid A \mid B$$

Next, consider production  $A \rightarrow aAA$ . The second and third positions hold nullable symbols, so again there are four choices of present/absent. In this case, all four choices are allowable, since the nonnullable symbol a will be present in any case. Our four choices yield productions:

$$A \rightarrow aAA \mid aA \mid Aa \mid a$$

Note that the two middle choices happen to yield the same production, since it doesn't matter which of the A's we eliminate so we decide to eliminate one of them. Thus, the final grammar  $G_1$  will only have three productions for A.

Similarly, the production B yield for  $G_1$ :

$$B \rightarrow bBB \mid bB \mid b$$

Thus, the grammar  $G_1$  is given by

$$S \rightarrow AB \mid A \mid B$$

$$A \rightarrow aAA \mid aA \mid Aa \mid a$$

$$B \rightarrow bBB \mid bB \mid b$$

## S.44 (d)

$$S \rightarrow aB \rightarrow aaBB \rightarrow aabB \rightarrow aabb$$

So (c) is wrong. We have

$$S \rightarrow aB \rightarrow ab$$

So (b) is wrong.

A careful observation of the productions will reveal a similarity. Change A to B, B to A, a to b and b to a. The new set of productions will be the same as the original set. So (a) is false.

#### S.46 (b)

Context free grammar is ambiguous if it produces more than one parse tree for some sentence. A grammar can lead to the generation of identical string or sentence by two or more different derivations. Such a grammar is known as ambiguous grammar.

A grammar is said to be ambiguous grammar in which sentence can be generated by two or more derivations corresponding to different trees.

#### S.47 (a, c)

Context-free languages and Regular languages are both closed under operation of **Union** and **Concatenation** by using Theorems.

#### S.49 (a, d)

Context-free languages are closed under **union** and **Kleene closure**, (By property)

#### S.52 (a)

Syntax of if-then-else statements cannot be captured by context-free grammar.

In if-then-else, to execute statements in 'then' or 'else' block, we have to check the 'if' block (condition). But such procedure cannot be implemented using context free grammar.

#### S.53 (d)

The grammar given is LR(K).

In this type of grammar, the extra information is added into the state by redefining items to include a terminal symbol as 2<sup>nd</sup> component.

For eg.  $[A \rightarrow \alpha\beta, a]$

where  $A \rightarrow \alpha\beta$  is a production, a is a terminal or right end marker \$. Such an object is called as LR(K) item.

#### S.54 (b)

Init (L) is the set of prefixes of L  
 $L \{w/w is non empty and has equal number of 0's and 1's\}$

#### S.55 (d)

Given two regular grammars  $G_1$  and  $G_2$ . It is undecidable whether  $L(G_1) = L(G_2)$

#### S.56 (a)

$\oplus$  is left associative while  $*$  is right associative.

#### S.57 (c)

The grammar  $A \rightarrow (A), A \rightarrow a$  is ambiguous. It is left and right recursive.

The grammar  $A \rightarrow A \times B, B \rightarrow y B, A \rightarrow a, B \rightarrow b$  is not ambiguous though it is left and right recursive.

#### S.58 (a, c)

$L_1$  is context free language

$L_2$  is regular language

$L_1 - L_2$  is context free. (By theorem)

$\sim L_1$  is not context free. (By theorem)

#### S.59 (a)

For the following expression grammar :

$E \rightarrow E^* F \mid F + E \mid F$

$F \rightarrow F - F \mid id$

\* has higher precedence than +.

#### S.60 (c)

L is context free but not regular because by rule of context free grammar

- (i) The left hand side of the production should contain only one non terminal
- (ii) Start symbol can appear on right hand side also.

$S \rightarrow 0S0 \mid 00$

#### S.61 (b)

The union of two context free languages is context free. By using theorem.

#### S.62 (b)

The complement of a recursively enumerable language is recursively enumerable, by using theorem.

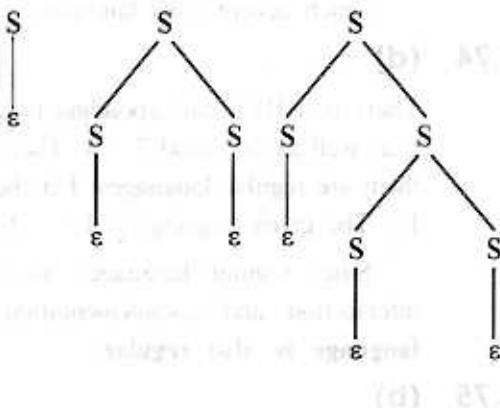
**S.63 (a)**

The C language is context free language.

**S.64 (c)**

The grammar is  $S \rightarrow aSb \mid SS \mid \epsilon$

- (a) G is not ambiguous is false, since  $\epsilon$  which belongs to  $L(G)$ , has infinite number of derivation trees, which makes G ambiguous. Some derivation trees are given below:



- (b) There exists  $x, y \in L(G)$  such that  $xy \notin L(G)$  is false, since  $S \rightarrow SS$  can be used to derive  $xy$ , whenever  $x \in L(G)$  and  $y \in L(G)$ .

- (c) is true since this language is  $L(G) = \{w \mid n_a(w) = n_b(w) \text{ and } n_a(v) \geq n_b(v) \text{ where } v \text{ is any prefix of } w\}$ .

This language happens to be deterministic context free language

$\therefore$  There exists a DPDA that accepts it.

- (d) is false, as the given language is not regular.  
 $\therefore$  no DFA exists to accept it.

**S.65 (c)**

A set of languages accepted by deterministic and non-deterministic pushdown automaton are different, so both deterministic as well as non-deterministic automaton not always accept the same set of language.

**S.66 (c)**

baaba is not a number of language accepted by given PDA.

**S.67 (a)**

$$L = \{0^n 1^n 2^n \mid 1 \leq n \leq 10^6\}$$

Every finite language is regular above language is finite hence it is regular language.

**S.68 (d)**

Let the language accepted by empty the stack and go to final state are same.

So if we not define any transaction on particular  $L(p)$  symbol then  $N(p)$  will not accept even it is part of  $\Sigma^*$ .

**S.69**

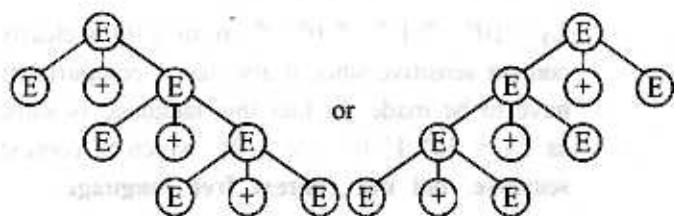
$$E \rightarrow E + E$$

$$E \rightarrow (E^*E)$$

$$E \rightarrow id$$

**(A) (a)**

option (a) can be produced by more than one parse tree as

**S.70 (a)**

$L_1$  and  $L_2$  are context free languages and therefore  $L_1 \cup L_2$  is a context free language.  $L_1 \cap L_2$  may or may not be context free, since CFL's are not closed under intersection. Now let us look at  $L_1 \cap L_2$

$$L_1 \cap L_2 = \{a^n b^n c^n \mid n > 0\}$$

Which is clearly not context free but is context sensitive.

**S.71 (b)**

$L_1$  is CFL but not a DCFL, since accepting  $ww^R$  necessarily involves finding the middle of the string, which involves non-determinism.

$\therefore$  (a) is false.

$L_2$  is a DCFL, is true, since "#" is a special symbol and middle of string can now be surely found by using "#", thereby eliminating the need for non-deterministic guessing. So (b) is true.

$L_3$  is a CSL and not a CFL. So (c) is false.

$L_3$  is not a DCFL either. so (d) is false.

### S.72 (d)

$L_1 = \{0^{n+m} 1^n 0^m \mid n, m \geq 0\}$  is context free (first  $n+m$  0's are pushed into the stack, then for each of the 1's following, we will pop the 0's in the stack one by one until we reach at the end of the word if the stack is empty then the word is accepted).

$L_2 = \{0^n + m 1^n + m 0^m \mid n, m \geq 0\}$  is not context free, since two comparisons have to be made here to determine if  $w \in L_2$ .

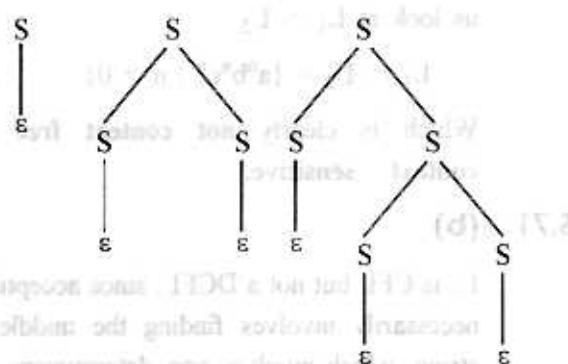
1. 0's and 1's are equal.
2. Since  $m < n+m$ , we have to ensure that, 0's which come after  $0^{n+m} 1^{n+m}$ , are less in number, compared to  $n+m$ .

$L_3 = \{0^{n+m} 1^{n+m} 0^{n+m} \mid n, m \geq 0\}$  is clearly context sensitive since it also has 2 comparisons have to be made. In fact this language is same as  $L_3 = \{0^x 1^x 0^x \mid x \geq 0\}$ , which is context sensitive, but not context free language.

### S.73 (b)

$$G = \{S \rightarrow SS, S \rightarrow ab, S \rightarrow ba, S \rightarrow \epsilon\}$$

1. "G is ambiguous", is true since " $\epsilon$ " has infinite number of derivations some of which are shown below:



2. "G produces all strings with equal number of a's and b's, is false since  $L(G) = (ab + ba)^*$  and this language cannot produce all strings with equal number of a's and b's. For example aabb has equal number of a's and b's but aabb  $\notin L(G)$ .

3. "G can be accepted by a DPDA" is true, since  $L(G) = (ab + ba)^*$  and this is a regular language, since we have written a regular expression for it. Since every regular language is also a DCFL, a DPDA exists, which accepts this language.

### S.74 (d)

There is a DFA corresponding to  $d(s) \bmod 5 = 2$  as well as  $d(s) \bmod 7 = 4$ . Therefore, both of them are regular languages. Let them be  $L_1$  and  $L_2$ . The given language is  $L_1 \cap L_2$ .

Since regular languages are closed under intersection and complementation, the given language is also regular.

### S.75 (b)

Given  $L_1$  be regular,  $L_2$  is DCFL and  $L_3$  is RE but not REC.

1. " $L_1 \cap L_2$  is a DCFL" is true since DCFL's are closed under regular intersection.
2. " $L_3 \cap L_1$  is recursive" is false since  $L_3$  is RE but not REC and therefore  $L_3 \cap L_1$  is surely RE according to closure of RE under regular intersection, but we cannot be sure that  $L_3 \cap L_1$  is REC.
3. " $L_1 \cup L_2$  is CFL" is true, since  $L_2$  is DCFL and hence a CFL and  $L_1$  is regular. Therefore, by closure of CFL's on regular union, we can say that  $L_1 \cup L_2$  is surely CFL.
4. " $L_1 \cap L_2 \cap L_3$  is RE" is true since all three are surely RE and RE languages are closed under intersection.

### S.76 (b)

- (a) Membership problem for CFG's is decidable (CYK algorithm exists).
- (b) Ambiguity problem of CFG's is undecidable.

- (c) Finiteness problem of FSA's is decidable, since there exists an algorithm to check if a given regular language L is finite or infinite.
- (d) Equivalence problem for FSA's is decidable, since there exists an algorithm to check whether  $L(M_1) = L(M_2)$  or not.

**S.77 (b)**

$\{0^i 2^{i+1} \mid i \geq 0\}$  is a DCFL since a DPDA can accept this language. 0's are pushed into the stack and then when 2 appears in input, state is changed and immediately after that for every 1, a zero is popped out of the stack. In the end, if stack has start stack symbol only, then the string is accepted. Else it is rejected. Since every DCFL is recursive, we can say that the language is recursive, and is a DCFL.

**S.78 (c)**

In  $\{wcw \mid w \in (a+b)^*\}$ , leftmost w is the identifier checking and rightmost w is the use of w.

In  $a^n b^m c^n d^m$  the actual parameters are  $a^n$  and  $b^m$  and the formal parameters are  $c^n$  and  $d^m$  such that the number of arguments of a and b are equal to c and d respectively.

The grammar  $\{X \rightarrow XbX \mid XcX \mid dXf \mid g\}$  generates the arithmetic expressions with matched pair of parentheses.

The grammar  $\{X \rightarrow bXb \mid cXc \mid \epsilon\}$  generates the palindromic string.

**S.79 (d)**

There is a theorem which states that whenever L and  $\bar{L}$  are both RE, then both will be recursive.

**S.80 (b)**

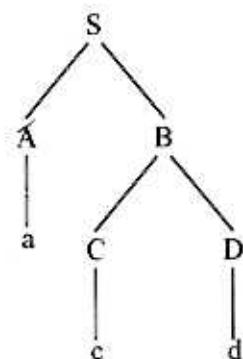
- "Intersection of two regular languages is infinite" is decidable, since we can construct a product DFA of the 2 given DFAs and then, using the algorithm to check finiteness or infiniteness on this DFA, we can solve the problem.
- "Whether a given CFL is regular" is undecidable.

- "Whether two pda's accept the same language" is undecidable, since equivalence of CFL's is undecidable.
- "Whether a given grammar is context-free" is decidable since we can easily check using a TM, whether the LHS of every production has a single variable and, then it is a CFG, else it is not a CFG.

**S.81 (c)**

- True
- All  $\epsilon$ -productions can be removed from only context free grammars that produce  $\lambda$ -free CFL's.  
If  $\lambda \in L(G)$ , then all  $\epsilon$ -productions cannot be successfully removed. So 2 is false.
- True
- True, since in chomsky normal form, every production is of the form of  $A \rightarrow BC$  or  $A \rightarrow a$ .

An example of a binary tree generated by CNF derivation is shown below:

**S.82 (c)**

$$L_1 = \{a^m b^m c^m a^n b^n \mid m, n \geq 0\}$$

$$L_2 = \{a^i b^j c^k \mid i, j, k \geq 0\}$$

$$L_1 \cap L_2$$

Here closure property should not be used, as  $L_1$  &  $L_2$  are some specific languages.

Let us find  $L = L_1 \cap L_2$

Any string belonging to both must have equal number of a's and b's followed by a single c followed by no a's or b's; which is the only string allowed by both  $L_1$  and  $L_2$ .

$$\text{i.e., } L = L_1 \cap L_2 = \{a^m b^m c\}$$

Now this is clearly context-free, but not regular.

(c) 18.2

This part involves of two numbers - 14 & 5 with 3 numbers that contains odd terms.



Moduloing w.r.t. the mod (10) i.e. 14 & 5 will be having same remainder as 4 & 5 respectively.

and 3

every third binary element in some diff. p to DB  $\leftarrow f$  to check w.r.t. to all numbers

$f \in \mathbb{Z}$

(d) 18.3  
An interesting fact stand is to ignore the whole number of numbers.



(c) 28.2

$$(0 \leq n \leq 704, 2^{n+1} \leq 10^6)$$

$$\Rightarrow 0 \leq 2^n \leq 500 \times 10^6 \Rightarrow 2^n \leq 5000000$$

$$\Rightarrow n \leq 24$$

For base of 10 binary strings upto 24 required strings are 1 & 0.

(c) 28.2

Let us consider the problem statement (c).  
It is asked to consider an array that is initially filled with zeros. A natural action may be to add 1 at each index. In this case, we can do it in  $O(n^2)$  time complexity.

(d) 28.2

Recall Q90 is same. By Q90 we know that sum of indices will be added with previous value. Hence if we do  $f(i) = \text{sum}(0:i)$ , then we have to add  $i + 1$  to  $f(i)$  and then add it to  $f(i+1)$ . Hence we have to add previous result & current cell just with index  $i$  and then add with  $i+1$ ? But as  $i$  is not yet updated so when  $f(i)$  is to be updated it is required

(c) 28.2

What we can do? If we do  $f(i) = \text{sum}(0:i)$  it will still be a constant time complexity

but  $f'(i) = \text{sum}(0:i) + i + 1$  will be constant time complexity. If this  $f'(i)$  is updated, then it will be  $f'(i+1)$  and it has to be updated to constant time complexity.

So  $f'(i) = f(i) + i + 1$  is of constant time complexity. So we can calculate it in linear time.

Example:  $f(0) = 0, f(1) = 1, f(2) = 3, f(3) = 6, f(4) = 10, f(5) = 15, f(6) = 21, f(7) = 28$

(D) 28.2

Consider individual digits of the number  $n = 1000$  and how many such 3H digit are there in  $n$ ?

(c) 28.2

engaged integers are to non-negative  
integers whose addition is minimum &  
having 7 unit in AHD number is natural  
leads to multiple of 7, since multiple of 10  
or 100 will be succeeded by second  
multiple of 7, similarly all other digits  
are also multiple of 7. Hence the number of  
such numbers are 7.

# 1.4

## TURING MACHINE

### LEVEL-1

**Q.1** State which of the following statements are True?

- For any unrestricted grammar  $G$ , there is a Turning Machine  $T$  with  $L(T) = L(G)$
- Every recursive language is recursively enumerable.
- If  $L_1$  and  $L_2$  are recursively enumerable languages over  $\Sigma$ , then  $L_1 \cap L_2$  and  $L_1 \cup L_2$  are also recursively enumerable.
  - True, False, False
  - False, True, False
  - False, False, False
  - None of these

**Q.2** If every string at a language can be determined, whether it is legal or illegal in finite time, the language is called

- intractable
- undecidable
- decidable
- non-deterministic

**Q.3** The encoding function 'e' that encodes the TM into binary string must be

- many-many
- many-one
- one-many
- one-one

**Q.4** Which of the following can be recognized using a TM

- Unary multiplication
  - Palindrome recognition
  - GCD calculation
  - SQUARE (SQUARE(.,(x)...))
- where  $\text{SQUARE}(x) = x^2$
- I, II, III, IV
  - Only I
  - Only II, III
  - Only I, IV

**Q.5** Which of the following is true

- I. A TM can be constructed with one tape and having only two internal states.
  - II. Any TM with  $m$  symbols and  $n$  states can be simulated by another TM with just two states and  $(4mn + m)$  symbols.
  - III. Any TM with  $m$  symbols and  $n$  states can be simulated by another TM with just two symbols and less than  $8mn$  states.
  - IV. There is no algorithm that takes an arbitrary string  $A$  over an alphabet  $\Sigma$  as the input and determines whether or not string  $A$  halts for every input.
- (a) Only IV is true  
 (b) Only I, II are true  
 (c) Only I, III, IV are true  
 (d) None of above are true

**Q.6** Consider the transition table of TM that accepts the language  $L = \{0^n 1^n \mid n \geq 1\}$

State	0	1	X	Y	B
start $\rightarrow q_0$	$(q_1, X, R)$	-	-	$(q_3, Y, R)$	-
$q_1$	$(q_1, 0, R)$	$(q_0, Y, L)$	-	$(q_1, Y, R)$	-
$q_2$	$(q_2, 0, L)$	-	$(q_0, X, R)$	$(q_2, Y, L)$	-
$q_3$	-	-	-	$(q_3, Y, R)$	$(q_4, B, R)$
$q_4$	-	-	-	-	-

Working of above Turing machine:

Starting at the left end of the input, it repeatedly changes a 0 to an X and moves to the right over whatever 0's and Y's it sees, until it comes to a 1. It changes the 1 to Y, and moves left, over Y's and 0's, until it finds an X. At that point, it looks for a 0, immediately to the right, and if it finds one, changes it to X and repeats the process, changing a matching 1 to Y. Now, using this information, answer the questions given below.

If the language to be accepted is  $\{0^n 1^n \mid n \geq 0\}$  then for which state a change in transition table will occur?

- (a)  $q_4$
- (b)  $q_3, q_2$
- (c)  $q_1$
- (d)  $q_0$

**Q.7** A pushdown automaton (PDA) behaves like a Turing machine (TM) when the number of auxiliary memory it has is,

- (a) 3
- (b) 2
- (c) 1
- (d) 0

**Q.8** Let  $L$  be a language recognizable by a finite automaton. The language Reverse ( $L$ ) =  $\{w \mid w \text{ is the reverse of } v \text{ where } v \in L\}$  is a

- (a) Context-free language
- (b) Context sensitive language
- (c) Recursively Enumerable language
- (d) Regular language

**Q.9** A nondeterministic pushdown acceptor (NPDA) is defined by the seven-tuple

$$M = (Q, \Sigma, T, \delta, q_0, z, F)$$

where transition function  $\delta$  is \_\_\_\_\_.

- (a)  $\delta : Q \times (\Sigma \cup \{\lambda\}) \times T \rightarrow Q$
- (b)  $\delta : Q \times (\Sigma \cup \{\lambda\}) \times T \rightarrow z$
- (c)  $\delta : Q \times (\Sigma \cup \{\lambda \times T\}) \rightarrow Q$
- (d)  $\delta : Q \times (\Sigma \cap \{\lambda\}) \times T \rightarrow Q$

**Q.10** Turing Machine (TM) is more powerful than FSM (Finite State Machine) because

- (a) it has no finite state
- (b) it has the capacity to remember arbitrarily long sequences of input symbols
- (c) tape direction is confined to one direction
- (d) none of these

**Q.11** I. If there is a TM which, when applied to any problem in the class, always eventually terminates with correct yes / no answer, we call the problem \_\_\_\_\_.

II. If there is a TM which, when applied to any problem in the class, always eventually terminates the correct answer when the answer is yes and may or may not terminate when the answer is no, we call the problem \_\_\_\_\_.

- (a) Solvable, unsolvable
- (b) Solvable, partially solvable
- (c) Unsolvable, partially solvable
- (d) Partially solvable, solvable

**Q.12** Which of the following statements about Turing Machine applications are correct?

- (a) To generate a language
- (b) To recognize a language
- (c) To evaluate a non-negative integer function
- (d) All of the above

**Q.13** Given a Turing machine T and a step-counting function f, is the language accepted by T in Time(f)? This decision problem is

- (a) solvable
- (b) uncertain
- (c) unsolvable
- (d) none of these

**Q.14** If T is a TM recognizing language L and T reads every symbol in the input string, then  $\tau_T(n) \geq 2n + 2$ . Then any language that can be accepted by a TM T with  $\tau_T(n) = 2n + 2$  is

- (a) uncertain
- (b) not regular
- (c) regular
- (d) none of these

**Q.15** How many tuple are there in turing machine.

- (a) 6 tuple
- (b) 5 tuple
- (c) 4 tuple
- (d) 7 tuple

**Q.16** A machine M is consist of

1. a finite nonempty set of states Q.
2. a finite nonempty set of tape symbols  $\Gamma$ .
3.  $b \in \Gamma$  is the blank.
4.  $\Sigma$  is a nonempty set of input symbols and is a subset of  $\Gamma$  and  $b \notin \Sigma$ .
5. the transition function ' $\delta$ '
6. the initial state  $q_0$ .
7. the set of final states, F.

Identify the type of machine.

- (a) Turing Machine
- (b) Moore Machine
- (c) Mealy Machine
- (d) PSM

## LEVEL-2

**Q.17** Which is the correct order of the given grammars from most restrictive to most general?

- (a) Recursively enumerable, Context-free, Context-sensitive, Regular
- (b) Regular, Context-free, Context-sensitive, Recursively enumerable
- (c) Recursively enumerable, Context-sensitive, Context-free, Regular
- (d) Regular, Context-sensitive, Context-free, Recursively enumerable

**Q.18** The diagonalization language,  $L_d$  is a

- (a) Non - recursively - enumerable (non-RE) language
- (b) Recursively enumerable but not recursive language
- (c) Recursive language
- (d) None of the above

**Q.19** Which of the following languages are recursive?

I. L where L is recursively enumerable

II.  $\bar{L}$  where L is recursive

III L where L and  $\bar{L}$  are recursively enumerable.

- (a) Only II and III
- (b) Only III
- (c) Only I
- (d) I, II & III

**Q.20** Consider the transition table of the TM given below:

$\delta$	0	1	b
$q_0$	$q_0\ 0\ R$	$q_1\ 1\ R$	$q_2\ b\ R$
$q_1$	$q_1\ 0\ R$	$q_0\ 1\ R$	-
$q_2$	-	-	-

Which of the following strings will not be accepted?

- (a) 010101
- (b) 101010
- (c) 100101
- (d) (a), (b) and (c)

- Q.21** Regarding the power of recognizing the languages, which of the following statements is false?
- Multitape TMs are equivalent to single tape TMs.
  - Non-deterministic TMs are equivalent to deterministic Pushdown Automata (DPDA)
  - NDPDA are equivalent to DPDA
  - The NDFA are equivalent to DFA
- Q.22** Which of the following is true for the transition function  $\delta$  of a TM? (Notations have their usual meanings used in the text).
- $\delta : (p, Y, D) \rightarrow (q, X, D)$
  - $\delta : (p, Y, D) \rightarrow (q, X)$
  - $\delta : (q, X) \rightarrow (p, D)$
  - $\delta : (q, X) \rightarrow (p, Y, D)$
- Q.23** Non-recursively-enumerable languages are accepted by
- Finite Automaton with memory
  - Turing machine
  - Finite Automaton
  - None of these
- Q.24** Which of the following statements are correct?
- TM permits two-way communication between machine and user
  - FSM cannot multiply arbitrarily long numbers
  - Neither (a) nor (b)
  - Both (a) and (b)

## LEVEL-3

### Common Data For Questions 25 to 28:

Assume  $X_1$  will always be the symbol 0,  $X_2$  will be 1, and  $X_3$  will be B, the blank.

Refer direction L as  $D_1$  and direction R as  $D_2$ .  
Encode the transition rule,

$\delta(q_i, X_j) = (q_k, X_\ell, D_m)$  for some integers i, j, k,  $\ell$ , m by the string

$$0^i 10^j 10^\ell 10^m$$

Let the TM M =  $(\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$

- Q.25** If the transition function  $\delta$  of a given TM consists of the rules

$$\delta(q_1, 1) = (q_3, 0, R)$$

$$\delta(q_1, 0) = (q_2, 1, R)$$

$$\delta(q_2, 1) = (q_2, 0, R)$$

$$\delta(q_2, B) = (q_3, 1, R)$$

$$\delta(q_3, B) = (q_3, 1, L)$$

then which of the following is a valid code of a transition rule of the TM?

- 1000101001
- 0011000101
- 01010100001
- 0001000100010010

- Q.26** Write the transition rule if the code is 0100100010100

- $\delta(q_1, 1) = (q_2, 0, R)$
- $\delta(q_1, 0) = (q_3, 1, R)$
- $\delta(q_1, 1) = (q_3, 0, R)$
- $\delta(q_1, 0) = (q_2, 1, R)$

- Q.27** If a code for the entire TM M consists of all codes for the transitions, in some order, separated by pairs of 1's:

$$C_1 1 1 C_2 1 1 C_3 1 1 \dots C_{n-1} 1 1 C_n$$

where each of the C's is the code for one transition of M; and if the given machine has only five transition rules, how many 1's will be there in the code for the TM?

- 33
- 30
- 13
- None of these

- Q.28** Encode  $\delta(q_3, 1) = (q_2, 0, R)$

- 00010010010100
- 00010101001010
- 01001000001000
- 0101000000010

- Q.29** Consider the transition table of the TM given below:

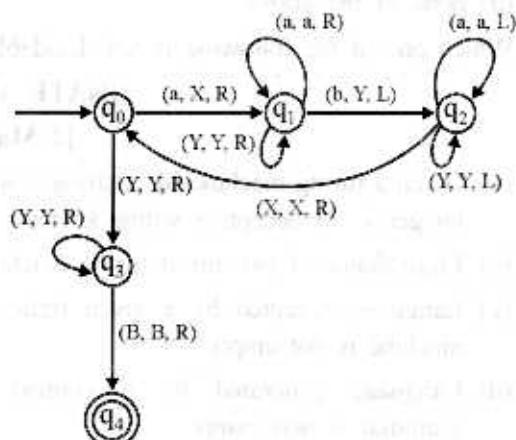
$\delta$	0	1	b
$q_0$	$q_0 \ 0 \ R$	$q_1 \ 1 \ R$	$q_2 \ b \ R$
$q_1$	$q_1 \ 0 \ R$	$q_0 \ 1 \ R$	--
$q_2$	--	--	--

The given Turing machine accept

- (a) strings over  $\{0, 1\}$  containing even no. of 1's
- (b) strings over  $\{0, 1\}$  containing even no. of 1's and odd no. of 0's
- (c) set of all even palindromes over  $\{0, 1\}$
- (d) None of the above

- Q.30** If we want to show a problem not to be R.E., where  $L_d$  is not R.E. and  $L_u$  is R.E. then
- (a) only  $L_u$  can be used
  - (b) only  $L_d$  can be used
  - (c) either  $L_u$  or  $L_d$  can be used
  - (d) none of them

- Q.31** The transition diagram for a TM is shown below:



Here : B  $\rightarrow$  Blank,

X, Y : tape alphabets

R  $\rightarrow$  Move right

L  $\rightarrow$  Move left

Which of the following strings can be simulated by the above TM

- (a) aaabb
- (b) aabbaba
- (c) bbaaaaa
- (d) aabb

- Q.32** If  $M_N$  is a non-deterministic Turing machine, then there is a deterministic Turing machine  $M_D$  such that

- (a)  $L(M_N) \cup L(M_D) = L(M_D)$
- (b)  $L(M_N) \cap L(M_D) = \emptyset$
- (c)  $L(M_N) \cap L(M_D) = L(M_N)$
- (d) both (a) and (c)

### Common Data For Questions 33 & 34:

Consider the transition table of TM that accepts the language  $L = \{0^n 1^n \mid n \geq 1\}$

State	0	1	X	Y	B
start $\rightarrow$ $q_0$	$(q_1, X, R)$	--	--	$(q_3, Y, R)$	--
$q_1$	$(q_1, 0, R)$	$(q_2, Y, L)$	--	$(q_1, Y, R)$	--
$q_2$	$(q_2, 0, L)$	--	$(q_0, X, R)$	$(q_2, Y, L)$	--
$q_3$	--	--	--	$(q_3, Y, R)$	$(q_4, B, R)$
$q_4$	--	--	--	--	--

Working of above Turing machine:

Starting at the left end of the input, it repeatedly changes a 0 to an X and moves to the right over whatever 0's and Y's it sees, until it comes to a 1. It changes the 1 to Y, and moves left, over Y's and 0's, until it finds an X. At that point, it looks for a 0, immediately to the right, and if its finds one, changes it to X and repeats the process, changing a matching 1 to Y. Now, using this information, answer the questions given below:

- Q.33** In state  $q_1$  machine can encounter \_\_\_\_\_ for any string  $\in (0, 1)^*$

- (a) B
- (b) X
- (c) Neither (a) nor (b)
- (d) Both (a) and (b)

- Q.34** If the input string is 00110, then in which state will the machine halt?

- (a)  $q_3$
- (b)  $q_2$
- (c)  $q_1$
- (d)  $q_0$

**Common Data For Questions 35 & 36:**

Assume  $X_1$  will always be symbol 0,  $X_2$  will be 1, and  $X_3$  will be B, the blank.

Refer direction L as  $D_L$  and direction R as  $D_R$ .

Encode the transition rule,

$\delta(q_i, X_j) = (q_k, X_\ell, D_m)$  for some integers i, j, k,  $\ell$ , m by the string

$$0^i 10^j 10^k 10^\ell 10^m$$

Let the TM M =  $(\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$

**Q.35** Encode  $\delta(q_3, 0) = (q_1, 1, R)$

- (a) 100101010010
- (b) 000101010010
- (c) 0001010100100
- (d) 0100100010100

**Q.36** Write the transition rule for the code

$$0001000100010010$$

- (a)  $\delta(q_3, B) = \delta(q_3, 1, L)$
- (b)  $\delta(q_3, B) = \delta(q_3, 1, R)$
- (c)  $\delta(q_2, 1) = \delta(q_1, 0, L)$
- (d)  $\delta(q_3, 0) = \delta(q_2, 0, R)$

**Q.37** We can define a restricted NPDA as one that can increase the length of the stack by at most one symbol in each move, so changing  $\delta$  as  $\delta' : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow 2^Q$ . The interpretation of this is that the range of d consists of sets of pair of the form  $(q_i, ab), (q_i, a)$  or  $(q_i, \lambda)$ . Select correct statement

- (a) For every NPDA M there exists such a restricted NPDA  $\hat{M}$  such that  $L(M) \leq L(\hat{M})$
- (b) For every NPDA M there exists such a unrestricted NPDA  $\hat{M}$  such that  $L(M) = L(\hat{M})$
- (c) For every NPDA M there exists such a restricted NPDA  $\hat{M}$  such that  $L(M) = L(\hat{M})$
- (d) For every NPDA M there exists such a restricted NPDA  $\hat{M}$  such that  $L(M) \geq L(\hat{M})$

**Q.38** A code for the entire TM M consists of all codes for the transitions, in some order, separated by pairs of 1's:

$$C_1 1 1 C_2 1 1 C_3 1 1 \dots C_{n-1} 1 1 C_n$$

where each of the C's is the code for one transition of M. If there are total 5 transition rules, then how many possible codes are there for the machine M?

- (a) 120
- (b) 240
- (c) 60
- (d) 1

## GATE QUESTIONS

**Q.39** It is undecidable whether : [GATE 1990]

- (a) an arbitrary Turing machine halts after 10 $\phi$  steps
- (b) a Turing machine prints a specific letter
- (c) a Turing machine computes the products of two numbers
- (d) none of the above

**Q.40** Which one of the following is not decidable?

[GATE 1997]  
[2-Marks]

- (a) Given a turing machine M, a string s and an integer k, M accepts s within k steps
- (b) Equivalence of two given turing machines
- (c) Language accepted by a given finite state machine is not empty
- (d) Language generated by a context free grammar is non empty.

**Q.41** Consider the following problem X :

Given a Turing machine M over the input alphabet  $\Sigma$ , any state q of M and a word  $w \in \Sigma^*$ , does the computation of M on w visit the state q?

Which of the following statements about X is correct? [GATE 2001]

- (a) X is decidable [2-Marks]
- (b) X is undecidable but partially decidable
- (c) X is undecidable and not even partially decidable
- (d) X is not a decision problem

- Q.42** A single tape Turing Machine M has two states  $q_0$  and  $q_1$ , of which  $q_0$  is the starting state. The tape alphabet of M is  $\{0, 1, B\}$  and its input alphabet is  $\{0, 1\}$ . The symbol B is the blank symbol used to indicate end of an input string. The transition function of M is described in the following table

	0	1	B
$q_0$	$q_1, 1, R$	$q_1, 1, R$	Halt
$q_1$	$q_1, 1, R$	$q_0, 1, L$	$q_0, B, L$

The table is interpreted as illustrated below.

The entry  $(q_1, 1, R)$  in row  $q_0$  and column 1 signifies that if M is in state  $q_0$  and reads 1 on the current tape square, then it writes 1 on the same tape square, moves its tape head one position to the right and transitions to state  $q_1$ .

Which of the following statements is true about M?

[GATE 2003]

[2-Marks]

- (a) M does not halt on any string in  $(0 + 1)^*$
- (b) M does not halt on any string in  $(00 + 1)^*$
- (c) M halts on all string ending in a 0
- (d) M halts on all string ending in a 1

- Q.43** Define languages  $L_0$  and  $L_1$  as follows

$$L_0 = \{\langle M, w, 0 \rangle \mid M \text{ halts on } w\}$$

$$L_1 = \{\langle M, w, 1 \rangle \mid M \text{ does not halts on } w\}$$

Here  $\langle M, w, t \rangle$  is a triplet, whose first component, M is an encoding of a Turing Machine, second component, w, is a string, and third component, t, is a bit.

Let  $L = L_0 \cup L_1$ . Which of the following is true?

[GATE 2003]

[2-Marks]

- (a) L is recursively enumerable, but  $\bar{L}$  is not
- (b)  $\bar{L}$  is recursively enumerable, but L is not
- (c) Both L and  $\bar{L}$  are recursive
- (d) Neither L nor  $\bar{L}$  is recursive enumerable

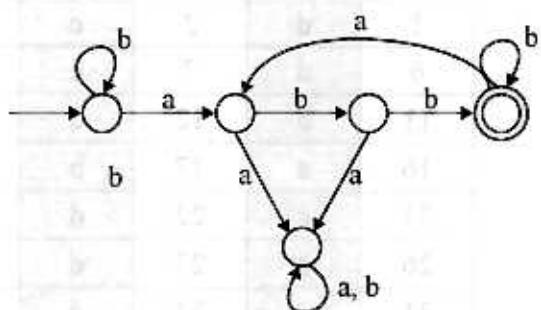
- Q.44** Consider three decision P<sub>1</sub>, P<sub>2</sub> and P<sub>3</sub>. It is known that P<sub>1</sub> is decidable and P<sub>2</sub> is undecidable. Which one of the following is TRUE?

[GATE 2005]

[2-Marks]

- (a) P<sub>3</sub> is decidable if P<sub>1</sub> is reducible to P<sub>3</sub>
- (b) P<sub>3</sub> is undecidable if P<sub>3</sub> is reducible to P<sub>2</sub>
- (c) P<sub>3</sub> is undecidable if P<sub>2</sub> is reducible to P<sub>3</sub>
- (d) P<sub>3</sub> is decidable if P<sub>3</sub> is reducible to P<sub>2</sub>'s complement

- Q.45** Consider the machine M



The language recognized by M is [GATE 2005]

[2-Marks]

- (a)  $\{w \in \{a, b\}^* \mid \text{every } a \text{ in } w \text{ is followed by exactly two } b's\}$
- (b)  $\{w \in \{a, b\}^* \mid \text{every } a \text{ in } w \text{ is followed by atleast two } b's\}$
- (c)  $\{w \in \{a, b\}^* \mid w \text{ contains the substring 'abb'}\}$
- (d)  $\{w \in \{a, b\}^* \mid w \text{ does not contain 'aa' as a substring}\}$

- Q.46** Let P(x) and Q(x) be arbitrary predicates. Which of the following statements is always TRUE?

[IT-GATE 2005]

[2-Marks]

- (a)  $((\forall x P(x)) \vee Q(x)) \Rightarrow ((\forall x(P(x)) \vee (\forall x Q(x)))$
- (b)  $((\forall x(P(x)) \Rightarrow Q(x))) \Rightarrow ((\forall x(P(x)) \Rightarrow (\forall x Q(x)))$
- (c)  $((\forall x(P(x)) \Rightarrow (\forall x Q(x))) \Rightarrow ((\forall x(P(x) \Rightarrow Q(x)))$
- (d)  $((\forall x(P(x))) \Leftrightarrow (\forall x Q(x))) \Rightarrow ((\forall x(P(x) \Leftrightarrow Q(x)))$

**Q.47** Which of the following is true for the language  $\{a^p \mid p \text{ is a prime}\}$ ? [GATE 2008]

[1-Mark]

- (a) It is not accepted by a Turing Machine
- (b) It is regular but not context-free
- (c) It is context-free but not regular
- (d) It is neither regular nor context-free, but accepted by a Turing machine

## ANSWER KEY

1	d	2	c	3	d	4	d	5	d
6	d	7	c	8	d	9	a	10	b
11	b	12	d	13	c	14	c	15	d
16	a	17	b	18	a	19	a	20	d
21	c	22	d	23	d	24	d	25	d
26	c	27	d	28	a	29	a	30	b
31	d	32	d	33	a	34	a	35	c
36	a	37	c	38	a	39	a, b	40	b
41	c	42	a	43	b	44	c	45	b
46	a	47	d						

## SOLUTIONS

### S.1 (d)

All the three given statements are true.

### S.3 (d)

The encoding function 'c' should be one-one, so that a string of 0's and 1's encodes atmost one TM.

### S.6 (d)

The only difference between the previous language and this language is the inclusion of ' $\epsilon$ ' i.e. null string. The new machine must also accept ' $\epsilon$ '. So in  $q_0$  if B occurs it should go to  $q_4$  (i.e. accepting state).

Therefore, the entry in  $q_0$  on B must be ( $q_4$ , B, R).

### S.7 (c)

A PDA with one stack can accept any language that a TM can accept.

### S.8 (d)

Since L is recognizable by a finite automaton, it is a regular language. The reversal of a regular language is regular.

### S.15 (d)

The formal notation for a TM is given by a 7-tuple,

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

where Q: The finite set of states

$\Sigma$ : The finite set of input symbols

$\Gamma$ : The complete set of tape symbols

- δ: The transition function  
 $q_0$ : The start state, a member of Q  
 B: The blank symbol  
 F: The set of final or accepting states

**S.16 (a)**

Symbolically, a turing machine M is 7-tuple viz.  
 $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ .

i.e. give 1, 2, ..., 7 as i/p and identify the type of definition.

**S.17 (b)**

Chomsky himself designated the four types as type 3, type 2, type 1 and type 0 from most restrictive to most general.

The Chomsky Hierarchy

Type	Language (Grammar)	
3	Regular	most restrictive
2	Context-free	
1	Context-sensitive	
0	Recursively enumerable	most general

**S.18 (a)**

Diagonalization language,  $L_d$  is the set of strings  $w_i$  such that  $w_i$  is not in  $L(M_i)$ . ( $w_i$  is the code for  $M_i$ )

And non-recursively-enumerable languages have no TM to accept them. Hence, we can say that  $L_d$  is a non-R.E. language.

**S.19 (a)**

Every recursive language is recursively enumerable but vice-versa is not always true. Hence, language I is not recursive.

We know that if L is recursive, so is  $\bar{L}$ . Hence, language of II is recursive.

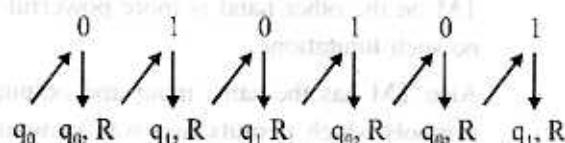
Also if L is R.E. language whose complement is R.E., then L is recursive.

Hence, language of III is also recursive.

**S.20 (d)**

Consider (a) 010101

Simulation: Since, here only one direction (the right) is used, we can simulate just as we did for FSMs.



Since the machine ends in  $q_1$  which is not an accepting state, we can say 010101 is not accepted.

Similarly (b) and (c) are also not acceptable. So the correct option is (d).

Alter:

The given turing machine accepts strings with even no. of 1's so all the three strings will not be accepted by this machine as they have odd no. of 1's.

**S.21 (c)**

A DPDA cannot accept the language of palindromes; whereas NDPDA can.

The switch from one type of move to the other (in the case of palindromes) should happen when we reach the middle of the string. However, without a symbol marking the middle explicitly, the DPDA has no way of knowing that the middle has arrived, whereas, NDPDA can guess and check whether the middle has arrived or not.

**S.22 (d)**

The transition function of a TM is  $\delta : (q, X) \rightarrow (p, Y, D)$ .

That is, the value of  $\delta(q, X)$  is a triple  $(p, Y, D)$ , where, p is the next state in Q

Y is the tape symbol

D is the direction, either L or R, standing for "left" or "right".

**S.23 (d)**

Non-recursively enumerable languages have no TM at all. And if TM cannot accept non-recursively enumerable languages then definitely Finite automaton cannot.

**S.24 (d)**

FSM has certain limitations.

It cannot multiply arbitrarily long numbers and check the well formedness of an arbitrarily long sequence of parenthesis.

TM on the other hand is more powerful and has no such limitations.

Also TM has the same input and output sets of symbols which permits two-way communication between machine and user. This enables the user to predetermine the functional matrix in order to carry out this task.

**S.25 (d)**

Here, we don't need to find the code for each of the transition rules.

Here, we notice that, all of  $i, j, k, \ell$  and  $m$  are at least one. Therefore, there will be no occurrences of two or more consecutive 1's within the code for a single transition. Hence, eliminate option (b).

Due to same reason and the format of the code  $0^i 1 0^j 1 0^k 1 0^\ell 1 0^m$ , we can say that the code cannot start and end with 1. Hence, eliminate (a) and (c).

**S.26 (c)**

The given code is  $0100100010100 = 0^1 1 0^2 1 0^1 1 0^2$

Comparing with  $0^i 1 0^j 1 0^k 1 0^\ell 1 0^m$ , we get

$i = 1, j = 2, k = 3, \ell = 1, m = 2$

$\therefore$  The transition rule is,

$$(q_1, X_2) = (q_3, X_1, D_2)$$

$$\Rightarrow (q_1, 1) = (q_3, 0, R)$$

$$[\because X_1 = 0, X_2 = 1, D_2 = 2]$$

**S.27 (d)**

For each transition rule, there will be only 4 1's, since the code is of the form

$$0^i 1 0^j 1 0^k 1 0^\ell 1 0^m$$

$\therefore$  There are 5 transitions, there will be  $5 \times 4 = 20$  1's. Also in the code for TM, each transition code is separated by two consecutive 1's. There will be four such pair for 5 transition rules. This adds another  $4 \times 2 = 8$  1's.

$$\therefore \text{Total no. of ones} = 20 + 8 = 28.$$

(b) S.2

**S.28 (a)**

Using the given information, the transition rule

$$\delta(q_3, 1) = (q_2, 0, R)$$

can be written as,

$$\delta(q_3, X_2) = (q_2, X_1, D_2)$$

Here,  $i = 3, j = 2, k = 2, \ell = 1, m = 2$

$\therefore$  the code is

$$= 0^i 1 0^j 1 0^k 1 0^\ell 1 0^m$$

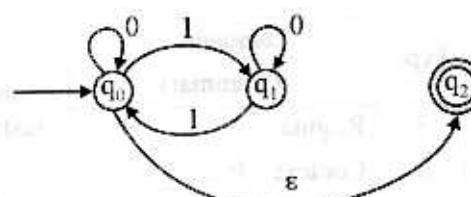
$$= 0^3 1 0^2 1 0^2 1 0^1 1 0^2$$

$$= 00010010010100$$

**S.29 (a)**

Since, the direction is only Right (R) and the TM doesn't change the symbol, we can treat it as an FSM. That is, the given TM can be represented by an equivalent FSM.

Transition Diagram:



Regular Expression:

$$0^* + 0^* (1 0^* 1)^*$$

This shows that it accepts strings with even number of 1's.

**S.30 (b)**

$L_d$  is not R.E. and  $L_u$  is R.E. The reduction of  $L_u$  to another problem P can be used to show there is no algorithm to solve P, regardless of whether or not P is R.E. However, reduction of  $L_d$  to P is not possible if P is not R.E.; so  $L_d$  cannot be used to show undecidability for those problems that are R.E. but not recursive. On the other hand, if we want to show a problem not to be R.E., only  $L_d$  can be used;  $L_u$  is useless since it is R.E.

**Note:**

**Universal Language:**

We define  $L_u$ , the universal language, to be the set of binary strings that encode a pair  $(M, w)$  where M is a TM with the binary input alphabet, and w is a string in  $(0 + 1)^*$ , such that w is in

$L(M)$ . That is,  $L_u$  is the set of strings representing a TM and an input accepted by that TM. And there is a TM  $U$ , often called the Universal Turing machine, such that  $L_u = L(U)$ . That means,  $L_u$  is R.E.

### S.31 (d)

$$\begin{aligned}
 q_0 a a b b &\rightarrow X q_1 a b b \\
 &\rightarrow X a q_1 b b \\
 &\rightarrow X q_2 a Y b \\
 &\rightarrow q_2 X a Y b \\
 &\rightarrow X q_2 a Y b \\
 &\rightarrow X q_0 a Y b \\
 &\rightarrow X X q_1 Y b \\
 &\rightarrow X X Y q_1 b \\
 &\rightarrow X X q_2 Y Y \\
 &\rightarrow X q_2 X Y Y \\
 &\rightarrow X X q_0 Y Y \\
 &\rightarrow X X Y q_3 Y \\
 &\rightarrow X X Y Y q_3 \\
 &\rightarrow X X Y Y q_3 B \\
 &\rightarrow X X Y Y q_4
 \end{aligned}$$

So, aabb is acceptable.

### S.32 (d)

The nondeterministic Turing machine (NTM) differs from the deterministic variety we have been studying by having a transition function  $\delta$  such that for each state  $q$  and tape symbol  $X$ ,  $\delta(q, X)$  is a set of triplets.

$\{(q_1, Y_1, D_1), (q_2, Y_2, D_2), \dots, (q_n, Y_n, D_n)\}$  when  $n$  is any finite integer. The NTM can choose any of the triplets to be next move.

But the non-deterministic TMs accept no languages not accepted by a deterministic TM. That is, if  $M_N$  is a non-deterministic TM, then there is a deterministic machine  $M_D$  such that

$$L(M_N) = L(M_D)$$

$$\text{i.e. } L(M_N) \cap L(M_D) = L(M_N) = L(M_D)$$

$$\text{and } L(M_N) \cup L(M_D) = L(M_N) = L(M_D)$$

### S.33 (a)

If the string is 001, then in  $q_2$ , machine will encounter B i.e. blank

State	string	Next state
$q_0$	001 ↑	$q_1$
$q_1$	X01 ↑	$q_1$
$q_1$	X01 ↑	$q_2$
$q_2$	X0Y ↑	$q_2$
$q_2$	X0Y ↑	$q_0$
$q_0$	X0Y ↑	$q_1$
$q_1$	XXY ↑	$q_1$
$q_1$	XXYB ↑	—

Here, in last step,  $q_1$  will encounter blank. There is no next step, neither currently it is in the accepting state. This is because the given string is invalid. Therefore, option (b) and (c) are eliminated.

Now, in  $q_1$ , it will never encounter X, because we enter  $q_1$  only after all the X's are on left side. That is,  $q_1$  expression is of the type  $X^*0^*Y^*1^*$  and we are at 0<sup>\*</sup> moving towards right skipping 0's and Y's and looking for 1.

### S.34 (a)

State	string	Next state
$q_0$	00110 ↑	$q_1$
$q_1$	X0110 ↑	$q_1$
$q_1$	X0110 ↑	$q_2$
$q_2$	X0Y10 ↑	$q_2$
$q_2$	X0Y10 ↑	$q_0$
$q_0$	X0Y10 ↑	$q_1$
$q_1$	XXY10 ↑	$q_1$
$q_1$	XXY10 ↑	$q_2$
$q_2$	XXYY0 ↑	$q_2$
$q_2$	XXYY0 ↑	$q_0$
$q_0$	XXYY0 ↑	$q_3$
$q_3$	XXYY0 ↑	$q_3$
$q_3$	XXYY0 ↑	—

In  $q_3$ , there is no move on 0.

**S.35 (c)**

$\because X_1 \Rightarrow 0, X_2 \Rightarrow 1, L \Rightarrow D_1$  and  $R \Rightarrow D_2$ , we can write the given transition rule as,

$$\begin{aligned}\delta(q_3, 0) &= (q_1, 1, R) \\ &\equiv \delta(q_3, X_1) = (q_1, X_2, D_2)\end{aligned}$$

- $\therefore i = 3, j = 1, k = 1, \ell = 2, m = 2$
- $\therefore$  code is,

$$\begin{aligned}0^i 10^j 10^k 10^\ell 10^m \\ = 0^3 10^1 10^1 10^2 10^2 \\ = \textbf{0001010100100}\end{aligned}$$

**S.36 (a)**

The given code is,

$$\begin{aligned}0001000100010010 \\ = 0^3 10^3 10^3 10^2 10^1\end{aligned}$$

$$\Rightarrow i = 3, j = 3, k = 3, \ell = 2, m = 1$$

$\therefore$  The transition function becomes,

$$\delta(q_3, X_3) = (q_3, X_2, D_1)$$

$$\therefore X_3 = B, X_2 = I, \text{ and } D_1 = L$$

the required transition rule is,

$$\delta(q_3, B) = (q_3, I, L)$$

Shortcut:

Consider the code 0001000100010010

Here, first 3 zeros imply  $q_3$ ,  $\therefore$  First part of the transition must be of the form  $\delta(q_3, ?)$

Hence option (c) rules out.

Also the last single '0' after 'I' shows that direction is  $D_1$  i.e. 'L'. Therefore, option (b) and (d) are eliminated.

Note that here we just looked at the end and start of the code to get the answer. We may get the answer by checking one part only. In the worst case also, it is not difficult to calculate the whole transition rule.

**S.37 (c)**

Here we use internal states to remember symbols to be put on the stack.

For example,

$$\delta(q_i, a, b) = \{(q_j, cde)\}$$

is replaced by

$$\delta(q_i, a, b) = \{(q_{je}, de)\}$$

$$\delta(q_{je}, \lambda, d) = \{(q_j, cd)\}$$

Since  $\delta$  can have only a finite number of elements and each can only add a finite amount of information to the stack, this construction can be carried out for any PDA.

**S.38 (a)**

Let  $C_1, C_2, C_3, C_4$  and  $C_5$  be the five codes for the 5 transition rules of the given machine.

Then, the code for the entire machine is given by,

$$C_1 1 1 C_2 1 1 C_3 1 1 C_4 1 1 C_5$$

Now here the order is not important.  $C_n$  can be replaced in any of the 5 places of  $C_5$ .

$\therefore$  There are total  $5!$  Combinations possible.

$$5! = 120.$$

Therefore, there are 120 possible codes for M. And any combination out of 120 will give the same M.

**S.39 (a, b)**

For a given configuration of a TM two cases arise :

1. The machine starting at this configuration will halt after a finite number of steps.
  2. The machine starting at this configuration never halts no matter how long it runs.
- $\therefore$  It is unsolvable that TM halts after  $10^6$  steps.
3. Can we decide whether a TM ever prints a given symbol  $a \in I$  of its alphabet? This is unsolvable.

**S.40 (b)**

We know that we can mechanically check for equivalence (a) Two combinational switching circuits (b) two sequential circuits (c) regular expression describing two FSM's. However, two TM's with the same alphabet cannot be checked for equivalence or inequivalence by an algorithm

**S.41 (c)**

Suppose  $w = \{0, 1\} \in \Sigma^*$ .

The problem is that whether or not M. M will enter state q on input {0, 1}.

For solving this problem, let us take a new turing machine M' which simulates M and has the additional transition given by

$$\delta(q, \varepsilon) = (q, 1, R).$$

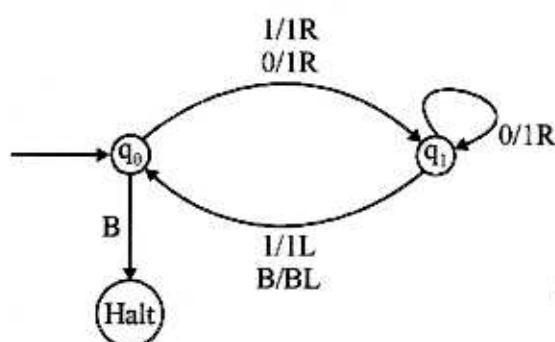
Then M enters q when it starts with a given tape configuration if and only if M' prints 1 when it starts with a given tape configuration. Hence the problem is undecidable and not even partially decidable.

### S.42 (a)

Transition function of M

	0	1	B
q <sub>0</sub>	q <sub>1</sub> , 1, R	q <sub>1</sub> , 1, R	Halt
q <sub>1</sub>	q <sub>1</sub> , 1, R	q <sub>0</sub> , 1, L	q <sub>0</sub> , B, L

The graphic representation of M is as follows,



The original configuration of tape of M is as follows

BBBq <sub>0</sub> 000....111....BBB	[S(q <sub>0</sub> , 0) = q <sub>1</sub> , 1, R]
BBBlq <sub>1</sub> 00....111....BBB	
BBBlq <sub>1</sub> 00....111....BBB	[S(q <sub>1</sub> , 0) = q <sub>1</sub> , 1, R]
BBBlq <sub>1</sub> 00....111....BBB	

If tape is finite then after K right move all 0's will be changed to 1's and M remain in state q<sub>1</sub>.

BBB111....q <sub>1</sub> 111....BBB	[S(q <sub>1</sub> , 1) = q <sub>0</sub> , 1, L]
BBB111....q <sub>0</sub> 111....BBB	
BBB111....lq <sub>1</sub> 111....BBB	[S(q <sub>0</sub> , 1) = q <sub>1</sub> , 1, R]
BBB111....q <sub>0</sub> 111....BBB	[S(q <sub>1</sub> , 1) = q <sub>0</sub> , 1, L]

So machine M moves one step right and one step left and goes in infinite loop so M can't reach a configuration  $\delta(q_0, B)$  and M doesn't halt on any string in  $(0 + 1)^+$ .

### S.43 (b)

If  $L_0 \cup L_1$  is recursively enumerable, it means we can find out for all  $\omega \in \Sigma^*$ , whether M halts or does not halt. This means that if  $L_0 \cup L_1$  is recursively enumerable, the halting problem would be decidable. But we know, that halting problem is undecidable. Therefore  $L = L_0 \cup L_1$  is not RE.

Since  $\bar{L} = (L_0 \cup L_1)^c = L_0^c \cap L_1^c = \emptyset$ , which is a regular language and hence is RE.

Therefore,  $\bar{L}$  is RE.

So,  $\bar{L}$  is RE but L is not.

### S.44 (c)

Note that in general if  $L_1 \leq L_2$ , then if  $L_1$  decidable and  $L_2$  undecidable  $\Rightarrow L_3$  is undecidable.

Given that  $P_1$  is decidable and  $P_2$  is undecidable.

Consider (a)  $P_3$  is decidable if  $P_1$  is reducible to  $P_3$  i.e.  $P_1 \leq_P P_3$

and if  $P_1$  is undecidable, then  $P_3$  would be undecidable.

But it is given that  $P_1$  is decidable, therefore we cannot use this theorem.

(a) is false.

Consider (b)  $P_3$  is undecidable if  $P_3$  is reducible to  $P_2$  i.e.  $P_3 \leq_P P_2$

Now if  $P_3$  is undecidable then  $P_2$  is undecidable but nothing is given regarding  $P_1$ .

Also if  $P_2$  is decidable, then  $P_3$  would be decidable, but it is given that  $P_2$  is undecidable, so we can't use this.

(b) is false.

Consider (c)  $P_3$  is undecidable if  $P_2$  is reducible to  $P_3$  i.e.  $P_2 \leq_P P_3$

Now if  $P_2$  is undecidable, this would mean  $P_3$  is undecidable. Since it is given that  $P_2$  is undecidable, therefore, surely  $P_3$  is undecidable is correct.

**S.45 (b)**

(d) EB.2

- (a) is false since M is accepting 'abbb'  
 (c) is false since 'abba' contains 'abb' as a substring, but is being rejected by the machine.  
 (d) is false, since  $\lambda$  does not contain 'aa' as a substring, but  $\lambda$  is being rejected by M.

**S.46 (a)**

P(x) and Q(x) are predicates then statement

$$((\forall x P(x)) \vee Q(x)) \Rightarrow ((\forall x(P(x)) \vee (\forall x Q(x)))$$

\therefore

P	Q	$P \vee Q$
T	T	T
T	F	T
F	T	T
F	F	F

statement satisfy the result, so it is true

**S.47 (d)**

Let  $L = \{a^p \mid p \text{ is a prime}\}$  the statement (d) is true. There is no DFA which recognizes the L or apply the pumping lemma then we can say that  $a^p$  is not a regular language. It is not context free either, but L is surely accepted by a (linearly bounded) Turing machine.

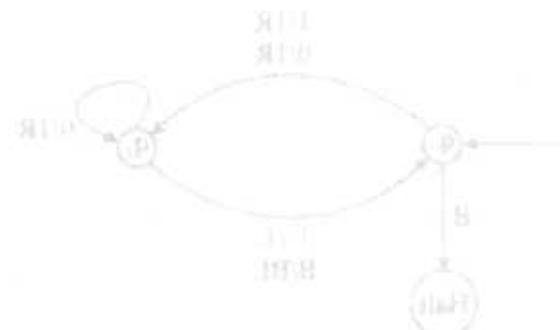
Suppose a turing machine M accept L then M must be able to accept 'M'. This leads to contradiction since Hanoi problem can never be solved since there is no less states than the number of states required for solving

(n) EB.2

17. In regular expression



so either S1 or S2 is non-deterministic state



so either S1 or S2 is non-deterministic state

000	111	0000000000
011	000	0000000000
000	111	0000000000
011	000	0000000000

Now go the own state A will come to S1 if  
a state in current M has  $\lambda^*$  as beginning

000	111	0000000000
111	000	0000000000
000	111	0000000000
111	000	0000000000

Since no language with  $\lambda^*$  as begining of  
a word M can not start in case, then the  
no final language M has  $\lambda^*$  as begining of  
 $(1+0)^*$  in general case

more difficult to prove that  $\text{P} \neq \text{NP}$ . In fact, it is not even known whether  $\text{P} = \text{NP}$  or  $\text{P} \neq \text{NP}$ . This is a major open problem in computer science.

Q.6

# ANALYSIS OF ALGORITHM AND COMPUTATIONAL COMPLEXITY

**1.5**

## LEVEL-1

- Q.1** If a function  $f(n)$  is given by  $6n^2 + n\log_2 n + 22n + 8$  then the asymptotic upper bound for  $f(n)$  is given by  
 (a)  $O(2n)$   
 (b)  $O(n)$   
 (c)  $O(n^2)$   
 (d)  $O(n \log n)$
- Q.2** Find the odd man out. The men used here are names of standard problems.  
 HAMPATH, COMPOSITES, PATH  
 (a) PATH  
 (b) HAMPATH  
 (c) COMPOSITES  
 (d) None of these
- Q.3** Which of the following formulae are satisfiable?  
 I.  $(x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge (\bar{x} \vee \bar{y})$   
 II.  $(\bar{x} \wedge y) \vee (x \vee \bar{z})$   
 (a) Only I  
 (b) Only II  
 (c) Both I & II  
 (d) Neither I nor II

- Q.4** For any function  $f(n) \geq \log(n)$   
 (a)  $\text{NSPACE}(f(n)) \subseteq \text{DSPACE}((f(n))^2)$   
 (b)  $\text{NSPACE}(f^2(n)) \subseteq \text{DSPACE}(f(n))$   
 (c)  $\text{NSPACE}(f^2(n)) \subseteq \text{DSPACE}(f^2(n))$   
 (d)  $\text{NSPACE}(f(n)) \subseteq \text{DSPACE}(f(n))$
- Q.5** The best case running time of Quick sort is  
 (a)  $O(n^2 \log n)$   
 (b)  $O(n \log n)$   
 (c)  $O(n^2)$   
 (d) All of above
- Q.6** A language  $L$  is PSPACE-hard if  
 (a) 1.  $L$  is not known to be in PSPACE and  
 2. Every  $A$  in PSPACE is polynomial time reducible to  $L$ .  
 (b) 1.  $L$  is in PSPACE and  
 2. Every  $A$  in PSPACE is not polynomial time reducible to  $L$ .  
 (c) 1.  $L$  is in PSPACE and  
 2. Every  $A$  in PSPACE is polynomial time reducible to  $L$ .  
 (d) None of these

**Q.7** Let  $t(n)$  be function, where  $t(n) \geq n$ . Then every  $t(n)$  time nondeterministic single-tape Turing machine has an equivalent \_\_\_\_\_ time deterministic single tape Turing machine.

(a)  $2^{O(t^2(n))}$

(b)  $2^{O(t(n))}$

(c)  $O(t^2(n))$

(d) None of these

**Q.8** Suppose  $f & g$  are two partial functions,  $f \neq O(g)$  means

- The ratio  $f(n)/g(n)$  is unbounded.
  - The ratio  $f(n)/g(n)$  may not be large for all large values of  $n$ , but it is large for infinitely many values of  $n$ .
  - It is impossible to find a constant  $C$  so that  $f(n) \leq C g(n)$  for all sufficiently large  $n$ .
  - The ratio is bounded.
- (a) Only I, II, III  
 (b) Only IV  
 (c) Only I & II  
 (d) All of the above

**Q.9** Consider the following algorithm:

Algorithm:

- Arrange elements in array A in the ascending order. Establish an element X to be searched.
- Let low = 1, high = n, found = false
- Repeat steps 4 and 5 till low > high or found = true
- Let mid = (low + high)/2
- $X = A[mid]$  then found = true otherwise if  $X < A[mid]$   
 then high = mid - 1  
 otherwise low = mid + 1
- If found = true then write ('present')  
 otherwise write ('not present')
- Stop

Find the time complexity of the above algorithm.

(a)  $O(\log(n))$

(b)  $O(n \log(n))$

(c)  $O(n)$

(d)  $O(r^2)$

**Q.10** Let  $f : N \rightarrow N$  be any step-counting function, then for some constant  $C$ , Time ( $f$ ) is a proper subset of \_\_\_\_\_.

(a) Time ( $Cn^2f^2$ )

(b) Time ( $Cnf^2$ )

(c) Time ( $n^2f^2$ )

(d) Time ( $Cn^2f$ )

**Q.11** If a graph is represented by an adjacency matrix, each 'for' loop in Prims algorithm must examine \_\_\_\_\_ nodes. Also, since the algorithm contains a nested for loop, its complexity is \_\_\_\_\_.

(a)  $O(\log n), O(n \log n)$

(b)  $O(n), O(n^2)$

(c)  $O(n^2), O(n^3)$

(d)  $O(n), O(n \log n)$

**Q.12** \_\_\_\_\_ can be used to make Prims algorithm more efficient and the corresponding time complexity is reduced to \_\_\_\_\_.

(a) adjacency matrix & stack,  $O(n^2)$

(b) priority queue,  $O(n)$

(c) adjacency lists,  $O(n \log n)$

(d) adjacency lists & priority queue,  $O((n + e)\log n)$

**Q.13** If L can be recognized by a TM T with a doubly infinite tape, and  $\tau_1 = f$ , then L can be recognized by an ordinary TM with time complexity

(a)  $O(h)$

(b)  $O(f)$

(c)  $\theta(f)$  or theta ( $f$ )

(d)  $\theta(h)$  or theta ( $h$ )

**Q.14** If there is an NP-complete language L whose complement is in NP, then the complement of any language in NP is in

(a) NP

(b) P

(c) neither P nor NP

(d) both NP and P

**Q.15** Both P and NP are closed under the operation of

(a) Kleen

(b) concatenation

(c) Union

(d) All the above

**Q.16** Which of the following decision problem is NP complete?

- (a) Given a finite set A, a collection C of subset of A, and an integer k, is there a subset  $A_1$  of A having k or fewer elements so that  $A_1 \cap S \neq \emptyset$  for each S in the collection C?
- (b) Given a graph G in which every vertex has even degree, and an integer k, does G have a vertex cover with k vertices? (The degree of a vertex is the number of edges containing it.) Hint: given an arbitrary graph G, find a way to modify it by adding three vertices so that all the vertices of the new graph have even degree.
- (c) Both (a) and (b) above
- (d) None of the above

**Q.17** The statement  $P = NP$  is

- (a) Still open for argument
- (b) False
- (c) True
- (d) None of these

**Q.18** L can be recognized by a multitape TM with time complexity  $O(f)$ , then L can be recognized by a one-tape machine with time complexity

- (a)  $O(h^2)$
- (b)  $O(h)$
- (c)  $O(f^2)$
- (d)  $O(f)$

## LEVEL-2

**Q.19** Which of the following statements are True?

- I.  $n = o(n \log \log n)$
- II.  $n \log \log n = o(n \log n)$
- (a) Only II
- (b) Only I
- (c) none of them
- (d) both I and II

**Q.20**  $\text{SUBSET\_SUM} = \{(S, t) \mid S = \{x_1, x_2, \dots, x_k\}$  and for some  $\{y_1, y_2, y_3, \dots, y_m\} \subseteq S$ , we have  $\sum y_i = t\}$

Consider  $S_1 = \{1, 2, 4, 6, 8, 16\}, n\}$

Which of the following values of n makes  $S_1$  not a SUBSET\_SUM?

- (a)  $n = 33$
- (b)  $n = 32$
- (c)  $n = 19$
- (d) None of these

**Q.21**  $\text{SUBSET\_SUM} = \{(S, t) \mid S = \{x_1, x_2, \dots, x_k\}$  and for some  $\{y_1, y_2, y_3, \dots, y_m\} \subseteq S$ , we have  $\sum y_i = t\}$

Which of the following belong to SUBSET\_SUM;

- I.  $\langle \{4, 11, 16, 21, 30\}, 32 \rangle$
- II.  $\langle \{1, 2, 3, 4, 5\}, 10 \rangle$
- (a) Only I
- (b) Only II
- (c) Neither I nor II
- (d) Both I and II

**Q.22** If  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = l \in \mathbb{R}^+$  then

- (a)  $f(n) \notin O(g(n))$  and  $g(n) \notin O(f(n))$
- (b)  $f(n) \notin O(g(n))$  and  $g(n) \in O(f(n))$
- (c)  $f(n) \in O(g(n))$  and  $g(n) \notin O(f(n))$
- (d)  $f(n) \in O(g(n))$  and  $g(n) \in O(f(n))$

**Q.23** Which of the following Boolean formulae is not satisfiable

- (a)  $(\bar{x} \vee \bar{y} \vee \bar{z}) \wedge (x \vee y \vee z) \wedge (x \vee y \vee \bar{z})$
- (b)  $(x \wedge \bar{x}) \vee (y \wedge \bar{y}) \vee (z \wedge \bar{z})$
- (c)  $(x \wedge \bar{y}) \vee (y \wedge \bar{x}) \vee (z \wedge \bar{x})$
- (d) All of above

- Q.24** Let  $f(n)$  be the function  $3n \log_2 n + 5n \log_2 n + 2$ , then which of the following is correct?
- $f(n) = O(n)$
  - $f(n) = O(n \log n)$
  - $f(n) = O(n \log \log n)$
  - $f(n) = O(\log n)$
- Q.25** If  $f, g, h, k : N \rightarrow N$ , such that  $f = O(h)$  and  $g = O(k)$  then
- $f + g = O(\max(h, k))$
  - $f + g = O(h) + O(k)$
  - $f + g = O(h + k)$
  - All of the above
- Q.26** The running time  $T(n)$ , where  $n$  is the input size of a recursive algorithm is given as
- $$T(n) = \begin{cases} c + T(n - 1), & \text{if } n > 1 \\ d, & \text{if } n \leq 1 \end{cases}$$
- where  $c$  and  $d$  are constants.
- The order of the algorithm is
- $2^n$
  - $n^3$
  - $n$
  - $n^2$
- Q.27** An algorithm is made up of 2 module  $M_1$  and  $M_2$ . If order  $M_1$  is  $f(n)$  and  $M_2$  is  $g(n)$ , then the order of the algorithm is \_\_\_\_\_.
- $f(n) \times g(n)$
  - $f(n) + g(n)$
  - $\min(f(n), g(n))$
  - $\max(f(n), g(n))$
- Q.28** Consider a simple connected graph  $G$  with  $n$  vertices and  $n$  edges ( $n > 2$ ). Then which of the following statements are true.
- $G$  has at least one cycle
  - $G$  has no cycles
  - The graph obtained by removing any edge from  $G$  is not connected
  - None of them
- Q.29** What is true about  $O(n \log(n))$ ,  $O(n^3)$ ,  $O(2^n)$ ,  $O(1)$ ? The notation  $O(f) = O(g)$  means that the set of functions  $O(f)$  is contained in the set of functions  $O(g)$ .
- $O(2^n) = O(n \log(n)) = O(n^3) = O(1)$
  - $O(1) = O(n \log(n))$  and  $O(2^n) = O(n^3)$
  - $O(n \log(n)) = O(n^3) = O(2^n) = O(1)$
  - $O(1) = O(n \log(n)) = O(n^3) = O(2^n)$
- Q.30** Consider the following program segment
- ```
void main ()
{
    int i, j = 0;
    for (i = 0; i < 100; i++)
    {
        print("%u", i^2);
        print("\n%u", 2);
    }
}
```
- The runtime complexity of the above code is \_\_\_\_\_.
- $O(n^2)$
  - $O(2^n)$
  - $O(1)$
  - $O(n \log(n))$
- Q.31** Let  $SHAM_3$  be the problem of finding a Hamiltonian cycle in a graph  $G = (V, E)$  with  $|V|$  divisible by 3 and  $DHAM_3$  be the problem of determining if a Hamiltonian cycle exists in such graphs. Which one of the following is true?
- Both  $DHAM_3$  and  $SHAM_3$  are NP-hard
  - $SHAM_3$  is NP-hard, but  $DHAM_3$  is not
  - $DHAM_3$  is NP-hard, but  $SHAM_3$  is not
  - Neither  $DHAM_3$  nor  $SHAM_3$  is NP-hard
- Q.32** Which of the following is true?
- $O(\log_4 n)$  is equivalent to  $O(\log_2 n)$
  - $O(\log_4 n)$  dominates  $O(\log_2 n)$
  - $O(\log_2 n)$  dominates  $O(\log_4 n)$
  - All of these

**Q.33** The time complexity of the following algorithm is \_\_\_\_\_.

Steps:

1. Establish an array A of 'n' non-negative integers
2. Let m = the max. no. of digits of an array element
3. Initialize the queues  $q_0, q_1, q_2, \dots, q_9$
4. Place elements of A into the queues  $q_0, q_1, q_2, \dots, q_9$
5. Create the array A using the elements  $q_0, q_1, q_2, \dots, q_9$
6. Repeat step 7 through 9, ' $m-1$ ' times
7. Initialize the queue  $q_0, q_1, \dots, q_9$
8. Place the elements of A into the queues according to the next significant digit
9. Create A using queue  $q_0, q_1, \dots, q_9$
10. Stop
  - (a)  $O(m \log(n))$
  - (b)  $O(n * m)$
  - (c)  $O(n/m)$
  - (d)  $O(n)$

**Q.34** If  $n = 4096$ , the run time of a certain algorithm is 512 ms. If  $n$  is increased by 12288, the run time of that algorithm is \_\_\_\_\_ if its time complexity is  $O(n)$ .

- (a) Insufficient data
- (b) 1536 ms
- (c) 2048 ms
- (d) 2080 ms

**Q.35** Time complexity of a turing machine accepting all strings of the language L which starts in a and ends in b, all b's follow all a's, and no. (a) = no.

- (b) is \_\_\_\_\_
- (a)  $O(n \log n)$
- (b)  $O(\log n)$
- (c)  $O(n^2)$
- (d)  $O(n)$

**Q.36** For the string aaaabbbb, determine its time complexity

- (a) 4
- (b) 16
- (c) 2.4082
- (d) 0.6021

## LEVEL-3

**Q.37** A clique is an undirected graph in a subgraph, wherein every two nodes are connected by an edge. A k-clique contains k nodes. The number of edges in a k-clique will be

- (a)  $\frac{k(k+1)}{2}$
- (b)  $\frac{k^2 - k}{2}$
- (c)  $k(k+1)$
- (d)  $k!$

**Q.38** Each of the functions  $n!$ ,  $n^n$  and  $2^n$  has growth rate \_\_\_\_\_ than that of exponential function.

- (a) cannot be determined
- (b) equal to
- (c) greater
- (d) less

**Q.39**  $O\left(\frac{n^2}{(\log_3 n)(\sqrt{n \log_e n})}\right)$  is equivalent to

- (a)  $O((n/\log n)^{1.5})$
- (b)  $O(n^2/(\log n)^{1.5})$
- (c)  $O(n/\log n)$
- (d) can't determine

**Q.40** An algorithm takes 0.5 ms for input size of 100. How long will it take for the input size of 500 if the running time is Linear,  $O(N \log N)$ , and Quadratic?

- (a) 2.5 ms, 3.37 ms, 12.5 ms
- (b) 3.37 ms, 12.5 ms, 3.02 ms
- (c) 2.5 ms, 3.37 ms, 62.5 ms
- (d) 2.8 ms, 3.8 ms, 63 ms

## GATE QUESTIONS

**Q.41** Which of the following problems is not NP-hard? [GATE 1992]

- (a) Hamiltonian circuit problem
- (b) The 0/1 Knapsack problem
- (c) Finding biconnected components of a graph
- (d) The graph coloring problem

**Q.42**  $\sum_{1 \leq k \leq n} O(n)$ , where  $O(n)$  stands for order  $n$  is :

[GATE 1993]

- (a)  $O(n)$
- (b)  $O(n^2)$
- (c)  $O(n^3)$
- (d)  $O(3n^2)$

**Q.43** Consider the following functions

$$f(n) = 3n^{\sqrt{n}}$$

$$g(n) = 2^{\sqrt{n}} \log_2 n$$

$$h(n) = n!$$

Which of the following is true? [GATE 2000]

[2-Marks]

- (a)  $h(n)$  is  $O(f(n))$
- (b)  $h(n)$  is  $O(g(n))$
- (c)  $g(n)$  is not  $O(f(n))$
- (d)  $f(n)$  is  $O(g(n))$

**Q.44** Let  $f(n) = n^2 \log n$  and  $g(n) = n(\log n)^{10}$  be two positive functions of  $n$ . Which of the following statements is correct? [GATE 2001]

[1-Mark]

- (a)  $f(n) = O(g(n))$  and  $g(n) \neq O(f(n))$
- (b)  $g(n) = O(f(n))$  and  $f(n) \neq O(g(n))$
- (c)  $f(n) \neq O(g(n))$  and  $g(n) \neq O(f(n))$
- (d)  $f(n) = O(g(n))$  and  $g(n) = O(f(n))$

**Q.45** Ram and Shyam have been asked to show that a certain problem  $\Pi$  is NP-complete. Ram shows a polynomial time reduction from the 3-SAT problem to  $\Pi$ , and Shyam shows a polynomial time reduction from  $\Pi$  to 3-SAT. Which of the following can be inferred from these reduction? [GATE 2003]

[1-Mark]

(a)  $\Pi$  is neither NP-hard nor in NP

(b)  $\Pi$  is NP-complete

(c)  $\Pi$  is in NP, but is not NP-complete

(d)  $\Pi$  is NP-hard but not NP-complete

**Q.46** Nobody knows yet if  $P = NP$ . Consider the language  $L$  defined as follows

$$L = \begin{cases} (0+1)^* & \text{if } P = NP \\ \emptyset & \text{otherwise} \end{cases}$$

Which of the following statements is true?

[GATE 2003]

[1-Mark]

(a) Whether  $L$  is recursive or not will be known after we find out if  $P = NP$

(b)  $L$  is not recursively enumerable

(c)  $L$  is recursively enumerable but not recursive

(d)  $L$  is recursive

**Q.47** Consider two languages  $L_1$  and  $L_2$  each on the alphabet  $\Sigma$ . Let  $f : \Sigma \rightarrow \Sigma$  be a polynomial time computable bijection such that  $(\forall x) [x \in L_1 \text{ iff } f(x) \in L_2]$ . Further, let  $f^{-1}$  be also polynomial time computable. Which of the following CANNOT be true? [GATE 2003]

[2-Marks]

(a)  $L_1 \in P$  and  $L_2$  is finite

(b)  $L_1 \in NP$  and  $L_2 \in P$

(c)  $L_1$  is undecidable and  $L_2$  is decidable

(d)  $L_1$  is recursively enumerable and  $L_2$  is recursive

**Q.48** The problems 3-SAT and 2-SAT are [GATE 2004] [1-Mark]

- (a) both in P
  - (b) both NP-complete
  - (c) NP-complete and in P respectively
  - (d) undecidable and NP-complete respectively

**Q.49** Consider the following two problems on undirected graphs

- $\alpha$  : Given  $G(V, E)$ , does  $G$  have an independent set of size  $|V| - 4$ ?  
 $\beta$  : Given  $G(V, E)$ , does  $G$  have an independent set of size 5?

Which one of the following is TRUE?

[GATE 2005]

- [2-M]
- (a)  $\alpha$  is in the P and  $\beta$  is NP-complete
  - (b)  $\alpha$  is NP-complete and  $\beta$  is in P
  - (c) Both  $\alpha$  and  $\beta$  are NP-complete
  - (d) Both  $\alpha$  and  $\beta$  are in P

**Q.50** In a schema with attributes A, B, C, D and E following set of functional dependencies are given.

A → B

A → C

CD → E

B → D

E → A

Which of the following functional dependencies is NOT implied by the above set?

[IT-GATE 2005]

- (a) CD → AC [2-Marks]  
 (b) BD → CD  
 (c) BC → CD  
 (d) AC → BC

**Q.51** Let S be an NP-complete problem Q and R be two other problems not known to be in NP. Q is polynomial-time reducible to S and S is polynomial-time reducible to R. Which one of the following statements is true? [GATE 2006]

- (a) R is NP-complete [1-Mark]  
(b) R is NP-hard  
(c) Q is NP-complete  
(d) Q is NP-hard

## ANSWER KEY

# SOLUTIONS

## S.1 (c)

In big-O notation, we consider the highest order term in the given expression. Here, the highest order is 2 for  $n^2$ .

## S.2 (a)

HAMPATH and COMPOSITES belong to class NP, whereas PATH belong to class P.

## S.3 (b)

A Boolean formula is satisfiable if some assignment of 0's and 1's to the variables makes the formula evaluate to 1.

Here, formula I is not satisfiable for any combination of x and y.

**formula II is satisfiable as for  $x = 0, y = 1$  and  $z = 0$ , it evaluates to 1.**

## S.4 (a)

By Walter Savitch's theorem which states that for any function  $f(n) \geq \log(n)$ .

$$\text{NSPACE}(f(n)) \subseteq \text{DSPACE}((f(n))^2)$$

## S.6 (a)

A language L is NP-hard (likewise, PSPACE-hard) if every language A in NP (likewise, PSPACE) is polynomial-time reducible to L, even if L is not in NP (likewise, PSPACE) itself.

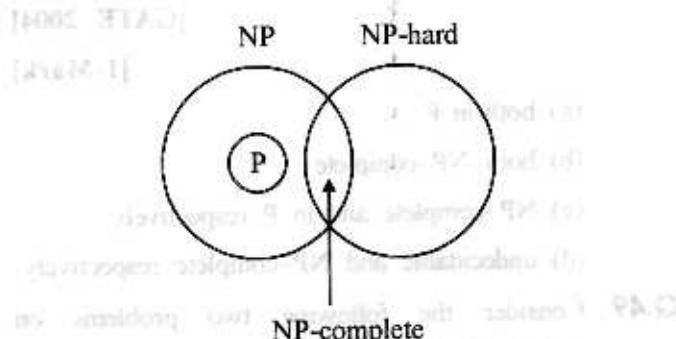
## S.7 (b)

Simulating a non-deterministic turing machine with a deterministic turing machine can be viewed as searching a tree of non-deterministic computations for accepting states. Since the non-deterministic TM is a  $O(f(n))$  time machine the path from the root to a leaf node is bounded by  $O(f(n))$  steps. Searching the tree for an accepting state is an exponential operation bounded by  $2^{O(f(n))}$  steps.

## S.9 (a)

The given algorithm is for binary search. The worst case time complexity of this is  $O(\log(n))$ .

## S.14 (a)



## S.17 (a)

The question of whether  $P = NP$  is one of the greatest unsolved problems in theoretical computer science and contemporary mathematics. Most researches believe that the two classes are not equal because people have invested enormous effort to find polynomial time algorithms for certain problems in NP without success. But the question is still unsolved and open for argument.

## S.19 (d)

From definition of small-o notation

Simple Method :

In small-o notation the order of expression in o-notation must be more than of the function. Here, order of  $n \log \log n$  is more than  $n$ , and then statement I is true. Similarly, order of  $n \log n$  is more than that of  $n \log \log n$ . Hence, statement II is also true.

Note:  $f(n)$  is never  $o(f(n))$

## S.20 (d)

All values of  $n$  makes S<sub>1</sub> a SUBSET\_SUM

For  $n = 33$ , we get  $16 + 8 + 6 + 2 + 1 = 33$

For  $n = 32$ , we get  $16 + 8 + 6 + 2 = 32$

For  $n = 19$ , we get  $8 + 6 + 4 + 1 = 19$

## S.21 (d)

Consider for (I),  $11 + 21 = 32$  where  $\{11, 21\} \subseteq S$

Hence (I)  $\in$  SUBSET\_SUM

Consider for (II),  $5 + 4 + 1 = 10$ ,

where  $\{5, 4, 1\} \subseteq S$

Hence (II)  $\in$  SUBSET\_SUM

**S.22 (d)**

Proof:

Assume  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \ell \in \mathbb{R}^+$ . Let  $\delta = \ell$  and

$$L = 2\ell$$

By definition of limit, the difference in absolute value between  $f(n)/g(n)$  and  $\ell$  is not more than  $\delta$  for all sufficiently large  $n$ . But this means that  $f(n)/g(n) \leq \ell + \delta = C$ . It thus exhibits a positive real constant  $C$  such that  $f(n) \leq C g(n)$  for all sufficiently large  $n$ , which proves that  $f(n) \in O(g(n))$ . The fact that  $g(n) \in O(f(n))$  is

automatic since  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \ell \in \mathbb{R}^+$  implies that

$\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = \frac{1}{\ell} \in \mathbb{R}^+$  and thus the above

reasoning applies with  $f(n)$  with  $g(n)$  interchanged.

**S.23 (b)**

A Boolean formula is satisfiable if some assignment of 0's and 1's to the variable makes the formula evaluate to 1.

Here, (a) evaluates to 1 for  $y = 0, x = 1, z = 0/1$

(c) evaluates to 1 for  $x = 0, y = 1, z = 1$

But (b) can not be evaluated to 1 for any combination of  $x, y & z$ .

**S.24 (b)**

In big-O notation, we consider only the highest order term. We may think that

$f(n) = O(n \log \log n)$  but  $\log n$  dominates  $\log \log n$ . That is, order of  $\log n$  is more than order of  $\log \log n$ . Hence  $f(n) = O(n \log n)$ .

**S.25 (d)**

Property of O-notation

$$O(h + k) = O(h) + O(k) = O(\max(h, k))$$

**S.26 (c)**

Recursively applying the relation, we finally get

$$T(n) = c + T(n-1)$$

$$= c + c + T(n-2)$$

$$= c + c + c + c \dots n \text{ times} + d$$

$$= n * c + d$$

$$= O(n) \dots c \text{ and } d \text{ are constants}$$

**S.27 (d)**

By definition of order, there exists constants  $C_1, C_2, n_1, n_2$  such that

$$T(n) \leq C_1 \times f(n), \text{ for all } n \leq n_1$$

$$T(n) \leq C_2 \times g(n), \text{ for all } n \leq n_2$$

$$\text{Let } N = \max(n_1, n_2) \text{ and } C = \max(C_1, C_2)$$

$$\text{So, } T(n) \leq C \times f(n) \text{ for all } n \leq N$$

$$T(n) \leq C \times g(n) \text{ for all } n \leq N$$

Adding,

$$T(n) \leq C/2 \times (f(n) + g(n))$$

without loss of generality, let

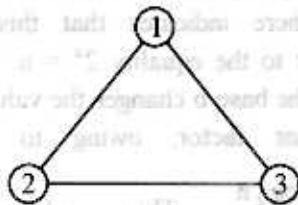
$$\max(f(n), g(n)) = f(n)$$

$$\therefore T(n) \leq C/2(f(n) + f(n)) \leq C \times f(n)$$

So, order is  $f(n)$ , which is  $\max(f(n), g(n))$ , by our assumption.

**S.28 (a)**

Let us take  $n = 3$ . Therefore the graph with 3 nodes and 3 edges is a triangular graph as shown below.



We can see that option (a) is satisfied.

Also, (b) is eliminated as this has a cycle.

Therefore, (b) is not true for all cases.

Similarly, (c) is eliminated, since if we remove any of the edge from the above figure, it will still be connected.

**S.29 (d)**

Since Big Oh defines a set of functions, the notation  $O(f) = O(g)$  means that the set of functions  $O(f)$  is contained in the set of functions  $O(g)$ .

The " $=$ " sign does not mean equality in the usual algebraic sense.

$$\therefore \text{if } O(f) = O(g)$$

$$\text{then } O(g) \neq O(f)$$

i.e. if  $f$  is contained in  $g$ , then  $g$  cannot be a subset of  $f$ .

$$\therefore O(1) < O(n \log n) < O(n^3) < O(2^n)$$

**S.30 (c)**

The loop runs from  $i = 1$  to 100. The runtime complexity of this program is  $O(n) = 100$ . Since 100 is a constant, we get complexity =  $O(1)$ .

**S.31 (a)**

$\text{SHAM}_3$  be the problem of finding Hamiltonian cycle in a graph  $G = (V, E)$  with  $|V|$  is divisible by 3 contains three remainders either 0, 1 and 2. We can reduce this problem from 3-SAT and hence  $\text{SHAM}_3$  is NP-complete problem.  $\text{DHAM}_3$  be the problem of determining if a Hamiltonian cycle exists is also a reduction from 3-SAT and hence it is also NP-complete problem. So both  $\text{DHAM}_3$  and  $\text{SHAM}_3$  are NP complete and hence NP Hard also.

**S.32 (a)**

The big-O interacts with logarithms in a particular way. Usually when we use logarithms we must specify the base, as in  $x = \log_2 n$ . The base 2 here indicates that this equality is equivalent to the equality  $2^x = n$ . Changing the value of the base  $b$  changes the value of  $\log_b n$  by a constant factor, owing to the identity

$$\log_b n = \frac{\log_2 n}{\log_2 b} \quad \text{Thus, when we write}$$

$f(n) = O(\log n)$ , specifying the base is no longer necessary because we are suppressing constant factors anyway.

Hence,  $O(\log_2 n)$  is equivalent to  $O(\log_4 n)$ .

**S.33 (b)**

The algorithm is that of Radix sort

$n$  = no. of elements to be sorted

$m$  = max. no. of digits

$\therefore$  time complexity is  $O(m * n)$

$\Rightarrow$  for 10 elements, say 12, 80, 100, 1, 16, 150, 48, 121, 66, 30

time complexity is  $O(m * n) = 3 * 10 = 30$

**S.34 (c)**

Since, the time complexity is given as  $O(n)$ , the computational time will linearly increase with  $n$ .

$$\frac{4096}{512} = \frac{12288}{x}$$

$$\therefore x = \frac{12288 \times 512}{4096} = 2048 \text{ ms}$$

**S.35 (c)**

$L = \{a^n b^n : n \geq 1\}$  is the language for  $w = a^n b^n$ , it takes roughly  $n$  steps to match each  $a$  with the corresponding  $b$ . Therefore the whole computation takes  $O(n^2)$  time.

**S.36 (b)**

$$\text{aaaabb} = a^4 b^4$$

$$\therefore O(n^2) = (4)^2 = 16$$

**S.37 (b)**

Since, in a clique, every node is connected to every other node, the total number of edges is given by

$$(k - 1) + (k - 2) + \dots + 2 + 1$$

$$\frac{(k-1)k}{2} = \frac{k^2 - k}{2}$$

Eg. for  $k = 3$



$$\text{No. of edges} = \frac{3^2 - 3}{2} = \frac{6}{2} = 3$$

Similarly for  $k = 4$ , number of edges = 6.

**S.38 (c)**

Growth rate does not consider the growth in absolute terms but it depends on the current value and the growth. It is a relative term. For this, consider an exponential function  $2^n$ .

$$1. \ 2^n \text{ Growth rate} = \frac{2^3 - 2^2}{2^2} \times 100 = 100\%$$

$$2. \ n! \text{ Growth rate} = \frac{3! - 2!}{2^2} \times 100 = 200\%$$

$$3. \ n^n \text{ Growth rate} = \frac{3^2 - 2^2}{2^2} \times 100 = 575\%$$

$$4. \ 2^{2^n} \text{ Growth rate} = \frac{2^8 - 2^4}{2^4} \times 100 = 1500\%$$

Hence we can see that these functions have growth rate greater than the exponential function  $2^n$ .

### S.39 (a)

It is usually unnecessary to specify the base of algorithm inside an asymptotic notation. This is because  $\log_a n = \log_b n \times \log_a b$  for all positive reals  $a, b$  and  $n$  such that neither  $a$  nor  $b$  is equal to 1. The point is that  $\log_a b$  is a positive constant when  $a$  and  $b$  are constant larger than 1. Therefore,  $\log_a n$  and  $\log_b n$  differ only by a constant multiplicative factor.

Now consider

$$\begin{aligned} & O\left(n^2 \sqrt{(\log_3 n)(\sqrt{n \log_e n})}\right) \\ &= O\left(n^2 \sqrt{(\log n)(\sqrt{n \log n})}\right) \quad \text{(a)} \quad \text{Ignoring base} \\ & \quad \because e = 2.73 \approx 3 \\ &= O\left(n^2 \sqrt{\log n \cdot n^{1/2} (\log n)^{1/2}}\right) \\ &= O\left(n^{3/2} \sqrt{(\log n)^{3/2}}\right) \\ &= O\left((n/\log n)^{1.5}\right) \end{aligned}$$

### S.40 (a)

#### I. Linear

$$\text{Time taken} = 500 * 0.5 / 100 = 2.5 \text{ ms}$$

#### II. $O(N \log N)$

$$\text{Time taken} = \frac{(0.5 * 500 * \log 500)}{100 * \log 100}$$

$$= 3.37 \text{ ms}$$

(b) 3.37

### III. Quadratic

$$\text{Time taken} = \frac{(0.5 * 500^2)}{100^2} = 12.5 \text{ ms}$$

### S.41 (b)

The 0/1 knapsack problem is not NP-hard.

### S.42 (a)

$$\sum_{1 \leq k \leq n} O(n), \text{ where } O(n) \text{ stands for order } n \text{ is}$$

$$O(n).$$

$$\Rightarrow O(1) + O(k) \dots O(n)$$

$$\Rightarrow O(n)$$

### S.43 (c)

$$f(n) = 3n^{\sqrt{n}}$$

$$O(f(n)) = O(n^{\sqrt{n}})$$

$$h(n) = n!$$

$$O(h(n)) = O(n^n)$$

$$g(n) = 2^{\sqrt{n}} \log_2 n$$

$$= 2^{\log_2 n^{\sqrt{n}}}$$

$$= n^{\sqrt{n}}$$

$$O(g(n)) = O(n^{\sqrt{n}})$$

### S.44 (c)

$$f(n) = n^2 \log n$$

$$g(n) = n (\log n)^{10}$$

$$f(n) \neq O(g(n)) \text{ and } g(n) \neq O(f(n)).$$

### S.45 (b)

Ram shows  $3\text{-SAT} \leq_p \Pi$

Shyam shows  $\Pi \leq_p 3\text{-SAT}$

Now if  $p_1 \leq_p p_2$  and if  $p_1$  is NP-complete, we can say that  $p_2$  is also NP-complete.

Now Ram has shown  $3\text{-SAT} \leq_p \Pi$  and we know that  $3\text{-SAT}$  is an NP-complete problem. Therefore, we can conclude that  $\Pi$  is NP-complete.

**S.46 (d)**

Either  $P = NP$  or  $P \neq NP$ .

$\therefore$  Either  $L = (0 + 1)^*$  or  $L = \emptyset$ .

In case  $L = (0 + 1)^*$ , a dumb turing machine which gives "Yes" answer to every  $w \in L$  question, will correctly answer the problem. In case  $L = \emptyset$ , then also a dumb turing machine which gives "No" answer to all the  $w \in L$  question will correctly solve the problem. Thus we can say that a turing machine exists which solve this membership problem correctly for every  $w \in \Sigma^*$ . Therefore,  $L$  is recursive.

**S.47 (c)**

Since  $f$  is polynomial time computable.

$$L_1 \leq_P L_2 \quad \dots \text{(i)}$$

Since  $f^{-1}$  is also polynomial time computable

$$L_2 \leq_P L_1 \quad \dots \text{(ii)}$$

Now consider (a)  $L_1 \in P$  and  $L_2$  is finite.

From (ii) we can say that  $L_1 \in P \Rightarrow L_2 \in P$ .

Now  $L_2 \in P$  and  $L_2$  is finite is not contradictory.

$\therefore$  (a) can be true.

Choice (b)  $L_1 \in NP$  and  $L_2 \in P$

Now  $L_2 \leq_P L_1$

and  $L_1 \in NP \Rightarrow L_2 \in NP$  which does not contradict  $L_2 \in P$ .

$$L_1 \leq_P L_2$$

and  $L_2 \in P \Rightarrow L_1 \in P$  which does not contradict  $L_1 \in NP$ .

$\therefore$  (b) can be true.

Consider (c)  $L_1$  is undecidable and  $L_2$  is decidable.

Now  $L_1 \leq_P L_2$

and  $L_2$  is decidable  $\Rightarrow L_1$  is decidable.

which contradicts  $L_1$  is undecidable.

$\therefore$  (c) cannot be true.

Consider (d)  $L_1$  is RE and  $L_2$  is REC

$L_2 \leq_P L_1$  and  $L_1$  is RE  $\Rightarrow L_2$  is RE

Which does not contradict  $L_2$  is REC.

$L_1 \leq_P L_2$  and  $L_2$  is REC  $\Rightarrow L_1$  is REC

Which does not contradict that  $L_1$  is RE.

$\therefore$  (d) can be true.

**S.48 (c)**

3-SAT problem is NP-complete problem but 2-SAT problem is solvable in polynomial time so it is P type problem.

**S.49 (c)**

Consider the statement

$\alpha$  : Given  $G(V, E)$ , does  $G$  have an independent set of size  $|V| - 4$  :

Let  $G$  contains  $n$  node and assume  $|n| - 4 = K$  for some constant  $K$  so there is a polynomial time reduction from 3-SAT then  $\alpha$  is also NP-complete.

$\beta$  : Given  $G(V, E)$ , does  $G$  have an independent set of size 5 is also NP-complete, reduction from 3-SAT.

**S.50 (b)**

$$A \rightarrow B \quad CD \rightarrow A$$

$$\therefore CD \rightarrow E, E \rightarrow A, A \rightarrow C$$

$$A \rightarrow C \quad CD \rightarrow C$$

$$CD \rightarrow E \quad BC \rightarrow CD$$

$$B \rightarrow D \quad B \rightarrow D$$

$$E \rightarrow A \quad CD \rightarrow E, E \rightarrow A, A \rightarrow C$$

$$AC \rightarrow BC, A \rightarrow B, A \rightarrow C$$

**S.51 (a)**

Note : If  $P_1 \leq_P P_2$  then  $P_1$  is NP-complete  $\Rightarrow P_2$  is NP-complete.

Given,  $S$  is NP-complete.

$$Q \leq_P S \text{ and } S \leq_P R$$

$$S \leq_P R$$

$S$  is NP-complete  $\Rightarrow R$  is NP-complete

# **UNIT-2**

# **DIGITAL LOGIC**

# DIGITAL LOGIC

Unit-2

# 2.1

## NUMBER SYSTEM

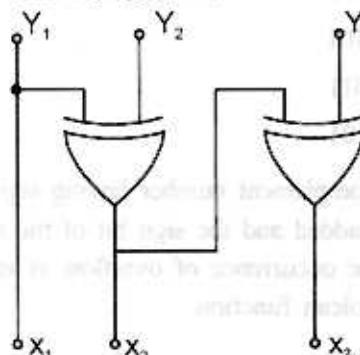
### LEVEL-1

- Q.1** Determine the decimal values of the binary numbers in 2's complement Number is: 10101010  
(a) -86  
(b) +86  
(c) -98  
(d) +98
- Q.2** The decimal number -39 is expressed in 2's complement form as  
(a) 11010001  
(b) 11011000  
(c) 00100111  
(d) 11011001
- Q.3** The 2's complement representation of  $(-539)_{10}$  in hexadecimal is  
(a) ABE  
(b) DE5  
(c) DBC  
(d) 9E7
- Q.4** The fraction 0.6810 is equal to  
(a) 0.010101<sub>2</sub>  
(b) 0.10101<sub>2</sub>  
(c) 0.101<sub>2</sub>  
(d) 0.10111<sub>2</sub>

- Q.5** The quantity 837 in excess-3 BCD code would be represented as  
(a) 1001 0011 0111  
(b) 1011 0110 1010  
(c) 1000 0001 1001  
(d) 1000 0011 0111

- Q.6** Booth's coding in 8 bits for the decimal number 57 is  
(a) 0 - 100 + 100 - 1  
(b) 0 - 100 + 1000  
(c) 0 - 1 + 100 - 10 + 1  
(d) 0 0 - 10 + 100 - 1

- Q.7** The logic circuit given below converts a binary code  $Y_1 Y_2 Y_3$  into



- (a) Gray code  
(b) Excess-3 code  
(c) BCD code  
(d) Hamming code

**Q.8** If  $(211)_x = (152)_8$ , then the value of base x is

- (a) 6
- (b) 5
- (c) 9
- (d) 7

**Q.9** A 7 bit Hamming code groups consisting of 4 information bits and 3 parity bits is transmitted. The group 1101100 is received in which at most a single error has occurred. The transmitted code is

- (a) 1001100
- (b) 1100100
- (c) 1111100
- (d) 1101000

**Q.10**  $(1101101.1011)_2 = (?)_{10}$

- (a) 109.7865
- (b) 109.6875
- (c) 109.6880
- (d) 119.08

**Q.11** The equivalent in decimal for the excess -3 code 843 is

- (a) 840
- (b) 510
- (c) 846
- (d) None of these

**Q.12** Subtraction of the given signed number is

$$00001100 - 11110111$$

- (a) 11101010
- (b) 00010101
- (c) 00001011
- (d) 00110101

**Q.13** Two 2's complement number having sign bits x and y are added and the sign bit of the result is z. then, the occurrence of overflow is indicated by the Boolean function

- (a) xyz
- (b)  $\bar{x}\bar{y}z$
- (c)  $\bar{x}\bar{y}z + xy\bar{z}$
- (d)  $xy + yz + zx$

**Q.14** 110111, 10111 and 1110111 correspond to the 2's complement representation of the following set of numbers

- (a) -7, 7 and -7 respectively
- (b) -9, -9 and -9 respectively
- (c) 29, 8, and 56 respectively
- (d) -29, -8 and -57 respectively

**Q.15** 0 0 1 1 0 1 0 1 1 0 1 0 0 1 0 0

The above excess-3 code equivalent to decimal:

- (a) 2391
- (b) 0271
- (c) 5642
- (d) 0358

**Q.16**  $(1\ 1\ 0\ 1\ 0\ 1)_2$  is numerically equivalent to:

- (i)  $(65)_8$
- (ii)  $(53)_{10}$
- (iii)  $(46)_{16}$

The correct answer is:

- (a) (i), (ii)
- (b) (ii), (iii)
- (c) (i), (iii)
- (d) none of these

### Linked Questions 17 & 18:

**Q.17** The 10's complement of a number is  $(47480)_{10}$ . Find the number in decimal equivalent.

- (a) 52520
- (b) 34380
- (c) 63620
- (d) 37480

**Q.18** The octal equivalent of the answer of Q.17 is

- (a)  $(146450)_8$
- (b)  $(63620)_8$
- (c)  $(52520)_8$
- (d) None of these

## LEVEL-2

**Q.19** Determine the value of x, if  $(211)_x = (152)_8$

- (a) -7.5
- (b) -6.5
- (c) -5.5
- (d) 0

**Q.20** Multiplication of  $296_{12}$  and  $57_{12}$

- (a) 11706
- (b) 13706
- (c) 12706
- (d) none.

**Q.21** Value of  $296_{14} * 57_{14}$

- (a) 40348
- (b) 109C0
- (c) A9C0
- (d) 109D0

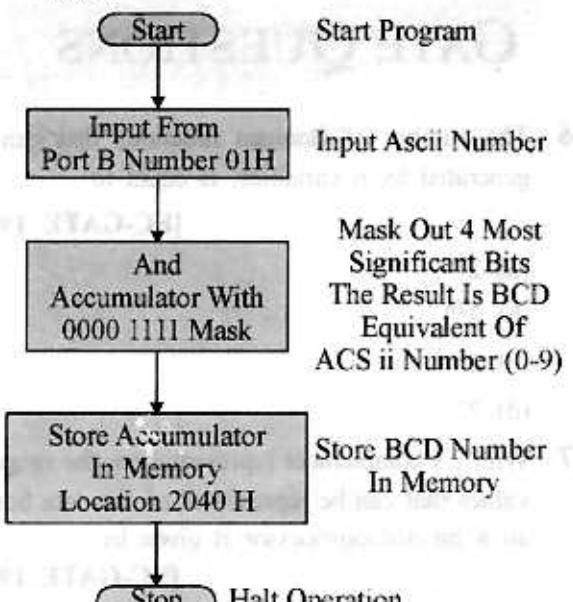
**Q.22** Convert decimal number 250.5 to base 7

- (a)  $(303.3)_7$
- (b)  $(404.3)_7$
- (c)  $(505.3)_7$
- (d) None.

**Q.23** Convert decimal number 250.0 to base 12

- (a)  $16A_{12}$
- (b)  $18A_{12}$
- (c)  $14A_{12}$
- (d) None.

**Q.24** The flowchart represents a program segment that inputs an ASCII coded number and masks out the most significant 4-bits. Determine the result:



- (a) Ternary
- (b) Binary
- (c) BCD
- (d) None.

**Q.25** The octal representation of an integer is  $(342)_8$

If this were to be treated as an eight bit integer in an 8085 based computer, its decimal equivalent is

- (a) -98
- (b) 76
- (c) -30
- (d) 226

**Q.26** Which of the following binary numbers are not divisible by 4?

- (a) 10101010101010
- (b) 111000111000
- (c) 1111000011
- (d) all of these

**Q.27** The  $100110_2$  is numerically not equivalent to

- (a)  $26_{16}$
- (b)  $212_4$
- (c)  $36_{10}$
- (d)  $46_8$

**Q.28** A signed integer has been stored in a byte using 2's complement format. We wish to store the same integer in 16-bit word. We should copy the original byte to the less significant byte of the word and fill the most significant byte with

- (a) 0
- (b) 1
- (c) complement of the MSB of the original byte
- (d) equal to the MSB of the original byte

**Q.29** The following point binary number is given as

| S | E        | F                        |
|---|----------|--------------------------|
| 1 | $\oplus$ | 100011100010000000000000 |

- (a) 1100011000110000000
- (b) -11000110001000000
- (c) -1100011000110000000
- (d) 110001100010000000

**Q.30** Consider the signed binary number  $A = 01010110$  and  $B = 11101100$  where  $B$  is the 1's complement and MSB is the sign bit. In list-I operation is given, and in list-II resultant binary number is given.

| List - I |          | List - II |         |         |
|----------|----------|-----------|---------|---------|
| P.       | $A + B$  | 1.        | 0 1 0 0 | 0 0 1 1 |
| Q.       | $B - A$  | 2.        | 0 1 1 0 | 1 0 0 1 |
| R.       | $A - B$  | 3.        | 0 1 0 0 | 0 0 1 0 |
| S.       | $-A - B$ | 4.        | 1 0 0 1 | 0 1 0 1 |
|          |          | 5.        | 1 0 1 1 | 1 1 0 0 |
|          |          | 6.        | 1 0 0 1 | 0 1 1 0 |
|          |          | 7.        | 1 0 1 1 | 1 1 0 1 |
|          |          | 8.        | 0 1 1 0 | 1 0 1 0 |

The correct match is

- |     | P | Q | R | S |
|-----|---|---|---|---|
| (a) | 3 | 4 | 2 | 5 |
| (b) | 3 | 6 | 8 | 7 |
| (c) | 1 | 6 | 2 | 5 |
| (d) | 1 | 4 | 8 | 7 |

**Q.31** Convert  $(14.34)_{10}$  into binary.

- (a) 1 0 1 1 . 1 1 0 1
- (b) 1 1 1 0 . 1 0 0 1
- (c) 1 1 1 0 . 0 1 0 1 0
- (d) 1 0 1 1 . 0 1 0 0 1

## LEVEL-3

**Q.32** Design a circuit using JK flip-flop and TTL gates, if required, to introduce three wait states in every Input-Output read machine cycle.

- (a) Cleared when  $S_1 = S_2 = \overline{I_0/M} = 0$
- (b) Cleared when  $S_1 = S_2 = \overline{I_0/M} = 1$
- (c) Both
- (d) None

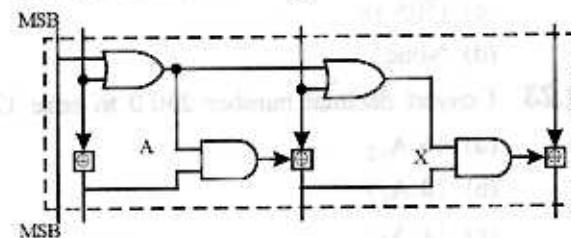
**Q.33** Arrange decade counter to obtain voltage of 1 kHz from line frequency of 50 Hz

- (a) 4 Hz
- (b) 2 Hz
- (c) 3 Hz
- (d) 1 Hz

**Q.34** Arrange 74293 to operate as Mod-10 counter with 10 KHz clock input

- (a)  $Q_1, Q_3$  are connected to MR 1
- (b)  $Q_1, Q_3$  are connected to MR 2
- (c)  $Q_1, Q_3$  are connected to MR 1 and MR 2
- (d) None

**Q.35** The circuit shown in fig converts



- (a) BCD to binary code
- (b) Binary to excess
- (c) Gray to Binary Code
- (d) Excess-3 to Gray Code

## GATE QUESTIONS

**Q.36** The number of Boolean functions that can be generated by  $n$  variables, is equal to

[IEC-GATE 1990]

- (a)  $2^n$
- (b)  $2^{2n}$
- (c)  $2^{n-1}$
- (d)  $2^n$

**Q.37** With 2's complement representation, the range of values that can be represented on the data bus of an 8-bit microprocessor is given by

[IEC-GATE 1990]

- (a) -128 to + 127
- (b) -127 to + 128
- (c) -127 to + 128
- (d) -256 to + 256

**Q.38** The minimal function that can detect a "divisible by 3" 8421 BCD code digit (representation is  $D_8D_4D_2D_1$ ) is given by [EC-GATE 1990]

- (a)  $D_8D_1 + D_4D_2 + D_8D_2D_1$
- (b)  $D_8D_1 + \bar{D}_4D_2D_1 + \bar{D}_1D_2D_4 + \bar{D}_8\bar{D}_4\bar{D}_2\bar{D}_1$
- (c)  $D_4D_1 + D_4D_2 + D_8\bar{D}_1\bar{D}_2D_1$
- (d)  $D_4D_2\bar{D}_1 + \bar{D}_4D_2D_1D_8D_4D_2D_1$

**Q.39** 2's complement representation of a 16-bit number (one sign bit and 15 magnitude bits) is FFFF. Its magnitude in decimal representation is [EC-GATE 1993]

[2-Marks]

- (a) 0
- (b) 1
- (c) 23,767
- (d) 64,535

**Q.40** A Signed Integer has been stored in a byte using 2's complement format. We wish to store the same integer in a sixteen bit word. We should [EC-GATE 1997]

- (a) copy the original byte to the least significant byte of the word and fill the more significant with zeros.
- (b) copy the original byte to the more significant byte of the word and fill the less significant with zeros.
- (c) copy the original byte to the less significant byte of the word and make each bit of the more significant byte equal to most significant bit of the original byte.
- (d) copy the original byte to the less significant byte of the word as well as more significant byte of the word.

**Q.41** An equivalent 2's complement representation of 2's complement number 1101 is

[EC-GATE 1998]

- (a) 110100
- (b) 001101
- (c) 110111
- (d) 111101

**Q.42** The 2's complement representation of -17 is [EC-GATE 2001]

[1-Mark]

- (a) 101110
- (b) 111110
- (c) 101111
- (d) 110001

**Q.43** Assuming all numbers are in 2's complement representation, which of the following numbers is divisible by 11111011? [GATE 2003]

[1-Mark]

- (a) 11100111
- (b) 11100100
- (c) 11010111
- (d) 11011011

**Q.44** The number of distinct Boolean expressions of 4 variables is

[EC-GATE 2003]

[1-Mark]

- (a) 65536
- (b) 1024
- (c) 256
- (d) 16
- (d) 11011011

**Q.45** Let A = 1111 1010 and B = 0000 1010 be two 8-bit 2's complement numbers. Their product in 2's complement is: [GATE 2004]

[2-Marks]

- (a) 1100 0100
- (b) 1001 1100
- (c) 1010 0101
- (d) 1101 0101

**Q.46** If 73 (in base-x number system) is equal to 54, (in base y-number system), the possible values of x and y are: [GATE 2004]

[1-Mark]

- (a) 8, 16
- (b) 10, 12
- (c) 9, 13
- (d) 8, 11

**Q.47** 11001, 1001 and 111001 correspond to the 2's complement representation of which one of the following sets of number? [EC-GATE 2004]

[2-Marks]

- (a) 25, 9 and 57 respectively
- (b) -7, -7 and -7 respectively
- (c) -6, -6 and -6 respectively
- (d) -25, -9 and -57 respectively

**Q.48** The range of signed decimal numbers that can be represented by 6-byte 1's complement number is [EC-GATE 2004]

[1-Mark]

- (a) -32 to +31
- (b) -63 to +63
- (c) -64 to +63
- (d) -31 to +31

**Q.49**

|    |    |
|----|----|
| 15 | 14 |
|    |    |

|   |   |
|---|---|
| 8 | 7 |
|   |   |

      0

The normalization representation for the above format is specified as follows. The mantissa has an implicit 1 preceding the binary (radix) point. Assume that only 0's are padded in while shifting a field. The normalization representation of the above number ( $0.239 \times 2^{13}$ ) is: [GATE 2005]

[2-Marks]

- (a) CA 20
- (b) 11 34
- (c) 49 DO
- (d) 4A E8

**Q.50**

|    |    |
|----|----|
| 15 | 14 |
|    |    |

|   |   |
|---|---|
| 8 | 7 |
|   |   |

      0

The decimal number  $0.239 \times 2^{13}$  has the following hexadecimal representation (without normalization and rounding off): [GATE 2005]

[2-Marks]

- (a) OD 24
- (b) OD 4D
- (c) 4D OD
- (d) 4D 3D

**Q.51** The hexadecimal representation of  $657_8$  is: [GATE 2005]

[1-Mark]

- (a) 1AF
- (b) D78
- (c) D71
- (d) 32F

**Q.52** The range of integers that can be represented by an n bit 2's complement number system is:

[GATE 2005]  
[1-Mark]

- (a)  $-2^{n-1}$  to  $(2^{n-1} - 1)$
- (b)  $-(2^{n-1} - 1)$  to  $(2^{n-1} - 1)$
- (c)  $-2^{n-1}$  to  $2^{n-1}$
- (d)  $-(2^{n-1} + 1)$  to  $(2^{n-1} - 1)$

**Q.53** Decimal 43 in Hexadecimal and BCD number system respectively is [EC-GATE 2005]

[1-Mark]

- (a) B2, 0100 001
- (b) 2B, 0011 0100
- (c) 2B, 0100 0011
- (d) B2, 0100 0100

**Q.54** Using Booth's algorithm for multiplication, the multiplier -57 will be recorded as

[IT-GATE 2005]  
[1 Mark]

- (a) 0 -1 0 0 1 0 0 -1
- (b) 1 1 0 0 0 1 1 1
- (c) 0 -1 0 0 1 0 0 0
- (d) 0 1 0 0 -1 0 0 1

**Q.55** A line L in a circuit is said to have a stuck at 0 fault if the line permanently has a logic value 0. Similarly a line L in a circuit is said to have a stuck at 1 fault if the line permanently has a logic value 1. A circuit is said to have a multiple stuck at fault if one or more lines have stuck at faults. The total number of distinct multiple stuck at faults possible in a circuit with N lines is:

[IT-GATE 2005]  
[2-Marks]

- (a)  $3^N$
- (b)  $3^N - 1$
- (c)  $2^N - 1$
- (d)  $2^N$

**Q.56** We consider the addition of two 2's complement numbers  $b_{n-1}b_{n-2}\dots b_0$  and  $a_{n-1}a_{n-2}\dots a_0$ . A binary adder for adding unsigned binary numbers is used to add the two numbers. The sum is denoted by  $c_{n-1}c_{n-2}\dots c_0$  and the carry-out by  $c_{\text{out}}$ . Which of the following options correctly identifies the overflow condition? [GATE 2006]

- (a)  $c_{\text{out}} \left( \overline{a_{n-1} \oplus b_{n-1}} \right)$
- (b)  $a_{n-1}b_{n-1}\overline{c_{n-1}} + \overline{a_{n-1}b_{n-1}}c_{n-1}$
- (c)  $c_{\text{out}} \oplus c_{n-1}$
- (d)  $a_{n-1} \oplus b_{n-1} \oplus c_{n-1}$

**Q.57** Consider the numbers represented in 4-bit gray code. Let  $h_3h_2h_1h_0$  be the gray code representation of a number  $n$  and let  $g_3g_2g_1g_0$  be the gray code of  $(n + 1)$  (modulo 16) value of the number. Which one of the following functions is correct? [GATE 2006]

[2-Marks]

- (a)  $g_0(h_3h_2h_1h_0) = \Sigma(1, 2, 3, 6, 10, 13, 14, 15)$
- (b)  $g_1(h_3h_2h_1h_0) = \Sigma(4, 9, 10, 11, 12, 13, 14, 15)$
- (c)  $g_2(h_3h_2h_1h_0) = \Sigma(2, 4, 5, 6, 7, 12, 13, 15)$
- (d)  $g_4(h_3h_2h_1h_0) = \Sigma(0, 1, 6, 7, 10, 11, 12, 13)$

**Q.58** A new Binary Coded Pentary (BCP) number system is proposed in which every digit of a base-5 number is represented by its corresponding 3-bit binary code. For example, the base-5 number 24 will be represented by its BCP code 010100. In this numbering system, the BCP code 100010011001 corresponds to the following number in base-5 system

[EC-GATE 2006]

[2-Marks]

- (a) 423
- (b) 1324
- (c) 4231
- (d) 2201

**Q.59** X = 01110 and Y = 11001 are two 5-bit binary numbers represented in two's complement format. The sum of X and Y represented in two's complement format using 6 bits is

[EC-GATE 2007]

[1-Mark]

- (a) 000111
- (b) 001000
- (c) 100111
- (d) 101001

**Q.60** Let  $r$  denote number system radix. The only value(s) of  $r$  that satisfy the equation  $\sqrt{121} = 11$  is/are [GATE 2008]

[1-Mark]

- (a) decimal 10
- (b) decimal 11
- (c) decimal 10 and 11
- (d) any value  $> 2$

**Q.61** In the IEEE floating point representation the hexadecimal value  $0 \times 00000000$  corresponds to

[GATE 2008]

[1-Mark]

- (a) The normalized value  $2^{-127}$
- (b) The normalized value  $2^{-126}$
- (c) The normalized value +0
- (d) The special value +0

**Q.62** The two numbers represented in signed 2's complement form are P = 11101101 and Q = 11100110. If Q is subtracted from P, the value obtained in signed 2's complement form is

[EC-GATE 2008]

[2-Marks]

- (a) 00000111
- (b) 100000111
- (c) 11111001
- (d) 111111001

**Q.63**  $(1217)_8$  is equivalent to [GATE 2009]  
[1-Mark]

- (a)  $(1217)_{16}$
- (b)  $(028F)_{16}$
- (c)  $(2297)_{10}$
- (d)  $(0B17)_{16}$

## ANSWER KEY

|    |   |    |   |    |   |    |   |    |   |
|----|---|----|---|----|---|----|---|----|---|
| 1  | a | 2  | d | 3  | b | 4  | b | 5  | b |
| 6  | a | 7  | a | 8  | d | 9  | a | 10 | b |
| 11 | b | 12 | b | 13 | d | 14 | b | 15 | b |
| 16 | a | 17 | a | 18 | a | 19 | a | 20 | b |
| 21 | b | 22 | c | 23 | b | 24 | c | 25 | d |
| 26 | b | 27 | c | 28 | d | 29 | b | 30 | c |
| 31 | c | 32 | b | 33 | d | 34 | c | 35 | c |
| 36 | b | 37 | a | 38 | b | 39 | b | 40 | c |
| 41 | d | 42 | c | 43 | a | 44 | a | 45 | a |
| 46 | d | 47 | b | 48 | d | 49 | d | 50 | d |
| 51 | a | 52 | a | 53 | c | 54 | a | 55 | c |
| 56 | c | 57 | b | 58 | c | 59 | a | 60 | d |
| 61 | b | 62 | a | 63 | b |    |   |    |   |

## SOLUTIONS

### S.1 (a)

We have, 2's complement of a number is  
 $10101010$

We know that 2's complement represent negative number. The most significant bit is sign bit. Then we have

$$\begin{array}{cccccccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ \Rightarrow & 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ \Rightarrow & -128 + 32 + 8 + 2 = -86 \end{array}$$

### S.2 (d)

In the 2's complement form, a negative number is the 2's complement of the corresponding positive number.

$$-(39)_{10} \Rightarrow 2\text{'s complement}$$

$$\therefore (39)_{10} \Rightarrow (0100111)_2$$

$$1\text{'s complement of } (0100111) \Rightarrow (1011000)$$

$$2\text{'s complement of } (0100111) = \underbrace{(1011001)}_{\text{Magnitude}}$$

$$\therefore 2\text{'s complement of } -39 = \underbrace{\overline{1011001}}_{\substack{\text{Sign} \\ \text{Magnitude}}}$$

**S.5 (b)**

$$837 \xrightarrow[\text{Code-representation}]{\text{Excess-3 BCD}} (?)$$

add 3 to each digit,

|                       |      |      |
|-----------------------|------|------|
| 8                     | 3    | 7    |
| 3                     | 3    | 3    |
| 11                    | 6    | 10   |
| ↓                     | ↓    | ↓    |
| Binary Representation |      |      |
| ↓                     | ↓    | ↓    |
| 1011                  | 0110 | 1010 |

$$\therefore (1011 \ 0110 \ 1010)_2$$

**S.6 (a)**

$0 - 100 + 100 - 1$  evaluates to 57 in Booth 1's coding below

$$(0 \times 2^7) - (1 \times 2^6) + (0 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) - (1 \times 2^0) = -64 + 8 - 1 = -57$$

**S.7 (a)**

In Gray code successive Code words differ by the complementation of a single bit and this logic has been complemented in the given circuit.

**S.8 (d)**

$$2x^2 + x + 1 = 64 + 5 \times 8 + 2 \Rightarrow x = 7$$

**S.9 (a)**

|                |                |                |                |                |                |                |   |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|---|
| b <sub>4</sub> | b <sub>3</sub> | b <sub>2</sub> | p <sub>3</sub> | b <sub>1</sub> | p <sub>2</sub> | p <sub>1</sub> |   |
| Received       | 1              | 1              | 0              | 1              | 1              | 0              | 0 |

$$C_1^* = b_4 \oplus b_3 \oplus b_1 \oplus p_1 = 0$$

$$C_2^* = b_4 \oplus b_3 \oplus b_1 \oplus p_2 = 1$$

$$C_3^* = b_4 \oplus b_3 \oplus b_2 \oplus p_3 = 1$$

C<sub>3</sub>\* C<sub>2</sub>\* C<sub>1</sub>\* = 110 which indicate position 6 in error Transmitted code **1001100**.

**Alternate:**Received Code  $\Rightarrow 1101100$ 

$$\therefore h_1 \ h_2 \ h_3 \ h_4 \ h_5 \ h_6 \ h_7$$

$$1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0$$

$$C_1 = h_1 \oplus h_3 \oplus h_5 \oplus h_7 = 1 \oplus 0 \oplus 1 \oplus 0 = 0$$

$$C_2 = h_2 \oplus h_3 \oplus h_6 \oplus h_7 = 1 \oplus 0 \oplus 0 \oplus 0 = 1$$

$$C_3 = h_4 \oplus h_5 \oplus h_6 \oplus h_7 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

$$\therefore C_3 C_2 C_1 = (010)_2 = 2;$$

Hence bit 2 is in error correct code word is obtained by complementing 2 bit  $\Rightarrow 1001100$

**S.10 (b)**

$$1 \times 2^6 + 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^2 + 2^0 = 109$$

$$1 \times 2^{-1} + 1 \times 2^{-3} + 1 \times 2^{-4} = 0.5 + 0.125 + 0.625 = 0.6875$$

**S.11 (b)**

$$\text{Excess-3} = 843$$

$$\text{Decimal no.} = (8 - 3)(4 - 3)(3 - 3) = 510$$

**S.12 (b)**

$$12 - (-9) = 21 = 00010101$$

**S.13 (d)**

Carry of a one bit full adder is given by expression  $xy + yz + zx$ .

**S.14 (b)**

All are 2's complement of 9

$$110111 \Rightarrow 001000$$

$$\begin{array}{r} + 1 \\ 001001 \\ \hline 0001001 = 9_{10} \end{array}$$

$$10111 \Rightarrow 01000$$

$$\begin{array}{r} + 1 \\ 01001 \\ \hline 001001 = 9_{10} \end{array}$$

$$1110111 \Rightarrow 0001000$$

$$\begin{array}{r} + 1 \\ 0001001 \\ \hline 00001001 = 9_{10} \end{array}$$

**S.15 (b)**

The given excess-3 code is:

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

To convert it into decimal, first the groups of 4 bits are made starting from right hand side. The each group code is converted into its decimal equivalent by subtracting 3 from each group.

$$\begin{array}{r} 0011 & 0101 & 1010 & 0100 \\ \underline{-3} & \underline{-5} & \underline{-10} & \underline{-4} \\ -3 & -3 & -3 & -3 \end{array}$$

Hence, the decimal equivalent of given  $X_{8-3}$  will be **0271**.

**S.16 (a)**

$$(110101)_2 = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$= (53)_{10}$$

**2.1.10****DIGITAL LOGIC**

a) b) Converting 53 into octal:

$$\begin{array}{r} 8 \mid 53 \mid 5 \\ \downarrow \\ 6 \end{array} \quad = (65)_8$$

**S.17 (a)**

The 10's complement of no. N with n bits is given as  $(10)^n - N$

The 10's complement of 47480 is

$$10^5 - 47480 = 52520$$

**S.18 (a)**

Octal equivalent of  $(52520)_{10}$  is  $(146450)_8$

**S.19 (a)**

$$(152)_8 = 1 * 8^2 + 5 * 8 + 2 * 8^0$$

$$= 64 + 40 + 2 = (106)_{10}$$

$$\text{Now, } 2x^2 + x + 1 = 106,$$

$$\text{or, } 2x^2 + x - 105 = 0,$$

$$\text{or } x = \frac{-1 \pm \sqrt{1+4*2*105}}{2*2}$$

$$= \frac{-1 \pm \sqrt{1+840}}{4}$$

$$\text{or, } x = \frac{-1 \pm \sqrt{1+841}}{4} = \frac{-1 \pm 29}{4}$$

$$= -7.5, +7$$

**S.20 (b)**

$$296_{12}$$

$$\times 57_{12}$$

$$\underline{1766}$$

$$\underline{11B6}$$

$$\underline{13706}$$

**S.21 (b)**

|                  |                                                |
|------------------|------------------------------------------------|
| $296_{14}$       | $392 + 126 + 6 = 524$                          |
| $\times 57_{14}$ | $70 + 7 = \frac{77}{3668}$                     |
| $14A0$           | $\underline{3668}$                             |
| $D52$            | $\underline{\text{Decimal} \Rightarrow 40348}$ |
| $109C0_{14}$     |                                                |

**S.22 (c)**

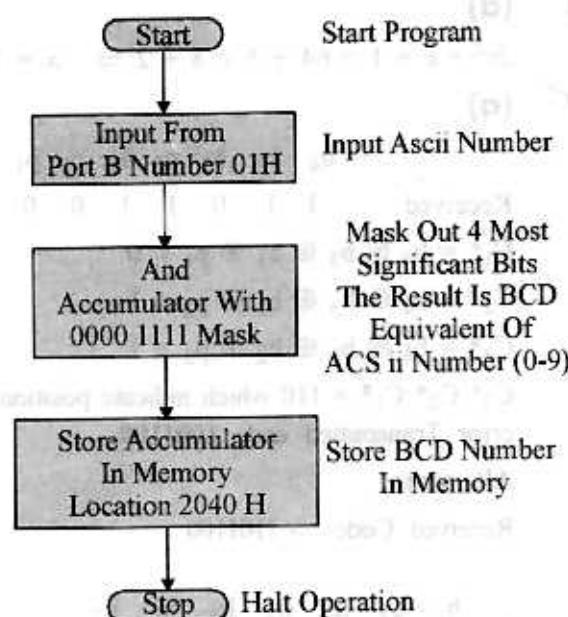
|               |     |   |                 |
|---------------|-----|---|-----------------|
| 7             | 250 | 5 | $505_7$         |
| 7             | 35  | 0 | $5 * 7^2 = 245$ |
|               | 5   |   | $5 * 7^0 = 5$   |
|               |     |   | 250             |
| 0.5 * 7 = 3.5 |     |   |                 |
| 0.5 * 7 = 3.5 |     |   | $(505.3..)_7$   |

**S.23 (b)**

|    |     |   |                  |
|----|-----|---|------------------|
| 12 | 250 | A | $18A_{12}$       |
| 12 | 20  | 8 | $1 * 12^2 = 144$ |
|    | 1   |   | $8 * 12^1 = 96$  |
|    |     |   | $A * 12^0 = 10$  |
|    |     |   | 250              |
|    |     |   | $(18A.0)_{12}$   |

**S.24 (c)**

The flowchart in figure represents a program segment that inputs an ASCII coded number and masks out the most significant 4-bits. The result is the BCD equivalent of the ASCII coded number.



Figure

**S.26 (b)**

If it is divisible by 4, then both the two least significant bits has to be 0. So, only option (b) is divisible by 4.

## S.27 (c)

$$10110_2 = 2^5 + 2^2 + 2^1 = 38_{10}$$

$$26_{16} = 2 \times 16 + 6 = 38_{10}$$

$$46_8 = 4 \times 8 + 6 = 38_{10}$$

$$212_4 = 2 \times 4^2 + 6^1 = 38_{10}$$

So  $36_{10}$  is not equivalent.

## S.28 (d)

See a example

$$42 \text{ in a byte} \quad 00101010$$

$$42 \text{ in a word} \quad 0000000000101010$$

$$-42 \text{ in a byte} \quad 11010110$$

$$-42 \text{ in a word} \quad 11111111010110$$

Therefore (d) is correct.

## S.29 (b)

The biased exponent is  $10010001 = 145$

$$\begin{aligned}\text{Number} &= (-1)^{\ell}(1+F)(2^{E-127}) \\ &= (-1)^{\ell}(1.10001110001)(2^{145-127}) \\ &= (-1)^{\ell}(1.10001110001)(2^{18}) \\ &= -1100011100010000000\end{aligned}$$

## S.30 (c)

Here  $\bar{A}, \bar{B}$  are 1's complement

$$A + B,$$

$$A \quad 01010110$$

$$B \quad \underline{+11101100}$$

$$101000010$$

$$\underline{+ \quad \quad \quad 1}$$

$$01000011$$

$$B - A = B + \bar{A},$$

$$B \quad 11101100$$

$$\bar{A} \quad \underline{+10101001}$$

$$110010101$$

$$\underline{+ \quad \quad \quad 1}$$

$$10010110$$

$$A - B = A + \bar{B},$$

$$A \quad 01010110$$

$$\bar{B} \quad \underline{+00010011}$$

$$01101001$$

(d) 86.2

$$-A - B = \bar{A} + \bar{B},$$

$$\bar{A} \quad 10101001$$

$$\bar{B} \quad \underline{+00010011}$$

$$10111100$$

## S.31 (c)

The given number has whole number (d) and fraction (F) 14 and .34 respectively. The conversion of given decimal number into binary is as follows:

**Whole number:** The whole number is converted into binary by dividing it by 2 as the method explained above. The remaining bits show the binary bits of its binary equivalent.

Remainder

|   |    |   |     |
|---|----|---|-----|
| 2 | 14 | 0 | LSB |
| 2 | 7  | 1 |     |
| 2 | 3  | 1 |     |
| 2 | 1  | 1 | MSB |
| 2 | 0  |   |     |

The binary equivalent of 14 is

$$(14)_{10} = (1110)_2$$

**Fraction:** The fraction part is converted into binary by multiplying it by 2.

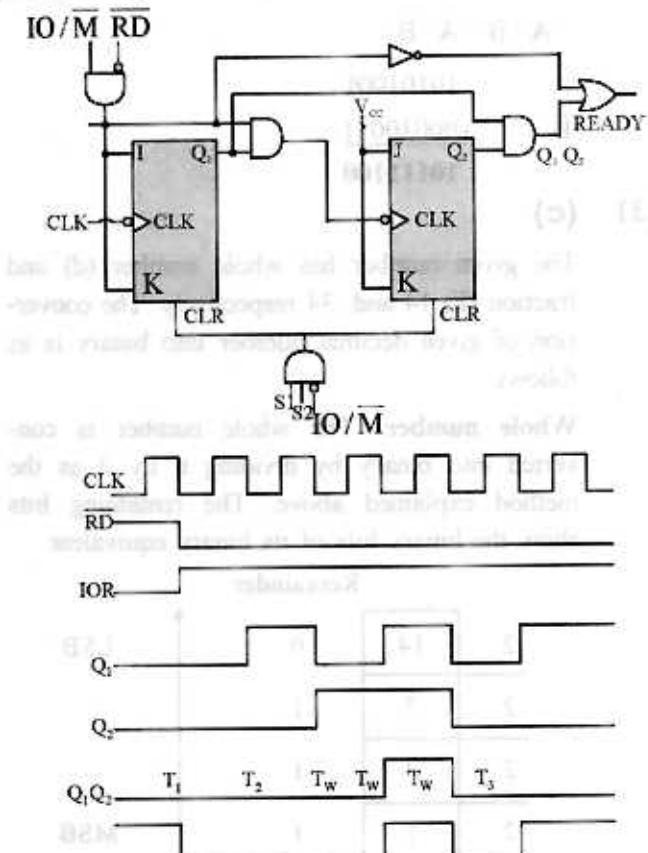
| Product                | Whole Number |          | Fraction                      |
|------------------------|--------------|----------|-------------------------------|
| $0.34 \times 2 = 0.68$ | 0            | $a_{-1}$ | MSB    0.68    F <sub>1</sub> |
| $0.68 \times 2 = 1.36$ | 1            | $a_{-2}$ | 0.36    F <sub>2</sub>        |
| $0.36 \times 2 = 0.72$ | 0            | $a_{-3}$ | 0.72    F <sub>3</sub>        |
| $0.72 \times 2 = 1.44$ | 1            | $a_{-4}$ | 0.44    F <sub>4</sub>        |
| $0.44 \times 2 = 0.88$ | 0            | $a_{-5}$ | LSB    0.88    F <sub>5</sub> |

The binary equivalent of  $(0.34)_{10}$  is  $(0.01010)_2$  so that binary equivalent of  $(14.34)_{10}$  is

$$(1110.01010)_2$$

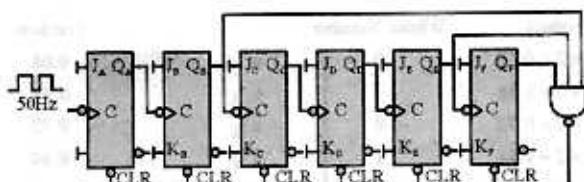
## S.32 (b)

The two J-K Flip-Flops form a 2-bit counter. This counter is cleared when  $S_1 = S_2 = IO/\bar{M} = 1$  i.e., OP Code Fetch is started. It then counts through 00, 01, 11. The 11 state ( $Q_1 = Q_2 = 1$ ) decoded is used to end the waiting, by pulling the Ready up.



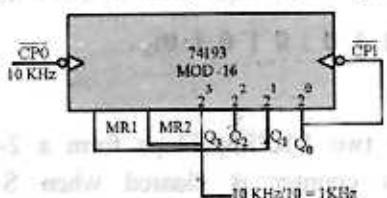
## S.33 (d)

As  $2^5 = 32$  and  $2^6 = 64$ , count 50 will require 6-FFs. The counter will be cleared when it reaches 50 (1 10010). Thus the output  $Q_F$ ,  $Q_E$  and  $Q_B$  of FFs should be connected to inputs of the NAND gate. The frequency of the  $Q_F$  is 1 Hz.



## S.34 (c)

Mod-10 counter require 4-flip-flop hence  $Q_0$  is



connected to CP1. In this case, we want that the counter should recycle back to 0000 from 1010

## S.36 (b)

The number of Boolean functions that can be generated by  $n$  variables =  $2^{2^n}$

## S.38 (b)

| D <sub>8</sub> | D <sub>4</sub> | D <sub>2</sub> | D <sub>1</sub> | Y |
|----------------|----------------|----------------|----------------|---|
| 0              | 0              | 0              | 0              | 1 |
| 0              | 0              | 0              | 1              | 0 |
| 0              | 0              | 1              | 0              | 0 |
| 0              | 0              | 1              | 1              | 1 |
| 0              | 1              | 0              | 0              | 0 |
| 0              | 1              | 0              | 1              | 0 |
| 0              | 1              | 1              | 0              | 1 |
| 0              | 1              | 1              | 1              | 0 |
| 1              | 0              | 0              | 0              | 0 |
| 1              | 0              | 0              | 1              | 1 |

$$Y = D_8 D_1 + \bar{D}_4 D_2 D_1 + \bar{D}_1 D_2 D_4 + \bar{D}_8 \bar{D}_4 \bar{D}_2 \bar{D}_1$$

## S.39 (b)

$$FFFF-0001 = FFFE$$

Taking complement, we get 0001.

## S.41 (d)

$$(1101)_2 \xrightarrow{1's} 0010 \xrightarrow{2's} 0011 \\ (0011)_2 \Rightarrow (00\ 0001) \xrightarrow{1's} (11\ 1100) \\ (11\ 1100) \xrightarrow{2's} (11\ 1101)$$

## S.42 (c)

The 2's binary representation of 17 is 010001

Its 1's complement is 101110

So 2's complement is

$$\begin{array}{r} 1\ 0\ 1\ 1\ 1\ 0 \\ + \quad \quad \quad 1 \\ \hline 1\ 0\ 1\ 1\ 1\ 1 \end{array}$$

## S.43 (a)

| 2'S Complement        | 1'S Complement | Original Number | Decimal Value |
|-----------------------|----------------|-----------------|---------------|
| 11100111              | 11100110       | 00011001        | 25            |
| 11100100              | 11100011       | 00011100        | 28            |
| 11010111              | 11010110       | 00101001        | 41            |
| 11011011              | 11011010       | 00100101        | 37            |
| 11111011<br>(Divider) | 11111010       | 00000101        | 5             |

So 25 is divisible by 5 so we can say 11100111 is divisible by 11111011.

| D <sub>2</sub> D <sub>1</sub> |             |
|-------------------------------|-------------|
| D <sub>8</sub> D <sub>4</sub> | 00 01 11 10 |
| 00                            | 1 1 1 0     |
| 01                            | 1 1 1 1     |
| 11                            | X X X X     |
| 10                            | 1 X X X     |

**S.44 (a)**

The number of distinct Boolean expression of  $n$  variable is  $2^{2^n}$ . Thus

$$2^{2^n} = 2^{2^4} = 2^{16} = 65536$$

**S.45 (a)**

$$\begin{aligned} A &= 1111010 \\ &= -2^7 \times 1 + 2^6 \times 1 + 2^5 \times 1 + 2^4 \times 1 + 2^3 \times 1 + 2^2 \times 0 + 2^1 \times 1 + 2^0 \times 0 \\ &= -2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 0 + 2^1 + 0 \\ &= -128 + 64 + 32 + 16 + 8 + 2 \\ &= -128 + 122 \\ &= -6 \end{aligned}$$

$$\begin{aligned} B &= 00001010 \\ &= 2^3 \times 1 + 2^1 \times 1 \\ &= 8 + 2 = 10 \\ A \cdot B &= (-6) \times 10 \\ &= -60 \end{aligned}$$

So the value of  $AB$  in 2's complement representation

$$AB = 11000100$$

**S.46 (d)**

We have

$$\begin{aligned} (73)_x &= (54)_y \\ (7 \times x^1 + 3 \times x^0)_{10} &= (5 \times y^1 + 4 \times y^0)_{10} \\ (7x + 3)_{10} &= (5y + 4)_{10} \\ \Rightarrow 7x - 5y &= 1 \end{aligned}$$

Now by Hit and Trial Method,

We have 8 and 11 which satisfies the equation.

Hence, 8 and 11 are the value of  $x$  and  $y$  respectively.

**S.47 (b)**

$$\begin{array}{r} 1 \ 1 \ 0 \ 0 \ 1 \\ 0 \ 0 \ 1 \ 1 \ 0 \\ + \qquad \qquad 1 \\ \hline 0 \ 0 \ 1 \ 1 \ 1 \end{array} \qquad \begin{array}{r} 1 \ 0 \ 0 \ 1 \\ 0 \ 1 \ 1 \ 0 \\ + \qquad \qquad 1 \\ \hline 0 \ 1 \ 1 \ 1 \end{array}$$

(b) 88.2

1 1 1 0 0 1

(b) 88.2

0 0 0 1 1 0

(b) 88.2

+ 1 1 1 1

(b) 88.2

0 0 0 1 1 1

(b) 88.2

7

(b) 88.2

Thus 2's complement of 11001, 1001 and 111001 is 7. So the number given in the question are 2's complement correspond to -7.

**S.48 (d)**

The range of signed decimal numbers that can be represented by  $n$ -bits 1's complement number is  $-(2^{n-1} - 1)$  to  $+(2^{n-1} - 1)$ .

Thus for  $n = 6$ , we have

$$\begin{aligned} \text{Range} &= -(2^{6-1} - 1) \text{ to } +(2^{6-1} - 1) \\ &= -31 \text{ to } +31. \end{aligned}$$

**S.51 (a)**

$$\begin{aligned} (657)_8 &= 6 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 \\ &= 6 \times 64 + 5 \times 8 + 7 \times 1 \\ &= 384 + 40 + 7 \\ &= 431 \end{aligned}$$

|    |     |    |
|----|-----|----|
| 16 | 431 | 15 |
| 16 | 26  | 10 |
| 1  |     |    |
| 1  | A   | F  |
| 1  |     |    |

**S.52 (a)**

2's complement of a number represents the negative number. If a number is  $n$ -bit long the most significant digit is a sign bit so starting from  $-2^{n-1}$  because only  $n-1$  place we represent the magnitude. So, the range is  $-2^{n-1}$  to  $(2^{n-1} - 1)$  including 0.

**S.53 (c)**

$$(43)_{10} \rightarrow \begin{array}{c|c|c} 16 & 43 \\ \hline 16 & 32 & B \\ \hline & 2 & \end{array} \rightarrow (2B)_{16}$$

Thus  $(43)_{10} \rightarrow (2B)_{16}$

Now  $(4)_{10} \rightarrow (0100)_2$

$(3)_{10} \rightarrow (0011)_2$

Thus  $(43)_{10} \rightarrow (01000011)_{BCD}$

**S.54 (a)**

In Booth's encoding technique a number is represented with value L-R.

where L is the weight of the zero before the leftmost 1 and R is the weight of rightmost 1.

-57 in 2's complement is 11000111

in Booth's form

$$0 - 100 + 100 - 1$$

**S.55 (c)**

Here, applying concept of total number of substack of fault. If there are 'N' lines then, total possible conditions are  $2^N$ , but at the case when all N lines are having '1', then no fault occurs. Hence, the possible conditions for fault becomes  $2^N - 1$ .

**S.56 (c)**

$$B = b_{n-1}b_{n-2} \dots b_0$$

$$A = a_{n-1}a_{n-2} \dots a_0$$

$$S = c_{n-1}c_{n-2} \dots c_0$$

The overflow condition V

$$V = b'_{n-1}a'_{n-1}c_{n-2} + b_{n-2}a_{n-1}c'_{n-2}$$

$c_{n-1}$  is  $c_{out}$  so:

$$V = c_{out} \oplus c_{n-1}$$

So, the overflow condition is

$$c_{out} \oplus c_{n-1}$$

**S.57 (b)**

$h_3h_2h_1h_0$  is the gray code representing of number n,  $g_3g_2g_1g_0$  is the gray code representing of  $(n+1)(\text{modulo } 16)$ .

$(n+1)$  is 5 bit in decimal its range is 0 to 31. So  $g_1(h_3h_2h_1h_0) = \Sigma(4, 9, 10, 11, 12, 13, 14, 15)$  is the correct function for  $(n+1) \bmod 16$ .

**S.58 (c)**

$$\begin{array}{cccccc} 1 & 0 & 0 & 1 & 0 & 0 \\ \hline 4 & 2 & 3 & 1 & & \end{array} \Rightarrow 4231$$

**S.59 (a)**

MSB of Y is 1, thus it is negative number and X is positive number.

Now we have  $X = 01110 = (14)_{10}$

and  $Y = 11001 = (-7)_{10}$

$$X + Y = (14) + (-7) = 7$$

In signed two's complement form 7 is

$$(7)_{10} = 000111.$$

**S.60 (d)**

We have

$$\sqrt{121_r} = 11_r$$

Let us suppose

Condition I  $\rightarrow$  When  $r = 10$

$$(121)_r = (121)_{10}$$

$$(11)_r = (11)_{10}$$

$\therefore$  It satisfies the equation.

Condition II  $\rightarrow$  When  $r = 8$

$$(121)_r = (121)_8 = (81)_{10}$$

$$(11)_r = (11)_8 = (9)_{10}$$

$\therefore$  It satisfies the equation.

Condition III  $\rightarrow$  When  $r = 16$

$$(121)_r = (121)_{16} = (289)_{10}$$

$$(11)_r = (11)_{16} = (17)_{10}$$

$\therefore$  It satisfies the equation.

Hence the given equation is satisfied for  $r > 2$ .

**S.61 (d)**

Hexadecimal value of  $0 \times 00000000$  in binary

0000 0000 0000 0000 0000 0000 0000 0000

So, it is normalized to value to 0.

**S.62 (a)**

MSB of both number are 1, thus both are negative number. Now we get

$$11101101 = (-19)_{10}$$

$$\text{and } 11100110 = (-26)_{10}$$

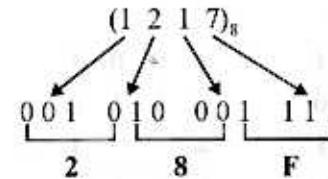
$$P - Q = (-19) - (-26) = 7$$

Thus 7 in signed 2's complement form is

$$(7)_{10} = 00000111.$$

**S.63 (b)**

$$(1217)_8 = (028F)_{16}$$



To appear, click on the image at the bottom of the page. 21.0  
and many more in our collection. 22.0

July 2014 2.0 last on table w/ books incl' 9.0  
negative charges left to TTL switch  
non-volatile mode (10% 20)

# LOGIC FUNCTIONS AND MINIMIZATION

**2.2**

## LEVEL-1

**Q.1** If one of the input to an \_\_\_\_\_ gate is inverted then it becomes an INHIBITOR.

- (a) AND
- (b) NAND
- (c) NOR
- (d) XOR

**Q.2**  $[(A + \bar{A}B)(A + \bar{A}\bar{B})][(CD + \bar{C}\bar{D})] + (C \oplus D)] =$

- (a) B
- (b) A
- (c) 0
- (d) 1

**Q.3**  $A = xyz + \bar{x}yz + \bar{x}\bar{y}z + xy\bar{z}$  To implement the above Boolean expression, minimum no. of gates required are

- (a) 1
- (b) 2
- (c) 3
- (d) 4

**Q.4** Which of the following logic families has the highest noise immunity?

- (a) RTL
- (b) HTL
- (c) TTL
- (d) DTL

**Q.5** Which of the following logic has the maximum fanout?

- (a) CMOS
- (b) ECL
- (c) IIL
- (d) RTL

**Q.6** Which of the following logic families has the lowest power-delay product?

- (a) IIL
- (b) TTL
- (c) ECL
- (d) MOS

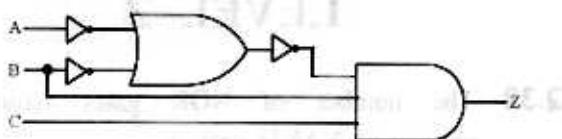
**Q.7** Which TTL sub-family has maximum speed?

- (a) Standard TTL
- (b) Low power TTL
- (c) High speed TTL
- (d) Schottky-clamped TTL

**Q.8** The switching speed of ECL is very high because

- (a) it uses positive logic
- (b) it uses negative logic
- (c) its transistors remain unsaturated
- (d) it uses high speed transistors

- Q.9** What should be added so that a CMOS gate drives TTL at the same supply voltage?
- Pull down transistor
  - A pull up transistor
  - A high value capacitor
  - A buffer
- Q.10** The propagation delay for ECL IC family is approximately
- 2 ns
  - 10 ns
  - 25 ns
  - 50 ns
- Q.11** The digital IC which does not use bipolar transistor technology is
- MOSFET
  - TTL
  - ECL
  - I<sup>2</sup>L
- Q.12** The maximum number of TTL loads that a TTL device can drive reliable over the specified temperature range is
- chip
  - fanout
  - bipolar
  - loading-point
- Q.13** FET's are used in linear ICs to
- increase device complexity
  - provide large resistances
  - increase input resistance
  - both (b) and (c)
- Q.14** Which of the following logic expression is incorrect?
- $1 \oplus 0 = 1$
  - $1 \oplus 1 \oplus 1 = 1$
  - $1 \oplus 1 \oplus 0 = 1$
  - $1 \oplus 1 = 0$
- Q.15** Let X and Y be the input and Z be the output of a NAND gate. The value of Z is given by
- $\overline{X} \cdot \overline{Y}$
  - $X + Y$
  - $X \cdot Y$
  - $\overline{X} \cdot \overline{Y}$
- Q.16** What logic gate is represented by the circuit shown below?
- 
- The circuit diagram shows a rectangular battery symbol with a '+' terminal at the top and a '-' terminal at the bottom. A horizontal line segment connects the '+' terminal to a switch. From the other end of the switch, a line goes up to a bulb symbol, and another line goes down from the bulb back to the '-' terminal of the battery.
- AND
  - NAND
  - NOR
  - EQUIVALENCE
- Q.17** Which of the following boolean algebra expression is incorrect?
- $AB + \overline{A}\overline{B}C + A = A + BC$
  - $A\overline{B} + \overline{A}B\overline{C} + \overline{A}BCD + \overline{A}\overline{B}C\overline{D}E = \overline{A}B$
  - $(A + \overline{A})(AB + A\overline{B}\overline{C}) = AB$
  - $AB + (\overline{A} + \overline{B})C + AB = AB + \overline{A}C + \overline{B}C$
- Q.18** The minterms corresponding to decimal number 15 is
- ABCD
  - $\overline{A}\overline{B}\overline{C}\overline{D}$
  - $\overline{A} + \overline{B} + \overline{C} + \overline{D}$
  - $A + B + C + D$
- Q.19**  $Z = ?$
- 
- The circuit diagram consists of two AND gates. The first AND gate has input A and a NOT gate as its other input. The output of this gate is connected to the second AND gate, along with input B. The second AND gate also has input C. The output of the second AND gate is labeled Z.
- $A\overline{B}C$
  - $\overline{A}BC$
  - 0
  - $ABC$

**Q.20**  $Z = ?$ 

- (a) 0  
 (b)  $\bar{A}\bar{B}C$   
 (c) ABC  
 (d)  $\bar{A}BC$

**Q.21** The no. of required gates to represent XOR using NAND only are

- (a) 3  
 (b) 6  
 (c) 4  
 (d) 4

**Q.22** The maxterm designator of the term

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

- (a) 10  
 (b) 13  
 (c) 05  
 (d) None of these

**Q.23** Given the following Karnaugh map, which one of the following represents the minimal Sum-of-Products of the map?

|  |  | wx | 00 | 01 | 11 | 10 |   |
|--|--|----|----|----|----|----|---|
|  |  | yz | 00 | 0  | X  | 0  | X |
|  |  | 00 | 0  | X  | 1  | X  | 1 |
|  |  | 01 | X  | 1  | X  | 1  |   |
|  |  | 11 | 0  | X  | 1  | 0  |   |
|  |  | 10 | 0  | 1  | X  | 0  |   |

- (a)  $w'x + y'z$   
 (b)  $wx'y' + xy + xz$   
 (c)  $xy + y'z$   
 (d)  $xz + y$

**Q.24** For the identity  $AB + \bar{A}C + BC = AB + \bar{A}C$ , the dual form is

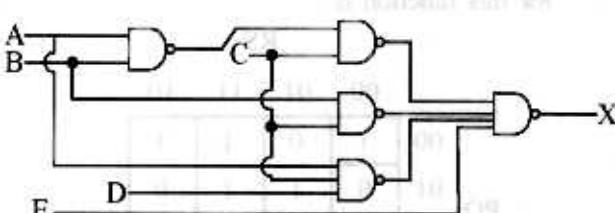
- (a)  $\bar{AB} + \bar{AC} + \bar{BC} + \bar{AB} + AC$   
 (b)  $(\bar{A} + \bar{B})(\bar{A} + \bar{C})(\bar{B} + \bar{C}) = (\bar{A} + \bar{B})(A + \bar{C})$   
 (c)  $(A + B)(\bar{A} + C)(B + C) = (\bar{A} + \bar{B})(A + \bar{C})$   
 (d)  $(A + B)(\bar{A} + C)(B + C) = (A + B)(\bar{A} + C)$

**Q.25** The number of NOR gates required to implement EX-NOR gate

- (a) 4  
 (b) 3  
 (c) 5  
 (d) 6

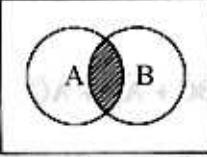
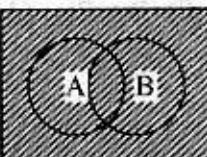
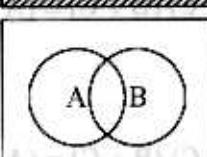
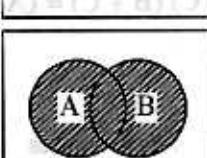
**Q.26** The precedence order while solving Boolean expressions is

- (a) ( ) < OR < AND < NOT  
 (b) ( ) > NOT > AND > OR  
 (c) ( ) < NOT < AND < OR  
 (d) ( ) < AND > NOT > OR

**Q.27** The below circuit is expressed as

- (a)  $X = (A' + B)C + (BD)' + ACE + D$   
 (b)  $X = (A' + B')C + BC + ACD + E'$   
 (c)  $X = (A' + B)C + B'D + (ACD)' + E'$   
 (d)  $X = (A' + B)C + B'D + ACD' + E$

**Q.28** The expression  $A + A'B$  is represented by

- (a) 
- (b) 
- (c) 
- (d) 

**Q.29** Which of the following gates is a series circuit gate?

- (a) AND gate
- (b) XOR gate
- (c) OR gate
- (d) None of these

**Q.30** Which one of the following is equivalent to OR-AND realization?

- (a) NAND-NOR realization
- (b) NAND-NAND realization
- (c) NOR-NOR realization
- (d) Both a and b

**Q.31** The K-map for a Boolean function is shown in figure. The number of essential prime implicants for this function is:

|    |  | RS |    |    |    |   |
|----|--|----|----|----|----|---|
|    |  | 00 | 01 | 11 | 10 |   |
| PQ |  | 00 | 1  | 0  | 1  | 1 |
|    |  | 01 | 0  | 1  | 1  | 0 |
|    |  | 11 | 0  | 1  | 0  | 0 |
|    |  | 10 | 1  | 0  | 0  | 1 |

- (a) 5
- (b) 8
- (c) 4
- (d) 6

## LEVEL-2

**Q.32** The number of NOR gates required to implement NAND gate is

- (a) 3
- (b) 4
- (c) 5
- (d) 2

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

- (a)  $\bar{A} + B$
- (b)  $A + \bar{B}$
- (c) A
- (d) B

$$(NOR).(XOR).(NAND) =$$

- (a) NOR
- (b) NAND
- (c) XOR
- (d) XNOR

**Q.35** The 0 V reading on the output of an LSTTL gate indicates that the output circuit is

- (a) defective
- (b) short circuited to ground
- (c) either (a) or (b) above
- (d) neither (a) nor (b) above

**Q.36** What is(are) the advantage(s) of MOS devices over bipolar device?

- (a) It allows higher bit densities and is also more cost effective.
- (b) It is easy to fabricate.
- (c) It has higher impedance.
- (d) All of these.

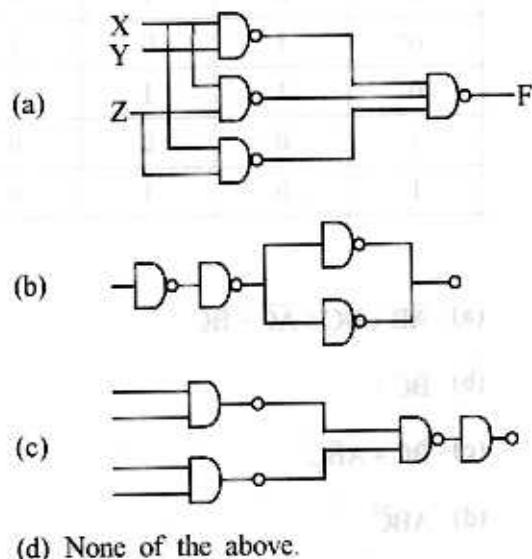
**Q.37** The main advantages of CMOS as compared to TTL are

- (a) higher speed operation with low power consumption
- (b) lower power consumption with low fanout
- (c) lower power consumption and better noise margins
- (d) higher power consumption with high fanout

**Q.38** A + B can be implemented by

- NAND gates alone
- NOR gates alone
- both (a) and (b)
- none of these

**Q.39** Circuit of majority function with 4-NAND gates



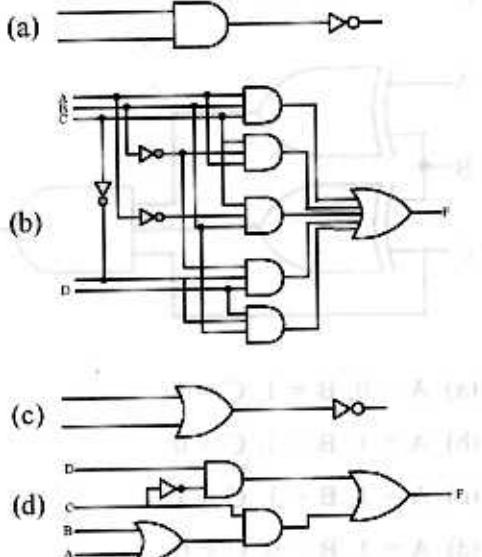
**Q.40** Minterms of the function are

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

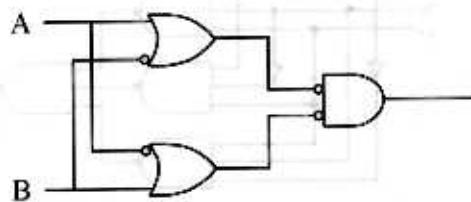
- $\Sigma 1, 5, 6, 7, 9, 10, 11, 13, 14, 15$
- $\Sigma 1, 6, 7, 9$
- $1, 2, 5, 7, 9$
- none.

**Q.41** Logic diagram using original Boolean expression

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

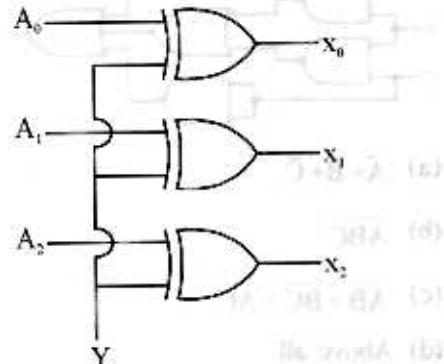


**Q.42** What is the Boolean expression for the following circuit?



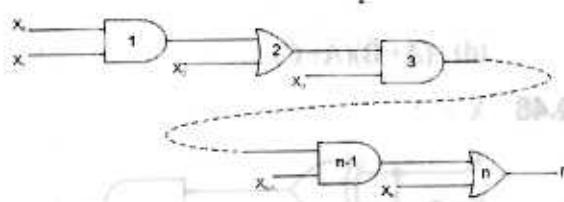
- $F(A,B) = 0$  (zero) (inconsistency)
- $F(A,B) = (A + B')(B + A')$
- $F(A,B) = 1$  (Tautology)
- $F(A,B) = A \oplus B$  (A exclusive OR'ed with B)

**Q.43** In the figure shown  $x_2x_1x_0$  will be 1's complement of  $A_2A_1A_0$  if



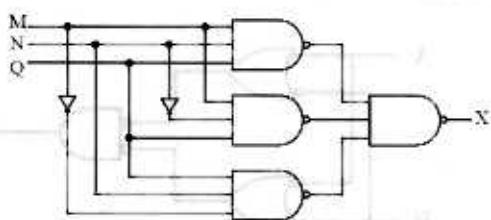
- $Y = 1$
- $Y = 0$
- $Y = \bar{A}_0 = \bar{A}_1 = \bar{A}_2$
- $Y = A_0 = A_1 = A_2$

**Q.44** In the given network of AND and OR gates f can be written as:



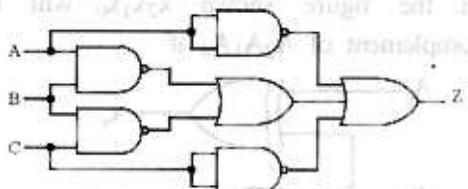
- $x_0 x_1 x_2 \dots x_n + x_1 x_2 \dots x_n + x_2 x_3 \dots x_n \dots x_n$
- $x_0 x_1 x_3 \dots x_{n-1} + x_2 x_3 x_5 \dots x_{n-1} + \dots + x_{n-2} x_{n-1} + x_n$
- $x_0 + x_1 + x_2 + \dots + x_n$
- $x_0 x_1 + x_2 x_3 + \dots + x_{n-1} x_n$

**Q.45**  $X = ?$



- (a)  $Q(M+N)$
- (b)  $MNQ = \overline{M}(\overline{N} + A) = \overline{B}A\overline{B}$
- (c)  $N(Q+M)$
- (d)  $M(Q+N)$

**Q.46**  $Z = ?$

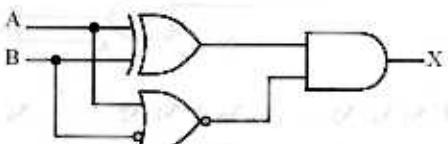


- (a)  $\overline{A} + \overline{B} + \overline{C}$
- (b)  $\overline{ABC}$
- (c)  $\overline{AB} + \overline{BC} + \overline{AC}$
- (d) Above all

**Q.47** Given that  $AB + \overline{AC} + BC = AB + \overline{AC}$ , then  $(\overline{A} + C)(B + C)(A + B)$  is equivalent to

- (a)  $(\overline{A} + B)(A + C)$
- (b)  $(A + \overline{B})(\overline{A} + C)$
- (c)  $(A + B)(\overline{A} + C)$
- (d)  $(A + \overline{B})(\overline{A} + \overline{C})$

**Q.48**  $X = ?$



- (a)  $AB$
- (b)  $\overline{AB}$
- (c)  $A\overline{B}$
- (d)  $0$

**Q.49** For the truth table shown below expression for  $\overline{X}$  is

| A | B | C | X |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |

(a)  $\overline{AB} + BC + \overline{AC} + \overline{BC}$

(b)  $\overline{BC}$

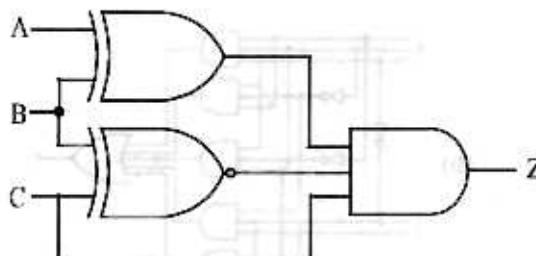
(c)  $\overline{BC} + A\overline{BC}$

(d)  $ABC$

**Q.50** Expression  $A + \overline{AB} + \overline{ABC} + \overline{ABCD} + \overline{ABCDE}$  would be simplified to

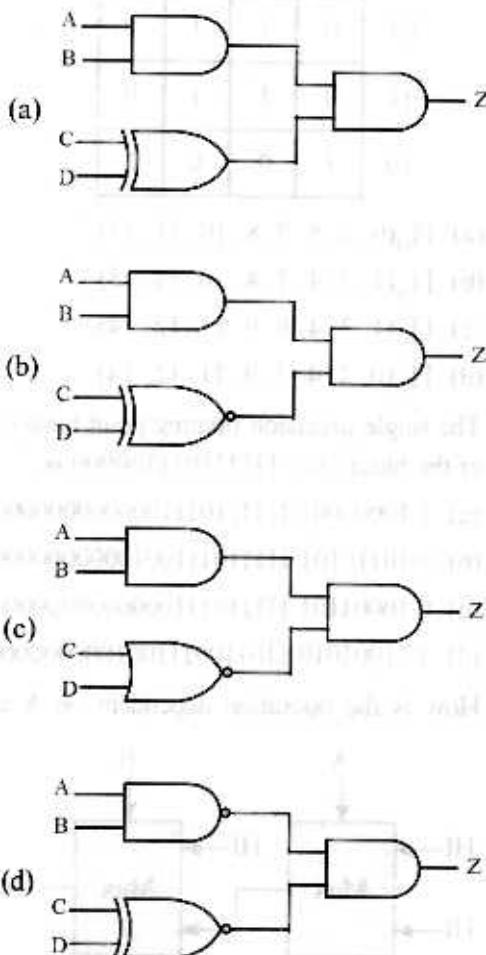
- (a)  $A + \overline{AB} + CD + E$
- (b)  $A + B + C + D + E$
- (c)  $A + B + CDE$
- (d)  $A + BC + CD + DE$

**Q.51** In below fig. the input condition needed to produce  $Z = 1$ , is

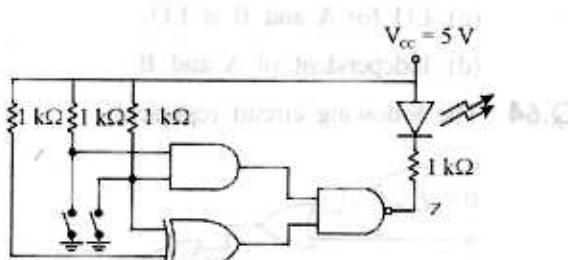


- (a)  $A = 0, B = 1, C = 1$
- (b)  $A = 1, B = 1, C = 0$
- (c)  $A = 1, B = 1, C = 1$
- (d)  $A = 1, B = 0, C = 0$

- Q.52** The output of logic circuit is HIGH whenever A and B are both HIGH as long as C and D are either both LOW or both HIGH. The logic circuit is

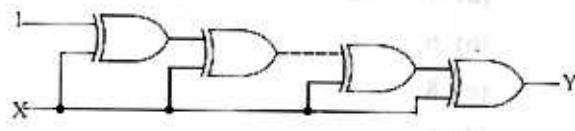


- Q.53** In below fig. the LED emits light when



- (a) both switch are closed
- (b) both switch are open
- (c) LED does not emit light irrespective of the switch positions
- (d) only one switch is closed

- Q.54** If the input to digital circuit shown in below fig. consisting of a cascade of 20 XOR gates is X, then the output Y is equal to



- (a) X
- (b)  $\bar{X}$
- (c) 1
- (d) 0

- Q.55** For the K-map shown in below fig. the minimized function in SOP form is

|    |    | YZ |    |    |    |
|----|----|----|----|----|----|
|    |    | 00 | 01 | 11 | 10 |
| WX | 00 | 1  | 1  |    | 1  |
|    | 01 |    |    |    |    |
| 11 |    |    | 1  | 1  |    |
| 10 | 1  |    | 1  | 1  |    |

- (a)  $\overline{WXY} + WY + \overline{XYZ} + \overline{WXZ}$
- (b)  $\overline{WXY} + WY + \overline{WXY\bar{Z}} + W\overline{XYZ}$
- (c)  $\overline{WXY} + \overline{XZ} + WY$
- (d)  $\overline{WXY} + WY + \overline{WXZ}$

- Q.56** Given Boolean equation as

$$y = (A, B, C, D)$$

$$y = \prod_M(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 15)$$

The minimized POS form of above equation is

- (a)  $(\bar{A})(C+D)(A+\bar{B}+\bar{C})$
- (b)  $(A+B)(A+\bar{D})(\bar{C}+\bar{D})(B+D)$
- (c)  $(A+B)(\bar{C}+\bar{D})(\bar{A}+B+C)$
- (d)  $(A)(\bar{C}+\bar{D})(B+C)$

2.2.8

- Q.57** To represent reduced expression of the following POS form, the no. of NOR gates required are  
 $y = \prod_M(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 15)$

- (a) 7
- (b) 6
- (c) 8
- (d) 9

- Q.58** The given expression can be reduced to

$$y = \overline{\overline{AB} + ABC + A(B + \overline{AB})}$$

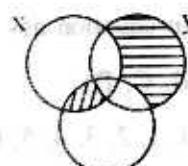
- (a) 1
- (b) A
- (c)  $\overline{A}$
- (d)  $\overline{AB}$

- Q.59** To realize minimized function of K-map shown below, no. of gates required are

|    | AB                       |
|----|--------------------------|
| CD | 1 1<br>1 1<br>1 1<br>1 1 |
| 00 | 1                        |
| 01 | 0 1                      |
| 11 | 0 1                      |
| 10 | 1 0                      |

- (a) 3
- (b) 1
- (c) 4
- (d) None of these

- Q.60** The Boolean expression for shaded area in the Venn diagram is



- (a)  $(\bar{x} + y) + \overline{xyz}$
- (b)  $\bar{x} + z + (\bar{x} + y + z)$
- (c)  $(\bar{x} + \bar{y}) + (\bar{x} + \bar{y} + z)$
- (d)  $(\bar{x} + z) + (\bar{x} + y + \bar{z})$

- Q.61** For the following K-map, POS form equation is

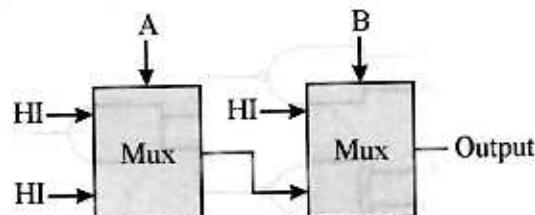
|    | AB | 00 | 01 | 11 | 10 |
|----|----|----|----|----|----|
| CD | 00 | 1  | 0  | 0  | 1  |
| 01 | 0  | 1  | 1  | 0  |    |
| 11 | 0  | 1  | 1  | 0  |    |
| 10 | 1  | 0  | 0  | 1  |    |

- (a)  $\prod_M(0, 2, 5, 7, 8, 10, 11, 15)$
- (b)  $\prod_M(1, 3, 4, 7, 8, 10, 11, 15)$
- (c)  $\prod_M(1, 3, 4, 6, 9, 11, 12, 14)$
- (d)  $\prod_M(0, 2, 4, 7, 9, 11, 12, 14)$

- Q.62** The single-precision floating point binary number of the binary no. 11111101110000000000000000000000 is

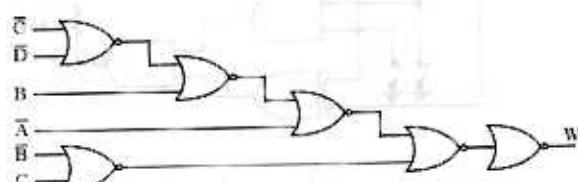
- (a) 1.10001001 11111101110000000000000000000000
- (b) 0.10110101 11111011100100000000000000000000
- (c) 0.10001101 11111011100000000000000000000000
- (d) 1.11001010 11011001110010000000000000000000

- Q.63** How is the operation dependent on A and B?



- (a) HI for A and B is HI
- (b) LO for A and B is HI
- (c) LO for A and B is LO
- (d) Independent of A and B

- Q.64** The following circuit represents:



- (a)  $W = A(B+CD)BC$
- (b)  $W = \overline{AB + \overline{C}} + DA$
- (c)  $W = AB + \overline{BC} + ACD$
- (d) None of these

- Q.65** Draw the K - map for the following equation  
 $f = f_1 + f_2$

$$f_1 = (A \oplus B)\bar{C}$$

$$f_2 = (A \odot B)C$$

(a)

|  |  | AB | 00 | 01 | 11 | 10 |
|--|--|----|----|----|----|----|
|  |  | C  | 0  | 1  |    | 1  |
|  |  | 0  | 1  |    |    |    |
|  |  | 1  |    |    | 1  |    |

(b)

|  |  | AB | 00 | 01 | 11 | 10 |
|--|--|----|----|----|----|----|
|  |  | C  | 0  | 1  |    | 1  |
|  |  | 0  | 1  |    |    |    |
|  |  | 1  |    | 1  |    | 1  |

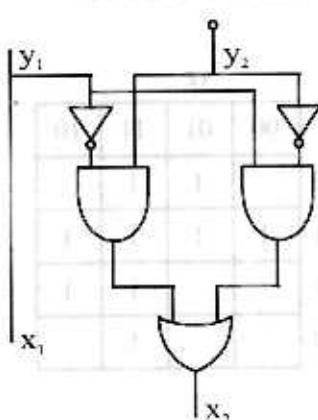
(c)

|  |  | AB | 00 | 01 | 11 | 10 |
|--|--|----|----|----|----|----|
|  |  | C  | 0  | 1  |    | 1  |
|  |  | 0  | 1  |    |    |    |
|  |  | 1  |    |    | 1  | 1  |

(d)

|  |  | AB | 00 | 01 | 11 | 10 |
|--|--|----|----|----|----|----|
|  |  | C  | 0  |    | 1  | 1  |
|  |  | 0  |    |    |    |    |
|  |  | 1  | 1  | 1  |    |    |

- Q.66** The logic circuit shown converts  $y_1-y_2$  into



(a) excess -3 code

(b) BCD

(c) error detecting code

(d) Gray code

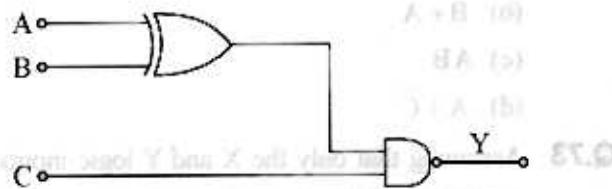
- Q.67** Consider a function that is defined by the following truth table:

| A | B | C | f(A, B, C) |
|---|---|---|------------|
| 0 | 0 | 0 | 1          |
| 1 | 0 | 0 | 1          |
| 0 | 1 | 0 | 1          |
| 1 | 1 | 0 | 1          |
| 0 | 0 | 1 | 0          |
| 1 | 0 | 1 | 0          |
| 0 | 1 | 1 | 0          |
| 1 | 1 | 1 | 0          |

Which of the following statements about the minimal sum-of-products and minimal product-of-sums implementations is correct?

- (a) They are logically not equivalent because the don't cares are used in different ways.
- (b) They are logically not equivalent by definition.
- (c) They are logically equivalent because the don't cares are used in the same way.
- (d) They are logically equivalent by definition.

- Q.68** The Boolean expression for the output of the logic circuit shown in the figure is



(a)  $Y = AB + AB + C$

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

(c)  $Y = \bar{A}\bar{B} + AB + \bar{C}$

(d)  $Y = AB + \bar{A}B + \bar{C}$

**Q.69** The implementation of  $(AB + B'C + AC)'$  is

- $A'B' + (BC' + A'C')$
- $(A' + B')(B + C')(A' + C')$
- $(A' + B')(B + C')(A' + C')$
- $(A' + B')(B + C')(A' + C')$

**Q.70** A seven bit Hamming code coming out of transmission line is 1000010. What was the code transmitted?

- 1 0 1 0 0 1 0
- 1 0 0 1 0 1 0
- 0 1 0 1 0 0 1
- 1 1 0 0 1 0 1

**Q.71** A combinational circuit has inputs A, B and C and its Karnaugh Map is as shown. The output of the circuit is given by

|  |  | A\B | 00 | 01 | 11 | 10 |
|--|--|-----|----|----|----|----|
|  |  | C   | 0  | 1  | 1  | 1  |
|  |  | 0   | 1  |    |    |    |
|  |  | 1   |    |    |    |    |

- $A \oplus B \oplus C$
- $(\bar{A}B + A\bar{B})\bar{C}$
- $\bar{A}\bar{B}\bar{C}$
- $(\bar{A}B + A\bar{B})C$

**Q.72** The boolean expression  $A[B + \bar{C}(AB + A\bar{C})]$  is equivalent to:

- $AC$
- $B + \bar{A}$
- $AB$
- $A + \bar{C}$

**Q.73** Assuming that only the X and Y logic inputs are available and their complements  $\bar{X}$  and  $\bar{Y}$  are not available, what is the minimum number of two-input NAND gate required to implement  $X \oplus Y$

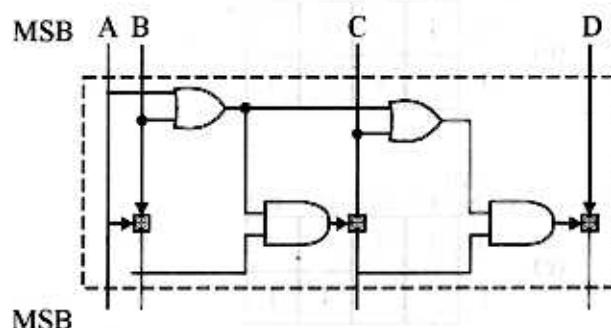
- 5
- 4
- 3
- 2

**Q.74** The complement of following Boolean function is:

$$(\bar{A} + \bar{A} + B)(\bar{B} + \bar{B} + C)$$

- $B + C$
- $B + \bar{C}$
- $\bar{A} + \bar{B}$
- $A + B$

**Q.75** The circuit shown in figure, converts



- BCD to binary code
- Gray to Binary code
- Excess to Gray code
- Binary to excess

**Q.76** The K-map for a Boolean function is shown in figure. The function would be

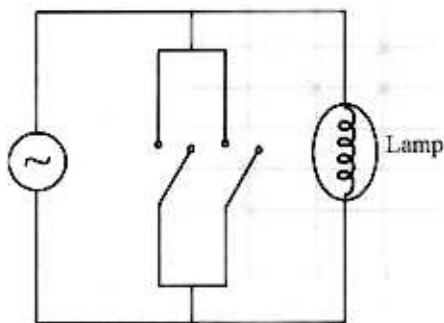
|  |  | yz |    |    |    |   |
|--|--|----|----|----|----|---|
|  |  | 00 | 01 | 11 | 10 |   |
|  |  | 00 |    | 1  | 1  |   |
|  |  | 01 | 1  | 1  | 1  | 1 |
|  |  | wx | 11 |    | 1  | 1 |
|  |  | 10 |    |    | 1  |   |

- $(x + \bar{z})(w + y)$
- $(\bar{x} + \bar{z})(w + \bar{y})$
- $(x + z)(\bar{w} + y)$
- none of these

**Q.77** The Boolean expression  $\bar{X}\bar{Y}\bar{Z} + \bar{X}\bar{Y}Z + XY\bar{Z} + X\bar{Y}Z + XYZ$  can be simplified to

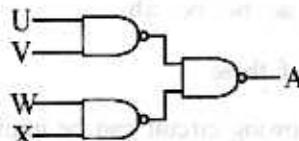
- (a)  $X\bar{Z} + \bar{X}Z + YZ$
- (b)  $\bar{X}Y + YZ + XZ$
- (c)  $XY + \bar{Y}Z + Y\bar{Z}$
- (d)  $\bar{X}Y + Y\bar{Z} + X\bar{Z}$

**Q.78** The switching diagram depicts



- (a) OR gate
- (b) NOR gate
- (c) AND gate
- (d) NAND gate

**Q.79** The value of A in the given circuit is



- (a)  $(\bar{U} + \bar{V})(\bar{W} + \bar{X})$
- (b)  $(U + V)(W + X)$
- (c)  $UV + WX$
- (d)  $\bar{U}V + WX$

**Q.80** The simplified form of the expression  $x = (\bar{A} + B)(A + B + D)\bar{D}$  is:

- (a)  $\bar{B}\bar{D}$
- (b)  $\overline{BD}$
- (c)  $\overline{BD}$
- (d)  $BD$

**Q.81** A logical function of three variable is given as:

$$f(A, B, C) = Y = (\bar{A} + \bar{B} + C)(A + C)(\bar{A} + \bar{B})$$

The canonical POS form is:

- (a)  $\Pi_M(1, 3, 9, 11)$
- (b)  $\Pi_M(0, 3, 5, 8)$
- (c)  $\Pi_M(0, 2, 6, 7)$
- (d)  $\Pi_M(1, 2, 8, 10)$

**Q.82** A logical function of three variable is given as:

$$f(A, B, C) = (A + BC)(B + \bar{C}A)$$

The canonical SOP form is:

- (a)  $\Sigma_m(2, 4, 8, 10)$
- (b)  $\Sigma_m(2, 4, 6, 7)$
- (c)  $\Sigma_m(3, 5, 8, 9)$
- (d)  $\Sigma_m(3, 4, 6, 7)$

## LEVEL-3

**Q.83** How many minimum NAND gates are required to implement following Boolean function?

$$y = \overline{(A + \bar{B}) \cdot (\bar{A} + C)}$$

- (a) 5
- (b) 6
- (c) 7
- (d) 8

**Q.84** The Boolean expression

$$A'BE + BCDE + BC'D'E + B'C'D'E'$$

can be simplified to  $BE + B'D'E'$ , if the don't care conditions are

- (a) ABCED + AB'CDE'
- (b) ABC'DE + AB'CDE' + ABCD'E
- (c) ABCD + AB'CDE' + ABCD'E
- (d) none of these

**Q.85** Using DeMorgan's theorem, convert the following Boolean expressions to equivalent expressions that have only OR and complement operation.

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

(a)  $(A + \bar{B}) + (\bar{A} + C) + (\bar{B} + C)$

(b)  $(\bar{A} + B) + (\bar{A} + \bar{C}) + (B + \bar{C})$

(c)  $AB + BC$

(d) None

**Q.86** Using De Morgan's theorem, convert the Boolean expression to equivalent expressions that have only AND and complement operations.

$$\bar{A} + \bar{B} + \bar{A}C + \bar{B}C$$

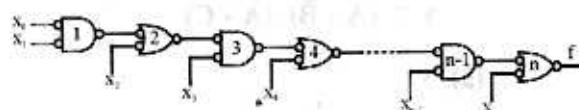
(a)  $(\bar{A}\bar{B})$

(b)  $\bar{A}\bar{B}\bar{A}C\bar{B}C$

(c)  $(A\bar{B})(\bar{A}B)(\bar{A}\bar{B})$

(d) None

**Q.87** The output 'y' of the circuit is:



(a)  $x_0x_2x_4\dots x_n + x_1x_2x_4x_6\dots x_n + x_3x_4x_6x_8\dots x_n + \dots + x_{n-1}x_n$

(b)  $x_0x_1x_2x_3\dots x_n + x_1x_2x_3\dots x_n + x_2x_3x_4\dots x_n + \dots + x_{n-1}x_n + x_n$

(c)  $x_0x_2x_4\dots x_n + x_1x_2x_4x_6\dots x_n + x_3x_4x_6\dots x_n + \dots + x_{n-1}x_n$

(d)  $x_0x_2x_4\dots x_n + x_1x_3x_5\dots x_{n-1} + x_3x_4\dots x_n + \dots + x_{n-2}x_n$

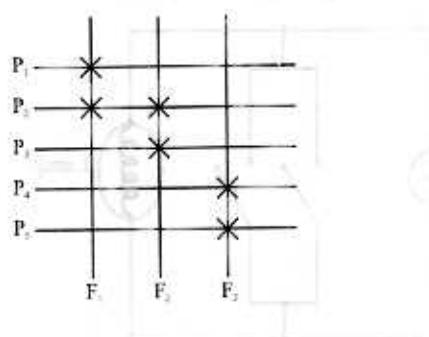
**Q.88** It is desired to generate the following three Boolean function

$$f_1 = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + ab\bar{c}$$

$$f_2 = \bar{a}\bar{b} + ab + \bar{a}bc$$

$$f_3 = \bar{a}\bar{b}\bar{c} + abc + \bar{a}c$$

by using OR gate array as shown in figure, where  $P_1$  and  $P_5$  are the product terms in one or more of the variable  $a, \bar{a}, b, \bar{b}, c$  and  $\bar{c}$ .



The terms of  $P_1, P_2, P_3, P_4$  and  $P_5$  are

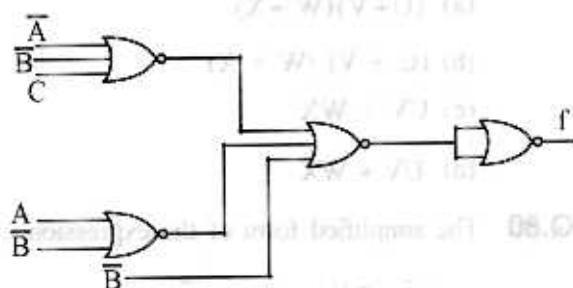
(a)  $ac, \bar{a}b, \bar{b}c, \bar{a}\bar{b}, bc$

(b)  $\bar{a}\bar{b}, \bar{b}c, ac, \bar{a}\bar{b}, bc$

(c)  $\bar{a}\bar{b}, ac, \bar{b}c, bc, \bar{a}\bar{b}$

(d) all of these

**Q.89** The following circuit can be implemented using:



(a) a single AND gate

(b) a NAND gate and a NOT gate

(c) an EXOR gate

(d) a NAND gate

**Q.90** The minimum number of NOR gates required to implement the function  $f = AB + \bar{A}\bar{B}$  are

- (a) 10
- (b) 12
- (c) 4
- (d) 6



**Q.91** The given expression can be reduced to

$$y = \overline{\overline{AB} + ABC} + A(B + \overline{AB})$$

- (a) A
- (b) 0
- (c)  $1 + A\bar{B} + B$
- (d)  $\overline{AB}$

**Q.92** A function with don't care condition is as follows:

$$f(w, x, y, z) = \sum_m(1, 5, 6, 12, 13, 14) + d(2, 4).$$

The minimize boolean expression is :

- (a)  $\overline{x}\overline{y} + x\overline{z} + \overline{w}\overline{y}z$
- (b)  $\overline{x}\overline{y} + \overline{x}\overline{y} + wz + \overline{x}y\overline{z}$
- (c)  $\overline{w}\overline{z} + \overline{x}z + wxy$
- (d)  $x\overline{z} + \overline{w}xy + \overline{x}wz$

**Q.93** The minimize form of the following Boolean expression using K-map is :

$$f(P, Q, R, S) = \sum_m(0, 4, 6, 7, 8, 9, 10, 13)$$

- (a)  $\overline{P}\overline{S}\overline{R} + \overline{P}\overline{Q}\overline{R} + \overline{P}RS + \overline{P}QR$
- (b)  $\overline{P}RS + \overline{P}\overline{Q}\overline{S} + \overline{P}RS + \overline{P}QR$
- (c)  $P\overline{Q}\overline{R} + PS\overline{Q} + \overline{R}SQ + \overline{P}\overline{Q}\overline{R}$
- (d)  $\overline{P}\overline{Q}\overline{R} + PQ + \overline{P}RS + \overline{Q}RS$

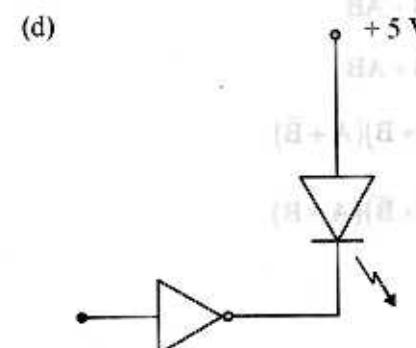
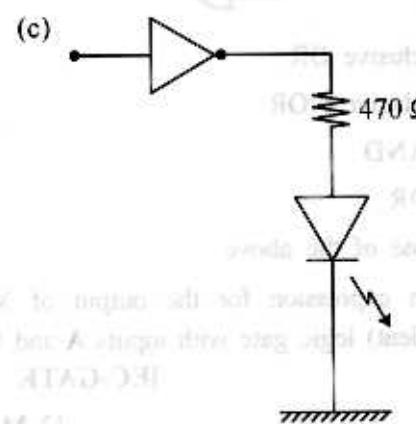
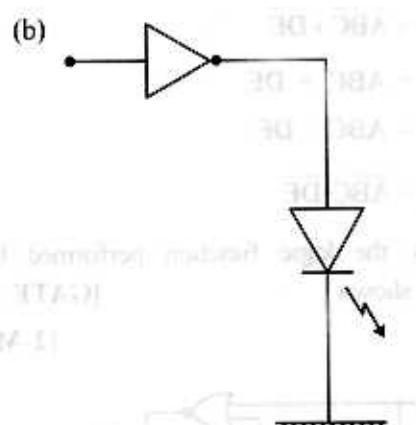
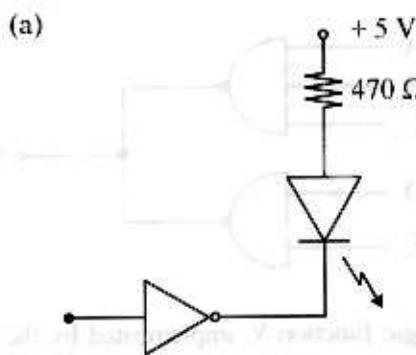
## GATE QUESTIONS

**Q.94** The total number of Boolean functions which can be realized with four variables is

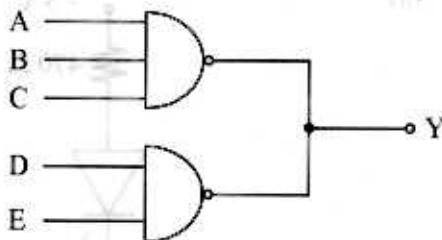
[GATE 1987]

- (a) 4
- (b) 17
- (c) 256
- (d) 65,536

**Q.95** The most satisfactory LED driver circuit using a TTL gate is : [GATE 1987]



- Q.96** Two NAND gates having open collector outputs are tied together as shown.



The logic function  $Y$ , implemented by the circuit is,

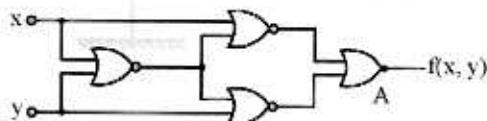
[GATE 1990]

- (a)  $Y = \overline{ABC} + \overline{DE}$
- (b)  $Y = ABC + DE$
- (c)  $Y = ABC \cdot DE$
- (d)  $Y = \overline{ABC} \cdot \overline{DE}$

- Q.97** Identify the logic function performed by the circuit shown

[GATE 1993]

[2-Marks]



- (a) exclusive OR
- (b) exclusive NOR
- (c) NAND
- (d) NOR
- (e) None of the above

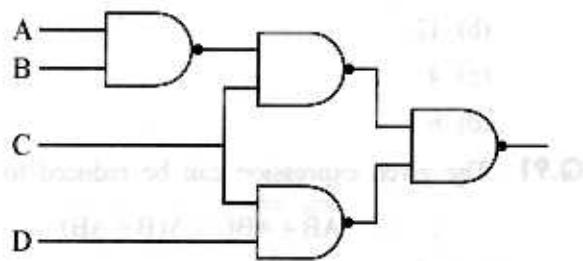
- Q.98** Boolean expression for the output of XNOR (equivalent) logic gate with inputs  $A$  and  $B$  is

[EC-GATE 1993]

[2-Marks]

- (a)  $A\bar{B} + \bar{A}B$
- (b)  $\overline{AB} + AB$
- (c)  $(\bar{A} + B)(A + \bar{B})$
- (d)  $(\bar{A} + \bar{B})(A + B)$

- Q.99** The logic expression for the output of the circuit shown in the figure is [GATE 1994]



- (a)  $\overline{AC} + \overline{BC} + CD$
- (b)  $A\bar{C} + B\bar{C} + \bar{C}\bar{D}$
- (c)  $ABC = \overline{CD}$
- (d)  $\overline{AB} + \overline{BC} + \overline{CD}$

- Q.100** The output of logic gate is '1' when all its inputs are at logic '0'. Then gate is either

[EC-GATE 1994]

- (a) a NAND or an EX-OR gate
- (b) a NOR or an EX-NOR gate
- (c) an OR or an EX NOR gate
- (d) an AND or an EX-OR gate.

- Q.101** What values of  $A$ ,  $B$ ,  $C$  and  $D$  satisfy the following simultaneous Boolean equations?

$$\overline{A} + AB = 0, AB = AC, AB + A\bar{C} + CD = \overline{CD}$$

[GATE 1995]

[2-Marks]

- (a)  $A = 1, B = 0, C = 0, D = 1$
- (b)  $A = 1, B = 1, C = 0, D = 0$
- (c)  $A = 1, B = 0, C = 1, D = 1$
- (d)  $A = 1, B = 0, C = 0, D = 0$

- Q.102** The minimum number of NAND gates required to implement the Boolean function

$$A + A\bar{B} + A\bar{B}C$$

, is equal to [EC-GATE 1995]

- (a) zero
- (b) 1
- (c) 4
- (d) 7

**Q.103** Consider the logic circuit shown in the figure below. The function  $f_1$ ,  $f_2$  and  $f$  (in canonical sum of products form in decimal notation) are:

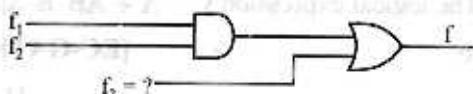
[GATE 1997]

[2-Marks]

$$f_1(w, x, y, z) = \Sigma 8, 9, 10$$

$$f_2(w, x, y, z) = \Sigma 7, 8, 12, 13, 14, 15$$

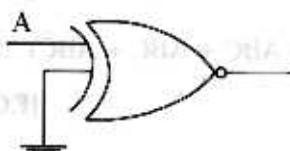
$$f(w, x, y, z) = \Sigma 8, 9,$$



The function  $f_3$  is

- (a)  $\Sigma 9, 10$
- (b)  $\Sigma 9$
- (c)  $\Sigma 1, 8, 9$
- (d)  $\Sigma 8, 10, 15$

**Q.104** The output of the logic gate in the figure given is



[EC-GATE 1997]

[1-Marks]

- (a) 0
- (b) 1
- (c) A
- (d)  $\bar{A}$

**Q.105** The boolean function  $A+BC$  is a reduced form of

[EC-GATE 1997]

- (a)  $AB + BC$
- (b)  $(A+B)(A+C)$
- (c)  $\bar{A}B + \bar{A}C$
- (d)  $(A+C)B$

**Q.106** Which of the following operations is commutative but not associative? [GATE 1998]

[2-Marks]

- (a) AND
- (b) OR
- (c) NAND
- (d) EXOR

**Q.107** The function represented by the Karnaugh map given below is [GATE 1998]

[2-Marks]

|  |  | BC | 00 | 01 | 10 | 11 |   |
|--|--|----|----|----|----|----|---|
|  |  | A  | 0  | 1  | 0  | 0  | 1 |
|  |  | 1  | 1  | 0  | 0  | 1  |   |
|  |  |    |    |    |    |    |   |

- (a)  $A \cdot B$
- (b)  $AB + BC + CD$
- (c)  $\overline{B} \oplus C$
- (d)  $A \cdot BC$

**Q.108** What happens when a bit-string is XORed with itself n-times as shown:

[ $B \oplus (B \oplus (B \oplus (B \dots n \text{ times}))$ ] [GATE 1998]

[1-Mark]

- (a) complements when n is even
- (b) complements when n is odd
- (c) divides by  $2^n$  always
- (d) remains unchanged when n is even

**Q.109** The K-map for a boolean function is shown in figure. The number of essential prime implicants for this function is

|  |  | AB | 00 | 01 | 11 | 10 |   |
|--|--|----|----|----|----|----|---|
|  |  | CD | 00 | 1  | 1  | 0  | 1 |
|  |  | 01 | 0  | 0  | 0  | 1  |   |
|  |  | 11 | 1  | 0  | 0  | 0  |   |
|  |  | 10 | 1  | 0  | 0  | 1  |   |

[EC-GATE 1998]

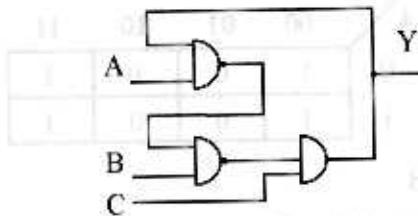
- (a) 4
- (b) 5
- (c) 6
- (d) 8

**Q.110** The minimum number of 2 input NAND gates required to implement the boolean function  $Z = A\bar{B}C$ , assuming that A, B and C are available, is

[EC-GATE 1998]

- (a) two
- (b) three
- (c) five
- (d) six

**Q.111** Consider the circuit shown below. In a certain steady state, the line Y is at '1'. What are the possible values of A, B and C in this state?



[GATE 1999]

[2-Marks]

- (a) A = 0, B = 0, C = 1
- (b) A = 0, B = 1, C = 1
- (c) A = 1, B = 0, C = 1
- (d) A = 2, B = 1, C = 1

**Q.112** Which of the following sets of component(s) is/are sufficient to implement any arbitrary Boolean function?

[GATE 1999]

[2-Marks]

- (a) XOR gates, NOT gates
- (b) 2 to 1 multiplexers
- (c) AND gates, XOR gates
- (d) Three input gates the output  $(A \cdot B) + C$  for the inputs A, B and C.

**Q.113** Which of the following function implements the Karnaugh map shown below? [GATE 1999]

[1-Mark]

|  |    | CD | 00 | 01 | 11 | 10 |
|--|----|----|----|----|----|----|
|  |    | AB | 00 | 01 | 11 | 10 |
|  | 00 |    | 0  | 0  | 1  | 0  |
|  | 01 |    | X  | X  | 1  | X  |
|  | 11 |    | 0  | 1  | 1  | 0  |
|  | 10 |    | 0  | 1  | 1  | 0  |

- (a)  $\bar{A}B + CD$
- (b)  $D(C + A)$
- (c)  $AD + \bar{A}B$
- (d)  $(C+D)(\bar{C}+D)+(A+B)$

**Q.114** Which of the following expression is not equivalent to  $\bar{x}$ ? [GATE 1999]

[1-Mark]

- (a)  $x \text{ NAND } x$
- (b)  $x \text{ NOR } x$
- (c)  $x \text{ NAND } 1$
- (d)  $x \text{ NOR } 1$

**Q.115** The logical expression  $y = A + \bar{A}B$  is equivalent to [EC-GATE 1999]

[1-Mark]

- (a)  $y = AB$
- (b)  $y = A + B$
- (c)  $y = \bar{A} + B$
- (d)  $y = \bar{A}\bar{B}$

**Q.116** The minimized form of the logical expression

$(\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C})$  is

[EC-GATE 1999]

[2-Marks]

- (a)  $\bar{A}\bar{C} + \bar{B}\bar{C} + \bar{A}\bar{B}$
- (b)  $\bar{A}\bar{C} + \bar{B}\bar{C} + \bar{A}\bar{B}$
- (c)  $\bar{A}\bar{C} + \bar{B}\bar{C} + \bar{A}\bar{B}$
- (d)  $A\bar{C} + \bar{B}\bar{C} + A\bar{B}$

**Q.117** Which function does NOT implement the Karnaugh map given below? [GATE 2000]

[2-Marks]

|      | Wz → | 00 | 01 | 11 | 10 |
|------|------|----|----|----|----|
| Xy ↓ |      |    |    |    |    |
| 00   | 0    | x  | 0  | 0  |    |
| 01   | 0    | x  | 1  | 1  |    |
| 11   | 1    | 1  | 1  | 1  |    |
| 10   | 0    | x  | 0  | 0  |    |

- (a)  $(w+x)y$
- (b)  $xy + yw$
- (c)  $(w+x)(\bar{w}+y)(\bar{x}+y)$
- (d) None of the above

**Q.118** Minimum sum of product expression of  $f(w, x, y, z)$  shown in Karnaugh-map below is

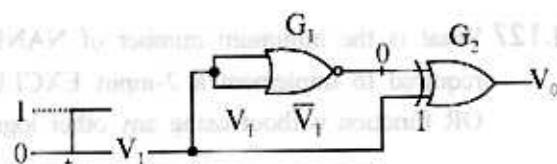
[GATE 2002]

[1-Mark]

| YZ \ WX | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00      | 0  | 1  | 1  | 0  |
| 01      | X  | 0  | 0  | 1  |
| 11      | X  | 0  | 0  | 1  |
| 10      | 0  | 1  | 1  | X  |

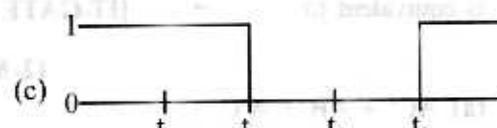
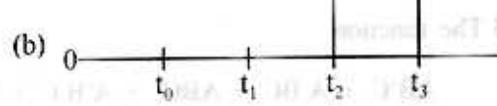
- (a)  $xz + y'z$
- (b)  $xz' + zx'$
- (c)  $x'y + zx'$
- (d) None of above

**Q.119** The gates  $G_1$  and  $G_2$  in the figure have propagation delays of 10 nsec and 20 nsec respectively. If the input  $V_1$  makes an abrupt change from logic 0 to 1 at time  $t = t_0$ , then the output waveform  $V_0$  is

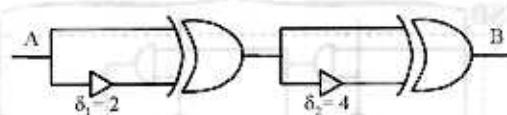


[IIT-GATE 2002]

[2-Marks]



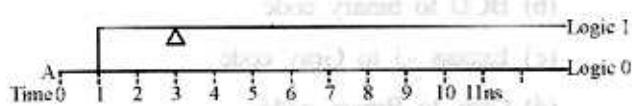
**Q.120** Consider the following circuit composed of XOR gates and non-inverting buffers.



The non-inverting buffers have delays  $\delta_1 = 2$  and  $\delta_2 = 4$  ns as shown in the figure. Both XOR gates and all wires have zero delay. Assume that all gate inputs, outputs and wires are stable at logic level 0 at time 0. If the following waveform is applied at input A, how many transition(s) (change of logic levels) occur(s) at B during the interval from 0 to 10ns

[GATE 2003]

[2-Marks]



- (a) 1
- (b) 2
- (c) 3
- (d) 4

**Q.121** The literal count of a Boolean expression is the sum of the number of times each literal appears in the expression. For example, the literal count of  $(xy + xz)$  is 4. What are the minimum possible literal counts of the product-of-sum and sum-of-product representations respectively of the function given by the following Karnaugh map ? Here, X denotes "don't care".

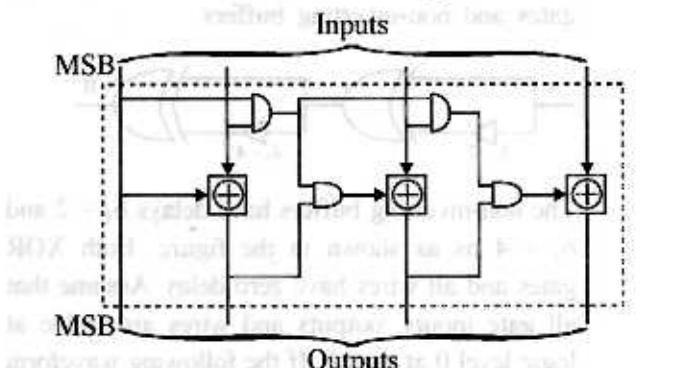
| ZW \ XY | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00      | X  | 1  | 0  | 1  |
| 01      | 0  | 1  | X  | 0  |
| 11      | 1  | X  | X  | 0  |
| 10      | X  | 0  | 0  | X  |

[GATE 2003]

[2-Marks]

- (a) (11, 9)
- (b) (9, 13)
- (c) (9, 10)
- (d) (11, 11)

**Q.122** The circuit shown in the figure converts



[EC-GATE 2003]

[2-Marks]

- (a) Binary to excess -3 code
- (b) BCD to binary code
- (c) Excess -3 to Gray code
- (d) Gray to Binary code

**Q.123** If the functions  $W_1X_1Y$  and  $Z$  are as follows

$$W = R + \bar{P}Q + \bar{R}S$$

$$X = PQ\bar{S} + \bar{P}Q\bar{S} + P\bar{Q}\bar{S}$$

$$Y = RS + PR + P\bar{Q} + \bar{P}Q$$

$$Z = R + S + PQ + \bar{P}Q\bar{R} + P\bar{Q}\bar{S}$$

Then, [EC-GATE 2003]

[2-Marks]

- (a)  $W = Z$ ,  $X = Y$
- (b)  $W = Z$ ,  $X = \bar{Z}$
- (c)  $W = Y$
- (d)  $W = Y = \bar{Z}$

**Q.124** A Boolean function  $x'y' + xy + x'y$  is equivalent to:

[GATE 2004]

[1-Mark]

- (a)  $x' + y'$
- (b)  $x + y$
- (c)  $x + y'$
- (d)  $x' + y$

**Q.125** Which are the essential prime implicants of the following Boolean function?

$$f(a, b, c) = a'c + ac' + b'c \quad [\text{GATE 2004}]$$

[2-Marks]

- (a)  $a'c$  and  $ac'$
- (b)  $a'c$  and  $b'c$
- (c)  $a'$  only
- (d)  $ac'$  and  $bc'$

**Q.126** The Boolean expression  $AC + B\bar{C}$  is equivalent to

[EC-GATE 2004]

[2-Marks]

- (a)  $ABC + \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}BC$
- (b)  $\bar{B}C + AC + B\bar{C} + \bar{A}CB$
- (c)  $AC + B\bar{C} + \bar{B}C + ABC$
- (d)  $\bar{A}C + B\bar{C} + AC$

**Q.127** What is the minimum number of NAND gates required to implement a 2-input EXCLUSIVE-OR function without using any other logic gate?

[IT-GATE 2004]

[1 Mark]

- (a) 3
- (b) 4
- (c) 5
- (d) 6

**Q.128** The function

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

is equivalent to

[IT-GATE 2004]

[2-Marks]

- (a)  $AC' + AB + A'C$
- (b)  $AB' + AC' + A'C$
- (c)  $A'B + AC' + AB'$
- (d)  $A'B + AC + AB'$

**Q.129** The Boolean expression for the truth table shown is

| A | B | C | f |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

[EC-GATE 2005]

[2-Marks]

- (a)  $\bar{B}(A + C)(\bar{A} + \bar{C})$
- (b)  $B(A + \bar{C})(\bar{A} + C)$
- (c)  $\bar{B}(A + \bar{C})(\bar{A} + C)$
- (d)  $B(A + C)(\bar{A} + \bar{C})$

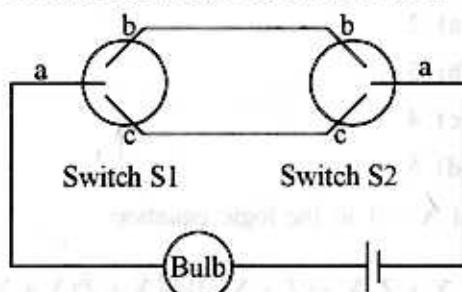
**Q.130** Which of the following expressions is equivalent to  $(A \oplus B) \oplus C$

[IIT-GATE 2005]

[1 Mark]

- (a)  $(A + B + C)(\bar{A} + \bar{B} + \bar{C})$
- (b)  $(A + B + C)(\bar{A} + \bar{B} + C)$
- (c)  $ABC + \bar{A}(B \oplus C) + \bar{B}(A \oplus C)$
- (d) None of the above

**Q.131** A two-way switch has three terminals a, b and c. In ON position (logic value 1) a is connected to b, and in OFF position, a is connected to c. Two of these two way switches S1 and S2 are connected to a bulb as shown below.



Which of the following expressions, if true, will always result in the lighting of the bulb?

[IIT-GATE 2005]

[1 Mark]

- (a)  $S_1 \bar{S}_2$
- (b)  $S_1 + S_2$
- (c)  $\overline{S_1 \oplus S_2}$
- (d)  $S_1 \oplus S_2$

**Q.132** Given two three bit numbers  $a_2a_1a_0$  and  $b_2b_1b_0$  and c, the carry in, the function that represents the carry generate function when these two numbers are added is: [GATE 2006]

[2-Marks]

- (a)  $a_2b_2 + a_2a_1b_1 + a_2a_1a_0b_0 + a_2a_0b_1b_0 + a_1b_2b_1 + a_1a_0b_2b_0 + a_0b_2b_1b_0$
- (b)  $a_2b_2 + a_2b_1b_0 + a_2a_1b_1b_0 + a_1a_0b_2b_1 + a_1a_0b_2 + a_1a_0b_2b_0 + a_2a_0b_1b_0$
- (c)  $a_2 + b_2 + (a_2 \oplus b_2)[a_1 + b_1 + (a_1 \oplus b_1)(a_0 + b_0)]$
- (d)  $a_2b_2 + \bar{a}_2a_1b_1 + \bar{a}_2\bar{a}_1a_0b_0 + \bar{a}_2a_0\bar{b}_1b_0 + a_1\bar{b}_2b_1 + \bar{a}_1a_0\bar{b}_2b_0 + a_0\bar{b}_2\bar{b}_1b_0$

**Q.133** The number of product terms in the minimized sum-of-product expression obtained through the following K-map is (where, "d" denotes don't care states)

|   |   |   |   |
|---|---|---|---|
| 1 | 0 | 0 | 1 |
| 0 | d | 0 | 0 |
| 0 | 0 | d | 1 |
| 1 | 0 | 0 | 1 |

[EC-GATE 2006]

[1-Mark]

- (a) 5
- (b) 4
- (c) 3
- (d) 2

**Q.134** Consider the following Boolean function of four variables:

$$f(w, x, y, z) = \Sigma(1, 3, 4, 6, 9, 11, 12, 14)$$

The function is:

[GATE 2007]

[1-Mark]

- (a) independent of one variables.
- (b) independent of two variables.
- (c) independent of three variables.
- (d) dependent on all the variables.

**Q.135** Define the connective \* for the Boolean variables X and Y as:  $X * Y = XY + X'Y'$ .

Let  $Z = X * Y$ . Consider the following expressions P, Q and R.

$$\pi P : X = Y * Z \quad Q : Y = X * Z$$

$$R : X * Y * Z = 1$$

Which of the following is TRUE?

[GATE 2007]

[2-Marks]

- (a) Only P and Q are valid.
- (b) Only Q and R are valid.
- (c) Only P and R are valid.
- (d) All P, Q, R are valid.

**Q.136** What is the maximum number of different Boolean functions involving n Boolean variables?

[GATE 2007]

[1-Mark]

- (a)  $n^2$
- (b)  $2^n$
- (c)  $2^{2n}$
- (d)  $2^{n^2}$

|   |   |   |   |
|---|---|---|---|
| 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |

**Q.137** Let  $f(w, x, y, z) = \Sigma(0, 4, 5, 7, 8, 9, 13, 15)$ . Which of the following expressions are NOT equivalent to f?

[GATE 2007]

[2-Marks]

- (a)  $x'y'z' + w'xy' + wy'z + xz$
- (b)  $w'y'z' + wx'y' + xz$
- (c)  $w'y'z' + wx'y' + xyz + xy'z$
- (d)  $x'y'z' + wx'y' + w'y$

**Q.138** The Boolean function  $Y = AB + CD$  is to be realized using only 2-input NAND gates. The minimum number of gates required is

[EC-GATE 2007]

[1-Mark]

- (a) 5
- (b) 4
- (c) 3
- (d) 2

**Q.139** In the Karnaugh map shown below, X denotes a don't care term. What is the minimal form of the function represented by the Karnaugh map?

|  |  | ab | 00 | 01 | 11 | 10 |
|--|--|----|----|----|----|----|
|  |  | cd | 00 | 1  | 1  |    |
|  |  | 01 | x  |    |    |    |
|  |  | 11 | x  |    |    |    |
|  |  | 10 | 1  | 1  |    | x  |

[GATE 2008]

[1-Mark]

- (a)  $\bar{b}\bar{d} + \bar{a}\bar{d}$
- (b)  $\bar{a}\bar{b} + \bar{b}\bar{d} + \bar{a}b\bar{d}$
- (c)  $\bar{b}\bar{d} + \bar{a}b\bar{d}$
- (d)  $\bar{a}\bar{b} + \bar{b}\bar{d} + \bar{a}\bar{d}$

**Q.140** What is the minimum number of gates required to implement the Boolean function  $(AB + C)$  if we have to use only 2-input NOR gates?

[GATE 2009]

[1-Mark]

- (a) 2
- (b) 3
- (c) 4
- (d) 5

**Q.141** If  $X = 1$  in the logic equation

$$[X + Z\{\bar{Y} + (\bar{Z} + X\bar{Y})\}] \{ \bar{X} + \bar{Z}(X + Y) \} = 1,$$

then

[EC-GATE 2009]

[2-Marks]

- (a)  $Y = Z$
- (b)  $Y = \bar{Z}$
- (c)  $Z = 0$
- (d)  $Z = 1$

## ANSWER KEY

# SOLUTIONS

**S.1 (a)**



When  $B = 1$ , the gate is inhibited.

When  $B = 0$ , the gate is enabled.

**S.2 (b)**

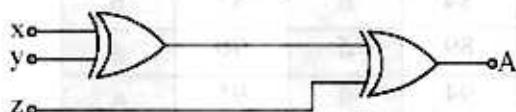
$$\begin{aligned} & [(A + \bar{A}B)(A + \bar{A}\bar{B})][(CD + \bar{C}\bar{D})] + (C \oplus D)] \\ &= [(A + B)(A + \bar{B})][(CD + \bar{C}\bar{D} + \bar{C}D + CD)] \\ &= (AA + A\bar{B} + AB + B\bar{B})(I) \\ &= (A + A + 0) \\ &= A \end{aligned}$$

$$\left\{ \begin{array}{l} X + \bar{X} = 1 \\ A + \bar{A}B = A + B \\ B\bar{B} = 0 \\ AA = A \end{array} \right.$$

**S.3 (b)**

The expression is also written as

$$A = x \oplus y \oplus z$$



**S.19 (d)**

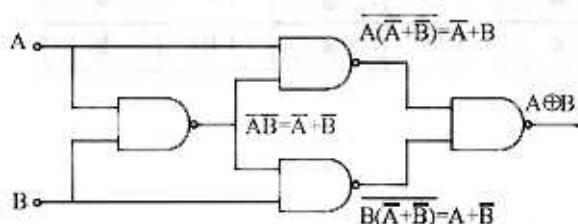
$$Z = A(A + \bar{A})BC = ABC$$

**S.20 (c)**

$$Z = (\bar{A} + \bar{B}).BC = (AB).BC = ABC$$

**S.21 (d)**

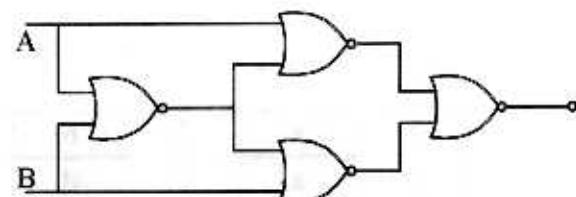
XOR gate using NAND gates only.



**S.24 (d)**

In the dual form we are changing + sign to  $\times$  and vice-versa.

**S.25 (a)**



**S.28 (d)**

$A + A'B = A + B$  which is the area common between B and complement of A.

**S.29 (a)**

AND gate is called series circuit gate.

**S.30 (c)**

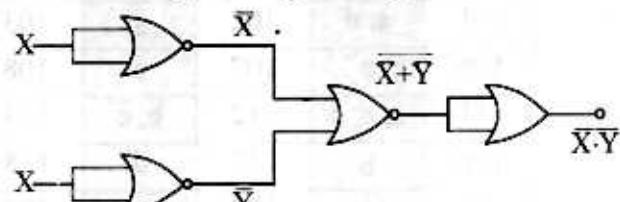
The OR-AND realization is best realized by NOR gates.

**S.31 (c)**

|    |    | RS |    |    |    |
|----|----|----|----|----|----|
|    |    | 00 | 01 | 11 | 10 |
| PQ | 00 | 1  | 0  | 1  | 1  |
|    | 01 | 0  | 1  | 1  | 0  |
|    | 11 | 0  | 1  | 0  | 0  |
|    | 10 | 1  | 0  | 0  | 1  |

**S.32 (b)**

NAND gate using NOR gate



**S.33 (a)**

$$\begin{aligned} (A + \bar{B})(\bar{A} + B) &= (\bar{A} + B) + (\bar{A} + B) \\ &= (\bar{A} \cdot B) + \bar{A} + B \\ &= \bar{A}(B + I) + B \\ &= \bar{A} + B \end{aligned}$$

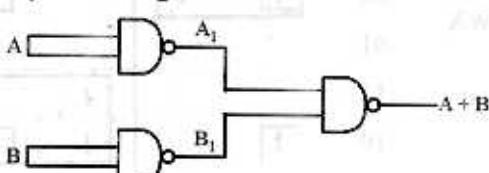
**S.34 (c)**

(NOR).(XOR).(NAND)

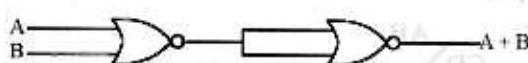
$$\begin{aligned}
 &= \text{NOR} \oplus \text{NAND} \\
 &= \text{XOR} \\
 y &= \text{NOR} \cdot \text{XOR} \cdot \text{NAND} \\
 y &= \overline{A+B} \oplus \overline{AB} \\
 &= \overline{A} \cdot \overline{B} \oplus \overline{AB} \\
 &= \overline{A} \cdot \overline{B} \overline{AB} + \overline{AB} \overline{A} \cdot \overline{B} \\
 &= (A+B)(\overline{AB}) + AB\overline{A}\overline{B} \\
 &= (A+B)(\overline{A}+\overline{B}) + 0 \\
 &= \overline{A}A + A\overline{B} + B\overline{B} + A\overline{B} \\
 &= \overline{A}B + A\overline{B} \\
 &= A \oplus B
 \end{aligned}$$

**S.38 (c)**

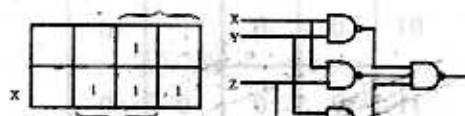
By NAND gate



By NOR gate

**S.39 (a)**

$$F = XY + YZ + XZ$$

**S.40 (a)**

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

$$\begin{aligned}
 F &= B\bar{C}D + \bar{B}\bar{C}D + \bar{A}BC + A\bar{B}C + ABC \\
 &= XB\bar{C}D + X\bar{B}\bar{C}D + \bar{A}BCX + A\bar{B}CX \\
 &\quad + ABCX \\
 &= A\bar{B}\bar{C}D + \bar{A}B\bar{C}D + A\bar{B}CD + \bar{A}\bar{B}CD \\
 &\quad + \bar{A}BCD + \bar{A}BC\bar{D} + A\bar{B}CD + A\bar{B}C\bar{D} \\
 &\quad + ABCD + ABC\bar{D}
 \end{aligned}$$

Values  $\rightarrow 1, 3, 5, 9, 1, 7, 6, 11, 10, 15, 14$ 

$$= \Sigma (1, 5, 6, 7, 9, 10, 11, 13, 14, 15)$$

**S.43 (a)**

In the given circuit

$$\begin{aligned}
 x_0 &= Y \oplus A_0 \\
 x_1 &= Y \oplus A_1 \\
 x_2 &= Y \oplus A_2
 \end{aligned}$$

When  $y = 1$ 

$$x_0 = 1 \oplus A_0$$

Let  $A_0 = 0$ 

$$\therefore x_0 = 1 \oplus 0 = 1$$

 $\therefore x_0 = \overline{A_0}$  (1's complement of  $A_0$ )Let  $A_0 = 1$ 

$$\therefore x_0 = 1 \oplus 1 = 0$$

 $\therefore x_0 = \overline{A_0}$  (1's complement of  $A_0$ ) $\therefore$  when  $y = 1$   $x_2 x_1 x_0$  will be 1's complement of  $A_2 A_1 A_0$ .**S.44 (b)**

In terms of Boolean operation,

Output of 1 is  $x_0 x_1$ Output of 2 is  $(x_0 x_1 + x_2)$ Output of 3 is  $(x_0 x_1 + x_2)x_3 = x_0 x_1 x_3 + x_2 x_3$ Output of 4 is  $x_0 x_1 x_3 + x_2 x_3 + x_4$ Output of 5 would be  $x_0 x_1 x_3 x_5 + x_2 x_3 x_5 + x_4 x_5$ Output of 6 would be  $x_0 x_1 x_3 x_5 + x_2 x_3 x_5 + x_4 x_5 x_6$ Thus for  $n$  gates connected as shown, the output would be

$$x_0 x_1 x_3 \dots x_{n-1}$$

$$+ x_2 x_3 x_5 \dots x_{n-1}$$

$$+ x_4 x_5 x_7 \dots x_n$$

$$+ \dots$$

$$+ x_{n-2} x_{n-1}$$

$$+ x_n$$

## S.45 (a)

$$\begin{aligned}
 X &= (\overline{MNQ})(\overline{M\bar{N}Q})(\overline{M\bar{N}\bar{Q}}) \\
 &= (\bar{M} + \bar{N} + \bar{Q})(\bar{M} + N + \bar{Q})(\bar{M} + \bar{N} + \bar{Q}) \\
 &= (\bar{M} + \bar{N} + \bar{Q}) + (\bar{M} + N + \bar{Q}) + (\bar{M} + \bar{N} + \bar{Q}) \\
 &= MNQ + M\bar{N}Q + \bar{M}NQ \\
 &= MQ(N + \bar{N}) + \bar{M}NQ \\
 &= MQ + \bar{M}NQ \\
 &= Q(M + \bar{M}N) \\
 &= Q(M + N)
 \end{aligned}$$

## S.46 (d)

$$Z = \bar{A} + (\bar{AB} + \bar{BC}) + \bar{C}$$

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

$$\overline{ABC} = \bar{A} + \bar{B} + \bar{C}$$

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

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

## S.47 (c)

Using duality

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

## S.48 (b)

$$X = (A\bar{B} + \bar{A}B)(A + \bar{B}) = (A\bar{B} + \bar{A}B)(\bar{A}B) = \bar{A}B$$

## S.49 (c)

$$\bar{X} = \bar{A}\bar{B}C + A\bar{B}C + AB\bar{C} = \bar{B}C + AB\bar{C}$$

## S.50 (b)

$$\begin{aligned}
 F &= A + \bar{A}B + \bar{A}BC + \bar{ABC}(D + \bar{D}E) \\
 &= A + \bar{A}B + \bar{A}\bar{B}(C + \bar{C}(D + E)) \\
 &= A + \bar{A}(B + \bar{B}(C + D + E)) = A + B + C + D + E
 \end{aligned}$$

## S.51 (a)

X will be HIGH when  $A \neq B$  and  $C = 1$ , thus  $C = 1, B = 1, A = 0$  is the input condition.

## S.52 (b)

$$X = (AB)(CD + \bar{CD})$$

## S.53 (c)

Output of NAND must be LOW for LED to emit light. So both input to NAND must be HIGH. If any one or both switch are closed, output of AND will be LOW. If both switch are open, output of XOR will be LOW. So there can't be both input HIGH to NAND. So LED doesn't emit light.

## S.54 (c)

$$\text{Output of 1st XOR} = \bar{X}I + X\bar{I} = \bar{X}$$

$$\text{Output of 2nd XOR} = \bar{XX} + XX = 1$$

## S.55 (c)

$$f = \overline{WXY} + \overline{XZ} + WY$$

|    |    | YZ |    |    |
|----|----|----|----|----|
|    |    | 00 | 01 | 11 |
| WX | 00 | 1  | 1  | 1  |
|    | 01 |    |    |    |
| WX | 11 |    | 1  | 1  |
|    | 10 | 1  |    | 1  |

## S.56 (d)

| CD | AB |    |   |   |
|----|----|----|---|---|
|    | 00 | 01 |   |   |
| 00 | 0  | 0  |   | 0 |
| 01 | 0  | 0  |   | 0 |
| 11 | 0  | 0  | 0 | 0 |
| 10 | 0  | 0  |   |   |

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

for POS form

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

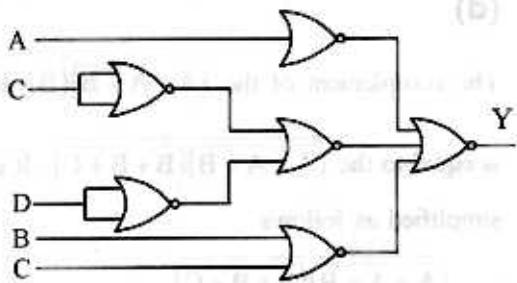
$$= A(\bar{C} + \bar{D})(B + C)$$

## S.57 (b)

To implement

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

Using NOR gate, we require 6 NOT gates.

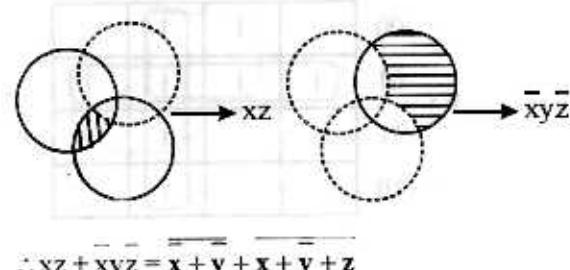
**S.58 (c)**

$$\begin{aligned}
 Y &= \overline{\overline{AB} + ABC + A(B + AB)} \\
 &= \overline{\overline{AB} + ABC + A(B + A)(B + \bar{B})} \\
 &= \overline{\overline{AB} + ABC + A(B + A)} \\
 &= \overline{\overline{AB} + ABC + AB + A} \\
 &= \overline{\overline{AB} + ABC + A(I + B)} \\
 &= \overline{\overline{AB} + ABC + A} \\
 &= \overline{\overline{AB} + ABC\bar{A}} \\
 &= (\overline{AB} + ABC)\bar{A} \\
 &= (\overline{A} + \overline{B} + ABC)\bar{A} \\
 &= \overline{A} + \overline{AB} + A\overline{ABC} \\
 &= \overline{A}(I + B)
 \end{aligned}$$

$$Y = \overline{A}$$

**S.59 (a)**The minimized solution is  $y = A \oplus B \oplus C \oplus D$ 

No. of gates required : 3 XNOR gates.

**S.60 (c)****S.64 (c)**

$$\begin{aligned}
 W &= \overline{\overline{\overline{C} + \overline{D} + B + \overline{A} + \overline{B} + C}} \\
 W &= \overline{\overline{\overline{C} + \overline{D} + B + \overline{A} + \overline{B} + C}} \\
 &= \overline{\overline{C}\overline{D} + B + \overline{A} + \overline{B} + C} \\
 &= \overline{\overline{C}\overline{D}\cdot B + \overline{A}\cdot B\cdot C} \\
 &= \overline{\overline{C}\overline{D}\cdot B\cdot A + B\cdot C} \\
 &= (CD + B)\cdot A + B\bar{C}
 \end{aligned}$$

$$W = ACD + AB + B\bar{C}$$

$$W = AB + B\bar{C} + ACD$$

**S.68 (c)**

$$\begin{aligned}
 (A \oplus B)\bar{C} &= (\overline{A \oplus B}) + \bar{C} \\
 &= \overline{\overline{AB} \cdot \overline{A\bar{B}}} + \bar{C} \quad [\because \overline{A\cdot B} = \bar{A} + \bar{B}] \\
 &= (\overline{A} + \bar{B}) \cdot (\bar{A} + B) + \bar{C} \\
 &= \overline{\overline{AB} + AB + \bar{C}}
 \end{aligned}$$

**S.69 (d)**

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

Using demorgans law  
and negating given eqn.

**S.70 (a)**

As we know that if parity relation is satisfied, put 0 otherwise 1. From the given code first parity relation  $P_1$ . For  $P_1$ ,  $D_3$ ,  $D_5$  and  $D_7$  is not satisfied as  $0 + 0 + 0 + 1 = 1$ , which is not even parity. Therefore, we have  $x = 1$ . Second parity relation  $P_2$  for  $P_2$ ,  $D_3$ ,  $D_6$  and  $D_7$  is satisfied as  $1 + 0 + 0 + 1 = 2$ , which is even parity. Therefore, we have  $y = 0$ . Third parity relation,  $P_4$  for  $P_4$ ,  $D_5$ ,  $D_6$  and  $D_7$  is not satisfied as  $0 + 0 + 0 + 1 = 1$ , which is not even parity. Therefore, we have  $z = 1$ .

Thus final position of error is in  $z y x = 1 0 1$ , i.e., fifth data bit which should be 1 in place of 0. Therefore, transmitted data was 1 0 1 0 0 1 0.

## S.71 (a)

$$\begin{aligned}\text{Output} &= \overline{AB}C + \overline{A}B\overline{C} + ABC + A\overline{BC} \\ &= \overline{C}(\overline{AB} + A\overline{B}) + C(AB + \overline{AB}) \\ &= A \oplus B \oplus C\end{aligned}$$

$$[\because \overline{\overline{AB}} + \overline{AB} = AB + \overline{AB}]$$

## S.72 (c)

$$A[B + \overline{C}(\overline{AB} + A\overline{C})] = A[B + \overline{C}(\overline{AB} \cdot \overline{AC})]$$

[By applying De Morgan's theorem  
 $\overline{X+Y} = \overline{X} \cdot \overline{Y}$ ]

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

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

$$= A[B + \overline{C}(\overline{A} \cdot \overline{A} + \overline{A} \cdot C + \overline{A} \cdot \overline{B} + \overline{B} \cdot C)]$$

[As  $X \cdot X = X$ ]

$$= A[B + \overline{C}(\overline{A} \cdot \overline{A} \cdot C + \overline{A} \cdot \overline{B} + \overline{B} \cdot C)]$$

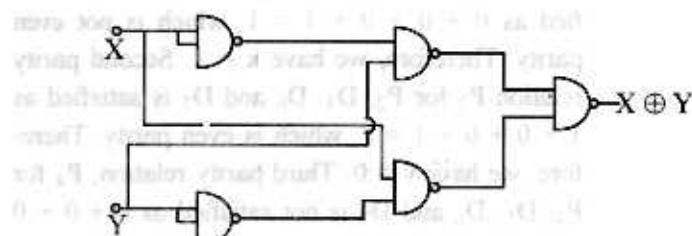
$$(0 \cdot 1 = 0) \Rightarrow A[B + \overline{A} \cdot \overline{C} + 0 + \overline{A} \cdot \overline{B} \cdot \overline{C} + 0] \quad [\text{As } X \cdot \overline{X} = 0]$$

$$= A[B + \overline{AC} + \overline{ABC}]$$

$$= AB + 0 + 0$$

$$= AB$$

## S.73 (a)



$$\overline{XY} \cdot \overline{XY} = (\overline{X} + \overline{Y})(\overline{X} + Y)$$

$$= (\overline{X} \cdot Y) + (\overline{X} \cdot \overline{Y}) = X \oplus Y$$

## S.74 (d)

The complement of the  $(\overline{A} + \overline{A} + B)(\overline{B} + \overline{B} + C)$

is equal to the  $(\overline{A} + \overline{A} + B)(\overline{B} + \overline{B} + C)$ . It can be simplified as follows:

$$= (\overline{A} + \overline{A} + B)(\overline{B} + \overline{B} + C)$$

[By applying De Morgan's theorem  
 $\overline{XY} = \overline{X} + \overline{Y}$  and  $\overline{X+Y} = \overline{X} \cdot \overline{Y}$ ]

$$= (\overline{A} + \overline{A} + B)(\overline{B} + \overline{B} + C)$$

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

$$= A \cdot (A + B) + B \cdot (B + C)$$

$$= A \cdot A + A \cdot B + B \cdot B + B \cdot B + B \cdot C$$

[As  $X \cdot X = X$ ]

$$= A + A \cdot B + B \cdot C$$

$$= A(1 + B) + B(1 + C) \quad [\text{As } 1 + X = 1]$$

$$= A \cdot 1 + B \cdot 1 \quad [\text{As } X \cdot 1 = X]$$

$$= A + B$$

## S.75 (b)

Let input ABCD be 1010

Output will be 1100

Let input ABCD be 0110

output will be 0100

This convert gray to Binary code

## S.76 (c)

$$f = \overline{wz} + yz + xy + \overline{wx}$$

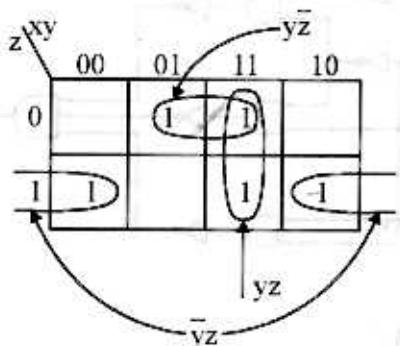
$$= (x + z)(\overline{w} + y)$$

$$= (x + z)(\overline{w} + y)$$

|    |    | yz |    |
|----|----|----|----|
|    |    | 00 | 01 |
|    |    | 11 | 10 |
| wx | 00 | 1  | 1  |
|    | 01 | 1  | 1  |
| 11 | 1  |    | 1  |
|    | 10 |    | 1  |

**S.77 (c)**

Drawing the k-map.



From the k-map, simplified function.

$$f = XY + \bar{Y}Z + Y\bar{Z}$$

**S.79 (c)**

$$A = \overline{(UV)(WX)} = UV + WX$$

**S.80 (a)**

The expression can be put into sum-of-products form by multiplying all the terms.

$$\therefore x = \bar{A}\bar{A}\bar{D} + \bar{A}B\bar{D} + \bar{A}D\bar{D} + B\bar{A}\bar{D} + BB\bar{D} + BDD$$

$$\text{As, } \bar{A}\bar{A} = 0, BB = B \text{ and } DD = 0$$

$$x = \bar{A}B\bar{D} + AB\bar{D} + B\bar{D}$$

$$= B\bar{D}(\bar{A} + A + 1)$$

The term inside the parentheses is 1.

$$\therefore x = B\bar{D}.$$

**S.81 (c)**

$$f(A, B, C) = Y = (\bar{A} + \bar{B} + C)(A + C)(\bar{A} + \bar{B})$$

In the given Boolean function, the first term has all the variables, so it is a maxterm. But the next two terms do not have B and C variable respectively. So we will convert these terms into maxterm by adding  $B\bar{B}$  and  $C\bar{C}$  respectively.

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

Since we know  $(x + yz) = (x + y)(x + z)$ , so apply this theorem in the above expression

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

$$(\bar{A} + \bar{B} + C)(\bar{A} + \bar{B} + \bar{C})$$

We also know that  $x \cdot x = x$ , then

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

$$= (A + B + C)(A + \bar{B} + C)(\bar{A} + \bar{B} + C)(\bar{A} + \bar{B} + \bar{C}) \\ M_0 \quad M_6 \quad M_2 \quad M_7$$

$$= M_0 M_2 M_6 M_7$$

$$= \Pi_M(0, 2, 6, 7)$$

**S.82 (d)**

The given Boolean expression is:

$$f(A, B, C) = (A + BC)(B + \bar{C}A)$$

$$= AB + AAC\bar{C} + BBC + C\bar{C}AB$$

$$= AB + A\bar{C} + BC + 0$$

$$[\because AA = A \text{ and } C\bar{C} = 0]$$

$$= AB + A\bar{C} + BC$$

Now we see that there are three product terms  $AB$ ,  $A\bar{C}$  and  $BC$ . There are  $C$ ,  $B$  and  $A$  variables which are not present in the product terms respectively. So, we will convert all the three product terms into minterms by multiplying  $(C + \bar{C})$ ,  $(B + \bar{B})$  and  $(A + \bar{A})$  to product terms respectively.

$$f(A, B, C) = AB(C + \bar{C}) + A\bar{C}(B + \bar{B}) \\ + BC(A + \bar{A}) \quad [\because y + \bar{y} = 1]$$

$$= ABC + AB\bar{C} + A\bar{C}B + A\bar{C}\bar{B} + ABC + \bar{ABC}$$

$$= ABC + AB\bar{C} + A\bar{B}\bar{C} + \bar{ABC} \quad [\because X + X = X]$$

$$= m_1 + m_6 + m_4 + m_3$$

Now all the terms are minterms, i.e., all the variables ( $A$ ,  $B$  and  $C$ ) are present in the produce terms. So this is the standard sum of product form. This can be represented as:

$$f(A, B, C) = m_1 + m_6 + m_4 + m_3$$

$$f(A, B, C) = \Sigma_m(3, 4, 6, 7)$$

**S.83 (a)**

$$y = (A + \bar{B}) \cdot (\bar{A} + C)$$

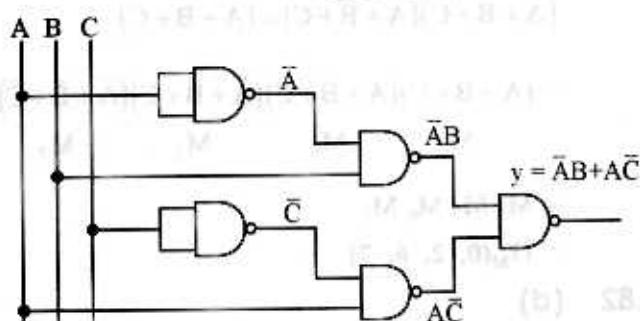
$$= (\bar{A} + \bar{B}) + (\bar{A} + C)$$

$$= \bar{A} \cdot \bar{B} + \bar{A} \cdot C$$

2.2.28

DIGITAL LOGIC

$$= \bar{A} \cdot B + A \cdot \bar{C}$$



S.84 (b)

|  |    | ABC |   |     |     |     |     |     |     |     |     |
|--|----|-----|---|-----|-----|-----|-----|-----|-----|-----|-----|
|  |    | D   | E | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|  |    |     |   |     |     |     |     |     |     |     |     |
|  | 00 |     |   |     |     |     |     |     |     |     |     |
|  | 01 |     |   |     |     | 1   | 1   | 1   | X   |     |     |
|  | 11 |     |   |     |     | 1   | 1   | X   | 1   |     |     |
|  | 10 | 1   | 1 |     |     |     |     |     |     | X   | 1   |

The terms  $A'BE$  corresponds to  $A' = 0$ ;  $B = 1$ ;  $E = 1$ ;  $C = 0$  or 1;  $D = 0$  or 1.

Similarly marks all 1's and the Karnaugh map as above.

The 1's can be covered in the optimal way, if slots marked X are set to 1's.

So there X' is in the position ABCD'E, ABC'DE, AB'CDE' are then don't care conditions to be set to 1 and used.

S.85 (b)

$$F = \overline{AB} + \overline{AC} + \overline{BC}$$

$$F = \overline{A+B} + \overline{A+\overline{C}} + \overline{B+\overline{C}}$$

S.86 (a)

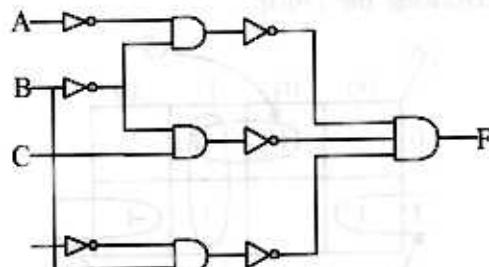
$$F = \overline{A} + \overline{B} + \overline{AC} + \overline{BC}$$

$$= \overline{A} + \overline{AC} + \overline{B} + \overline{BC}$$

$$= \overline{A}(1+C) + \overline{B}(1+C)$$

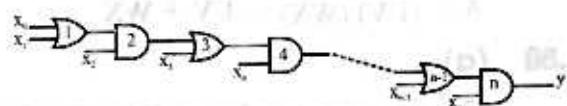
$$= \overline{A} + \overline{B}$$

$$F = \overline{AB}$$



S.87 (a)

Bubbled NAND is equivalent to OR gate and bubbled NOR gate is equivalent to AND gate. Therefore the equivalent circuit is:



Take  $n = 4$

∴ Expression becomes

$$\begin{aligned} & [(x_0 + x_1)x_2] + x_3 \cdot x_4 \\ & x_0 x_2 x_4 + x_1 x_2 x_4 + x_3 x_4 \end{aligned} \quad \dots(1)$$

Options (c) and (d) are ruled out.

Generalizing, we observe that  $x_5$  term will not come with term starting with  $x_3$ .

In every term, maximum one odd term can be there. So option (b) is ruled out.

S.88 (c)

$$f_1 = \overline{abc} + \overline{ab}\overline{c} + bc = ac + \overline{ab}$$

$$f_2 = \overline{abc} + ab + \overline{a}\overline{b}\overline{c} = ac + \overline{bc}$$

$$f_3 = \overline{a}\overline{b}\overline{c} + abc + \overline{ac} = \overline{ab} + bc$$

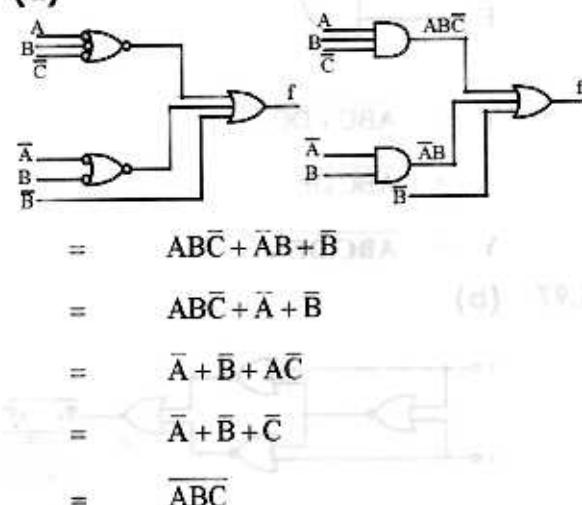
Thus,

$$P_1 = \overline{ab}, P_2 = ac, P_3 = b\overline{c}, P_4 = bc, P_5 = \overline{ab}$$

|   |  | bc |    | (a) 29.2 |    |
|---|--|----|----|----------|----|
|   |  | 00 | 01 | 11       | 10 |
| a |  |    |    | 1        | 1  |
|   |  | 1  | 1  |          |    |

|   |  | bc |    | (b) 29.2 |    |
|---|--|----|----|----------|----|
|   |  | 00 | 01 | 11       | 10 |
| a |  |    |    |          | 1  |
|   |  | 1  | 1  |          | 1  |

|   |  | bc |    | (c) 29.2 |    |
|---|--|----|----|----------|----|
|   |  | 00 | 01 | 11       | 10 |
| a |  | 1  | 1  | 1        |    |
|   |  |    |    | 1        |    |

**S.89 (d)**

$\therefore f = \overline{ABC}$  can be realized using a NAND gate.

**S.90 (d)**

Using negation law and Demorgan's law

$$f = AB + \bar{A}\bar{B}$$

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

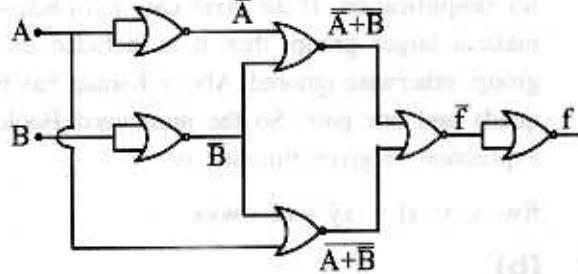
$$\bar{f} = (\overline{A \cdot B})(\overline{\bar{A} \cdot \bar{B}})$$

$$\bar{f} = (\bar{A} + \bar{B})(A + \bar{B})$$

$$\bar{f} = \bar{A}\bar{B} + A\bar{B} + \bar{B}$$

$$\bar{f} = \bar{B}$$

$$f = B$$

**S.91 (b)**

$$\overline{AB} + \overline{ABC} + A(\overline{B} + \overline{AB})$$

$$= \overline{AB} + \overline{ABC} + A(\overline{B} + A)$$

$$= A(\overline{B} + BC) + A(B + A)$$

$$= A(\overline{B} + C) + A(B + A)$$

$$= A(\overline{B} + C) + AB + AA$$

$$= A(\overline{B} + C) + AB + A$$

$$= A(\overline{B} + C) + A(B + 1)$$

$$= A(\overline{B} + C) + A = A(\overline{B} + C)\bar{A}$$

$$= A(\overline{B} + C)\bar{A} = 0$$

**S.92 (a)**

The given 4-variable function has minterms with two don't care terms. It can be minimized as follows:

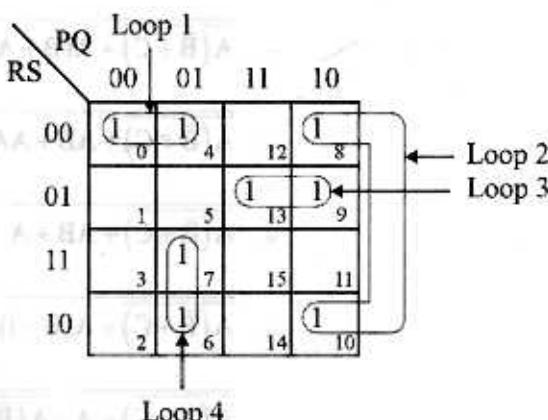
| wx | yz | 00 | 01 | 11 | 10 |
|----|----|----|----|----|----|
| 00 | 0  | d  | 1  |    | 0  |
| 01 | 1  | 1  | 1  | 1  | 0  |
| 11 | 0  | 0  | 0  | 0  | 0  |
| 10 | d  | 1  | 1  | 1  | 0  |

Here, the don't care terms are considered as 1 for simplification. If the don't care term helps to make a larger group, then it is included in the group, otherwise ignored. Above K-map has two quads and one pair. So the minimized Boolean expression of given function is:

$$f(w, x, y, z) = \bar{w}\bar{y} + x\bar{z} + \bar{w}yz$$

### S.93 (b)

The given Boolean expression has four variables, so we draw as 4-variable K-map. Then we enter 1 in the cells corresponding to the minterms present in the Boolean function. Now first look for octets, then quads and finally for pairs. It is observed that there is neither octet nor quad. There are 4 pairs of 1's as shown in figure below:



According to K-map, from first loop, the logic term is  $\bar{P}\bar{R}\bar{S}$  because in this pair B is varying and P, R and S are common, i.e., equal to 0 so that Q is removed and output is  $\bar{P}\bar{R}\bar{S}$ . Similarly in loop 2, loop 3 and loop 4 outputs are  $\bar{P}\bar{Q}\bar{S}$ ,  $\bar{P}\bar{R}S$  and  $\bar{P}QR$  respectively so that the simplified Boolean expression is logic sum of all above terms generated by each pair and output is equal to :

$$Y = \bar{P}\bar{R}\bar{S} + \bar{P}\bar{Q}\bar{S} + \bar{P}\bar{R}S + \bar{P}QR$$

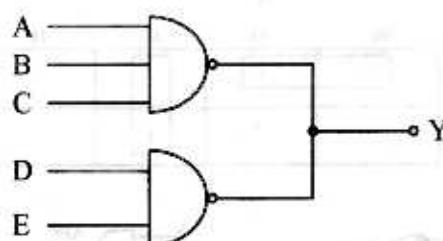
### S.94 (d)

With four Boolean variables, we can have '16' outputs and with '16' outputs we can have  $2^{16} = 65,536$  Boolean functions.

### S.95 (a)

In (b), (d) no current limiting resistors are provided. In addition, in (c) the current required to drive LED ON is not provided by the TTL inverter, since TTL  $I_{OH} = -400$  mA only. So in option (a) the +5V dc supply will provide sufficient current of the order of 10mA to turn LED on.

### S.96 (d)

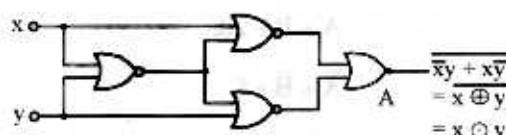


$$Y = \overline{ABC + DC}$$

$$= \overline{ABC} \cdot \overline{DC}$$

$$Y = \overline{ABC} \cdot \overline{DE}$$

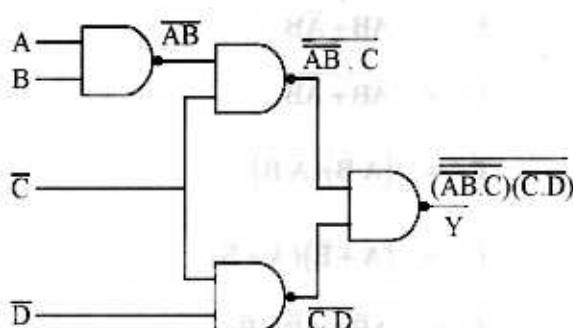
### S.97 (b)



### S.98 (c)

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

### S.99 (a)



$$Y = \overline{(ABC)(CD)}$$

$$\begin{aligned}
 &= \overline{[(\bar{A} + \bar{B})C](\bar{C} + \bar{D})} \\
 &= \overline{[\bar{A} + \bar{B} + \bar{C}] (\bar{C} + \bar{D})} \\
 &= \overline{[A \cdot B + \bar{C}] (\bar{C} + \bar{D})} \\
 &= \overline{[A \cdot B + \bar{C}] + [\bar{C} + \bar{D}]} \\
 &= \overline{[A \cdot B \cdot C] + [C \cdot D]} \\
 &= \overline{[(\bar{A} + \bar{B})C] + [C \cdot D]} \\
 &= \overline{A \cdot C + \bar{B} \cdot C + C \cdot D}
 \end{aligned}$$

**S.101 (a, d)**

$$\begin{aligned}
 A + AB &= 0 \\
 \therefore \bar{A} + B &= 0 \\
 AB &= AC \\
 \therefore AB + A\bar{C} + CD &= \bar{C} + \bar{D} \\
 \therefore AB + A\bar{C} + CD + \bar{C} &= \bar{C} + \bar{D} \\
 \therefore AB + A\bar{C} + \bar{C} + D &= \bar{C} + \bar{D} \\
 \therefore AB + A\bar{C} + \bar{C} + D + \bar{D} &= \bar{C} + \bar{D} \\
 \therefore \bar{C} + \bar{D} &= 1 \\
 \therefore \bar{CD} &= 1 \\
 \therefore CD &= 0
 \end{aligned}$$

**S.102 (a)**

$$A + A\bar{B} + A\bar{B}C = A(1 + \bar{B} + \bar{B}C) = A$$

**S.103 (b)**

$$\text{Given } f_1(w, x, y, z) = \Sigma 8, 9, 10$$

$$f_2(w, x, y, z) = \Sigma 7, 8, 12, 13, 14, 15$$

$$f(w, x, y, z) = \Sigma 8, 9$$

| wx | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| yz |    |    |    | 1  |
| 00 |    |    |    |    |
| 01 |    |    |    | 1  |
| 11 |    |    |    | 1  |
| 10 |    |    |    |    |

$$\begin{aligned}
 f_1 &= yz(\bar{w}) + y\bar{z}(x) \\
 &= yz(\bar{w} + x)
 \end{aligned}$$

(a) 011.2

| wx | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| yz |    |    |    |    |
| 00 |    |    | 1  | 1  |
| 01 |    |    | 1  |    |
| 11 | 1  | 1  |    |    |
| 10 |    |    | 1  |    |

$$f_2 = yz + \bar{w}xy + wxz$$

| wx | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| yz |    |    |    |    |
| 00 |    |    |    | 1  |
| 01 |    |    | 1  | 1  |
| 11 |    |    |    |    |
| 10 |    |    |    |    |

$$f = y\bar{z}w$$

$$\begin{aligned}
 f_1 f_2 &= y\bar{z}(\bar{w} + x)(yz + \bar{w}xy + wxz) \\
 &= (y\bar{z}w + y\bar{z}x)(yz + \bar{w}xy + wxz) \\
 &= \bar{w}y\bar{z}x = y\bar{z}wx
 \end{aligned}$$

$$\text{From figure } f = f_1 f_2 + f_3$$

$$\begin{aligned}
 y\bar{z}w &= y\bar{z}wx + f_3 \\
 &= y\bar{z}wx + y\bar{z}w\bar{x} \\
 &= y\bar{z}w(\bar{x} + x)
 \end{aligned}$$

$$\text{So, } f_3 = y\bar{z}wx$$

$$f_3 = \bar{w}xy\bar{z}$$

$$f_3 = \Sigma 9$$

**S.106 (c)**

$$A + B = B + A$$

Associative

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

(i) AND Operation

$$AB = BA$$

(commutative)

$$(AB)C = A(BC)$$

(associative)

## (ii) OR Operation

$$A + B = B + A$$

(commutative)

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

(associative)

## (iii) NAND Operation

$$\overline{AB} = \overline{BA}$$

(commutative)

$$(A \uparrow B) \uparrow C \neq A \uparrow (B \uparrow C)$$

$$\overline{A \cdot B \cdot C} \neq \overline{A} \cdot \overline{B} \cdot \overline{C}$$

$$AB + \overline{C} \neq \overline{A} + BC$$

## (iv) EX-OR Operation

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

(commutative)

$$(A \oplus B) \oplus C \neq A \oplus (B \oplus C)$$

(associative)

## S.107 (c)

|   |   | BC  |    |     |    |
|---|---|-----|----|-----|----|
|   |   | 00  | 01 | 11  | 10 |
| A | 0 | 1   | 0  | (1) | 0  |
|   | 1 | (1) | 0  | (1) | 0  |
|   |   | BC  |    | BC  |    |

$$\therefore \overline{BC} + BC = \overline{B} \oplus C$$

## S.108 (d)

$$B \oplus (B \oplus (B \oplus \dots \text{ n times}))$$

then if 'n' is even string will evaluate to 'B'  
i.e. it remain unchanged.

## S.109 (a)

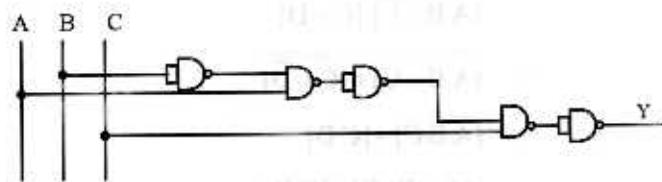
| AB |   | CD |   |    |   |   |
|----|---|----|---|----|---|---|
| CD |   | AB |   | AB |   | B |
| 1  | 1 | 1  | 1 | 1  | 1 | 1 |
| 1  | 0 | 1  | 0 | 1  | 0 | 0 |
| 0  | 1 | 0  | 1 | 0  | 1 | 1 |
| 0  | 0 | 0  | 0 | 0  | 0 | 0 |

$$\therefore \text{Essential prime Implicants} = 4$$

## S.110 (c)

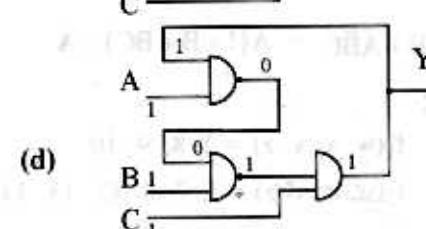
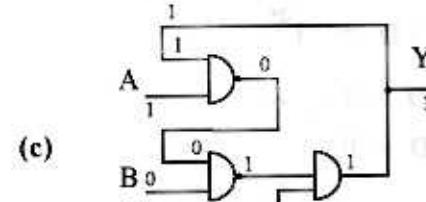
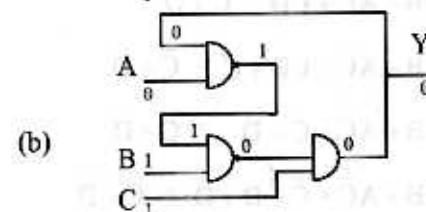
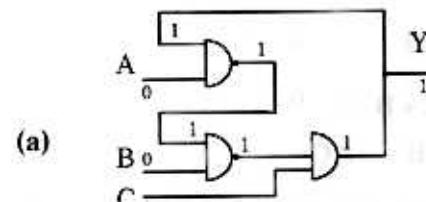
$$Z = A\bar{B}C$$

$$\therefore \bar{Z} = Z = \overline{\overline{A}\overline{B}C} = \overline{\overline{A}\overline{B}} \cdot \overline{C}$$



## S.111 (a, c, d)

The circuit diagrams for the different choices are shown below.



## S.112 (b, c)

The XOR gate can be used to implement NOT by doing  $A \oplus 1$ . Using the AND gate and the NOT realization of XOR, we can get a NAND gate, which is a universal gate. So, (c) is right.

The 2 to 1 multiplexer gives the function  $AX + BX'$ , where X is the selectro input. It can be used to implement a NOT by making  $A = 0$  and  $B = 1$ , thus we get  $X'$ . We can get an AND gate by putting  $B = 0$  and thus getting  $A \cdot X$ . Using

NOT and AND we can get a NAND, which is a universal gate. We also can't implement a NOT gate using  $A \cdot B + C$ . So d is wrong.

## S.113 (b)

|    | CD | 00 | 01 | 11 | 10 |
|----|----|----|----|----|----|
| AB | 00 |    |    | 1  |    |
|    | 01 | X  | X  | 1  | X  |
|    | 11 |    | 1  | 1  |    |
|    | 10 |    | 1  | 1  |    |

$$AD + DC = D(C + A)$$

## S.114 (d)

$$x \text{ NAND } x \Rightarrow \overline{x \cdot x} = \overline{x}$$

$$x \text{ NOR } x \Rightarrow \overline{x+x} = \overline{x}$$

$$x \text{ NAND } y \Rightarrow \overline{x \cdot y} = \overline{x}$$

$$x \text{ NOR } 1 \Rightarrow \overline{x+1} = \overline{1} = 0$$

## S.115 (b)

$$y = A + \bar{A}B$$

From Distributed law,

$$A + BC = (A + B)(A + C)$$

$$\therefore y = (A + \bar{A})(A + B)$$

$$y = A + B$$

## S.116 (b)

$$y = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{A}BC + ABC$$

$$y = \bar{A}\bar{C}(B + \bar{B}) + \bar{A}BC + ABC$$

$$= \bar{A}(\bar{C} + BC) + ABC$$

$$= \bar{A}(C + \bar{C})(\bar{C} + B) + ABC$$

$$= \bar{A}\bar{C} + \bar{A}B + ABC = \bar{A}\bar{C} + B(\bar{A} + AC)$$

$$= \bar{A}\bar{C} + B(A + \bar{A})(\bar{A} + \bar{C}) = \bar{A}\bar{C} + \bar{A}B + BC$$

## S.118 (b)

For taking minterm (SOP)

| YZ \ WX | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00      | 0  | 1  | 1  | 0  |
| 01      | X  | 0  | 0  | 1  |
| 11      | X  | 0  | 0  | 1  |
| 10      | 0  | 1  | 1  | X  |

$$f = xz' + zx'$$

## S.119 (c)

$G_1$  has delay  $\rightarrow 10 \text{ nsec}$

$G_2$  has delay  $\rightarrow 20 \text{ nsec}$

Considering the delays of both gates the table will be

| Input | Output of $G_1$ |   | Output of $G_2$ |
|-------|-----------------|---|-----------------|
|       | 0               | 1 | 1               |
| 0     | 1               | 1 | 1               |
| 1     | 1               | 0 | 1               |
| 1     | 0               | 0 | 0               |
| 1     | 0               | 0 | 1               |

## S.120 (d)

| T               | Inputs of XOR (A)                |        | Input of XOR (B) | Output |
|-----------------|----------------------------------|--------|------------------|--------|
|                 | Buffer Output [ $\delta_1 = 2$ ] | Output |                  |        |
| t <sub>0</sub>  | 0                                | 0      | 0                | 0      |
| t <sub>1</sub>  | 1                                | 0      | 1                | 1 }    |
| t <sub>2</sub>  | 1                                | 0      | 1                | 0 }    |
| t <sub>3</sub>  | 1                                | 1      | 0                | 0 }    |
| t <sub>4</sub>  | 1                                | 1      | 0                | 0 }    |
| t <sub>5</sub>  | 1                                | 1      | 0                | 1 }    |
| t <sub>6</sub>  | 1                                | 1      | 0                | 1 }    |
| t <sub>7</sub>  | 1                                | 1      | 0                | 0 }    |
| t <sub>8</sub>  | 1                                | 1      | 0                | 0 }    |
| t <sub>9</sub>  | 1                                | 1      | 0                | 0 }    |
| t <sub>10</sub> | 1                                | 1      | 0                | 0 }    |

## S.121 (c)

Find the sum-of-product form as follows

| ZW | XY | 00 | 01 | 11 | 10 |
|----|----|----|----|----|----|
| XY | 00 | X  | 1  |    | 1  |
| XY | 01 |    | 1  | X  |    |
| XY | 11 | 1  | X  | X  |    |
| XY | 10 | X  |    |    | X  |

$$= wy + w'y' + z'wx' + xyz'$$

Total literal count = 10 literals

Find product-of-sum form as follows

| ZW | XY | 00 | 01 | 11 | 10 |
|----|----|----|----|----|----|
| XY | 00 | X  |    | 0  |    |
| XY | 01 | 0  |    | X  | 0  |
| XY | 11 |    | X  | X  | 0  |
| XY | 10 | X  | 0  | 0  | X  |

$$f(z, w, x, y) = (y' + z')(w' + z')(z' + y)(x + z + w)$$

Total literal count = 9 literals

## S.122 (b)

We can show it by taking an example

I/P = 0 1 0 1 (say)

O/P = 0 1 0 1

∴ BCD to binary conversion.

## S.123 (b)

(1) W =

| RS | PQ | 00 | 01 | 10 | 11 |
|----|----|----|----|----|----|
| PQ | 00 |    | 1  | 1  | 1  |
| PQ | 01 | 1  | 1  | 1  | 1  |
| PQ | 11 |    | 1  | 1  | 1  |
| PQ | 10 |    | 1  | 1  | 1  |

$$W = R + \bar{P}Q + \bar{R}S$$

(2) X =

| RS | PQ | 00 | 01 | 11 | 10 |
|----|----|----|----|----|----|
| PQ | 00 | 1  |    |    |    |
| PQ | 01 |    |    |    |    |
| PQ | 11 | 1  |    |    |    |
| PQ | 10 | 1  |    |    |    |

$$X = PQR\bar{S} + \bar{P}Q\bar{R}\bar{S} + P\bar{Q}\bar{R}S$$

(3) Y =

| RS | PQ | 00 | 01 | 11 | 10 |
|----|----|----|----|----|----|
| PQ | 00 |    |    | 1  |    |
| PQ | 01 | 1  | 1  | 1  | 1  |
| PQ | 11 | 1  | 1  | 1  |    |
| PQ | 10 | 1  |    | 1  |    |

$$Y = RS + \bar{P}R + P\bar{Q} + \bar{P}\bar{Q}$$

$$= RS + \bar{P}R \cdot P\bar{Q} \cdot \bar{P}\bar{Q}$$

$$= RS + (\bar{P} + \bar{R})(\bar{P} + Q)(P + Q)$$

$$= RS + (\bar{P} + \bar{P}Q + \bar{P}\bar{R} + Q\bar{R})(P + Q)$$

$$= RS + \bar{P}Q + \bar{P}\bar{R} + P\bar{Q}\bar{R} + PQ\bar{R} + QR\bar{R}$$

$$= RS + \bar{P}Q + QR(P + \bar{P}) + QR$$

$$= RS + \bar{P}Q + QR$$

(4) Z =

| RS | PQ | 00 | 01 | 11 | 10 |
|----|----|----|----|----|----|
| PQ | 00 |    | 1  | 1  | 1  |
| PQ | 01 | 1  | 1  | 1  | 1  |
| PQ | 11 |    | 1  | 1  | 1  |
| PQ | 10 |    | 1  | 1  | 1  |

$$\begin{aligned}
 Z &= R + S + \overline{PQ} + \overline{P}\overline{Q}\overline{R} + \overline{P}\overline{Q}\overline{S} \\
 &= R + S + \overline{PQ}, \overline{P}\overline{Q}\overline{R}, \overline{P}\overline{Q}\overline{S} \\
 &= R + S + (\overline{P} + Q)(P + Q + R)(\overline{P} + Q + S) \\
 &= R + S + (\overline{P}Q + \overline{P}R + P\overline{Q} + \overline{Q}R)(\overline{P} + Q + S) \\
 &= R + S + \overline{P}Q + \overline{P}R + \overline{P}QS + \overline{P}R + \overline{P}QR \\
 &\quad + \overline{P}RS + \overline{P}\overline{Q} + \overline{P}\overline{Q}S + \overline{P}\overline{Q}R + \overline{Q}RS \\
 &= R + S + \overline{P}Q + \overline{P}R + \overline{P}QS + \overline{P}QR + \overline{P}RS \\
 &\quad + \overline{P}\overline{Q}S + \overline{P}\overline{Q}R + \overline{Q}RS \\
 &= R + S + \overline{P}Q(1 + S) + \overline{P}R(1 + Q) + \overline{P}RS \\
 &\quad + \overline{P}\overline{Q}S + \overline{P}\overline{Q}R + \overline{Q}RS \\
 &= R + S + \overline{P}Q + \overline{P}R + \overline{P}RS + \overline{P}\overline{Q}S + \overline{P}\overline{Q}R + \overline{Q}RS \\
 &= R + S + \overline{P}Q + \overline{P}R + \overline{P}QS + \overline{P}\overline{Q}R + \overline{Q}RS \\
 &= R + S + \overline{P}Q + \overline{P}R(1 + \overline{Q}) + \overline{P}QS + \overline{Q}RS \\
 &= R + S + \overline{P}Q + \overline{P}R + \overline{P}QS + \overline{Q}RS
 \end{aligned}$$

K. Map (1) = K. map (4)

$\therefore W = Z$

From map (2) & (4)

$$X = \bar{Z}$$

#### S.124 (d)

$$\begin{aligned}
 x'y' + xy + x'y &= x'y' + x'y + xy \\
 &= x'(y' + y) + xy \\
 &\quad [a + a' = 1] \\
 &= x' + xy \\
 &= x' + y
 \end{aligned}$$

#### S.125 (a)

$$f(a, b, c) = a'c + ac' + bc'$$

K-map for f is

|   |   | ab | 00 | 01 | 11 | 10 |
|---|---|----|----|----|----|----|
|   |   | c  | 0  | 1  | 1  | 1  |
| a | b | 0  | 1  | 1  | 1  | 1  |
|   |   | 1  | 1  | 1  | 1  | 1  |

$$\begin{aligned}
 f(a, b, c) &= a'c + ac' + bc' \\
 &= a(b + b') + a(b + b')c' \\
 &\quad + (a + a)b'c \\
 &= a'bc + a'b'c + abc' + ab'c' \\
 &\quad + ab'c + a'b'c \\
 &= 011 + 001 + 110 + 100 + 101 \\
 &\quad + 001
 \end{aligned}$$

$\therefore$  Product term  $a'c$  and  $ac'$  are essential prime implicants.

#### S.126 (a)

|  |  | BC            | 00 | 01 | 11 | 10 |
|--|--|---------------|----|----|----|----|
|  |  | A             | 0  |    |    |    |
|  |  | AC + B\bar{C} | 0  |    |    |    |
|  |  | 1             |    | 1  | 1  | 1  |
|  |  |               | 1  |    | 1  | 1  |

$$(a) ABC + \bar{A}\bar{B}\bar{C} + A\bar{B}C + AB\bar{C}$$

|  |  | BC            | 00 | 01 | 11 | 10 |
|--|--|---------------|----|----|----|----|
|  |  | A             | 0  |    |    |    |
|  |  | AC + B\bar{C} | 0  |    |    |    |
|  |  | 1             |    | 1  | 1  | 1  |
|  |  |               | 1  |    | 1  | 1  |

$$(b) \bar{B}C + AC + \bar{B}\bar{C} + \bar{A}CB$$

|  |  | BC            | 00 | 01 | 11 | 10 |
|--|--|---------------|----|----|----|----|
|  |  | A             | 0  |    |    |    |
|  |  | AC + B\bar{C} | 0  |    |    |    |
|  |  | 1             |    | 1  | 1  | 1  |
|  |  |               | 1  |    | 1  | 1  |

$$(c) AC + B\bar{C} + \bar{B}C + ABC$$

|  |  | BC            | 00 | 01 | 11 | 10 |
|--|--|---------------|----|----|----|----|
|  |  | A             | 0  |    |    |    |
|  |  | AC + B\bar{C} | 0  |    |    |    |
|  |  | 1             |    | 1  | 1  | 1  |
|  |  |               | 1  |    | 1  | 1  |

$$(d) \bar{A}C + B\bar{C} + AC$$

|  |  | BC            | 00 | 01 | 11 | 10 |
|--|--|---------------|----|----|----|----|
|  |  | A             | 0  |    |    |    |
|  |  | AC + B\bar{C} | 0  |    |    |    |
|  |  | 1             |    | 1  | 1  | 1  |
|  |  |               | 1  |    | 1  | 1  |

$\therefore$  Figure (a) matches with question.

Alternate method

$$AC + BC = AC \cdot 1 + BC \cdot 1$$

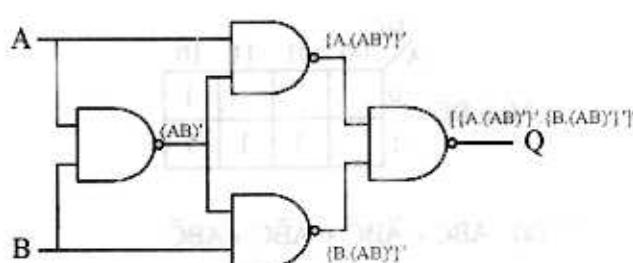
$$= AC(B + \bar{B}) + BC(A + \bar{A})$$

$$= ABC + A\bar{B}C + AB\bar{C} + \bar{A}\bar{B}C$$

### S.127 (b)

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

Now consider,



$$\text{So, } ((A.(AB)')'.(B.(AB)'))'$$

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

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

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

$$= A'B + AB'$$

$$= A \oplus B$$

Therefore, 4 NAND gates are required

### S.128 (b)

Given function

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

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

$$= AB' + A'C + ABC'$$

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

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

$$= AB' + AC' + A'C$$

### S.129 (d)

$$f = \bar{ABC} + \bar{ABC} = B(\bar{AC} + \bar{AC})$$

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

SOP of XOR = POS of XNOR

### S.130 (c)

$$A \oplus B \oplus C$$

$$= (\bar{AB} + AB) \oplus C$$

$$= (\bar{AB} + AB)C + \bar{C}(\bar{AB} + AB)$$

$$= \bar{A}\bar{B}ABC + A\bar{B}C + \bar{A}\bar{B}C$$

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

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

$$= \bar{ABC} + ABC + A\bar{B}C + \bar{A}\bar{B}C$$

Now taking options

$$ABC + \bar{A}(B \oplus C) + \bar{B}(A \oplus C)$$

$$= ABC + \bar{A}(\bar{B}C + B\bar{C}) + \bar{B}(\bar{A}C + A\bar{C})$$

$$= ABC + \bar{ABC} + \bar{ABC} + \bar{ABC} + ABC$$

$$= ABC + \bar{ABC} + \bar{ABC} + ABC$$

(i) & (ii) are equal.

### S.131 (c)

$$\overline{S_1 \oplus S_2} = \overline{\overline{S}S_2 + S\overline{S}_2}$$

If both are 0's or 1's result is always 1, in this case only.

### S.132 (d)

Consider  $a_2a_1a_0$  and  $b_2b_1b_0$

| $a_2$ | $a_1$ | $a_0$ | $b_2$ | $b_1$ | $b_0$ |
|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 0     | 0     | 0     | 0     |
| 0     | 0     | 1     | 0     | 0     | 1     |
| 0     | 1     | 0     | 0     | 1     | 0     |
| 0     | 1     | 1     | 0     | 1     | 1     |
| → 1   | 0     | 0     | 1     | 0     | 0     |
| → 1   | 0     | 1     | 1     | 0     | 1     |
| → 1   | 1     | 0     | 1     | 1     | 0     |
| → 1   | 1     | 1     | 1     | 1     | 1     |

So, the carry c will be generated if:

$$a_0 = a_1 = a_2 = b_0 = b_1 = b_2 = 1$$

$$a_2 = b_2 = 1 \text{ or } a_2 \oplus b_2 = 1 \text{ or }$$

$$a_2b_2 + a_2'b_2 = 1$$

$$a_1 = b_1 = 1 \text{ or } a_2 \oplus b_2 = a_1 \oplus b_1 = 1$$

∴ (d)

## S.133 (d)

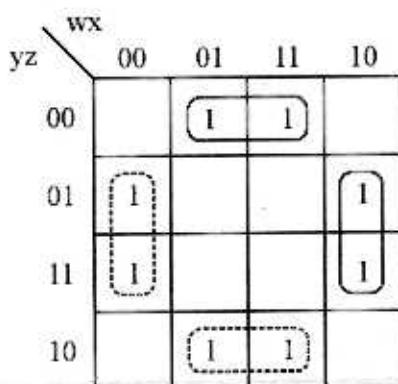
|        |   |   |        |
|--------|---|---|--------|
| $\cup$ | 0 | 0 | $\cup$ |
| 0      | d | 0 | 0      |
| 0      | 0 | d | $\cup$ |
| $\cup$ | 0 | 0 | $\cup$ |

Here, No. of product terms in SOP would be  
= 2

## S.134 (b)

Given  $f(w, x, y, z) = \Sigma(1, 3, 4, 6, 9, 11, 12, 14)$

K-Map:



$$\begin{aligned}f(w, x, y, z) &= xy'z' + w'x'z + xyz' + wx'z \\&= xz'(y' + y) + x'z(w' + w) \\&= xz' + x'z\end{aligned}$$

$\therefore f(w, x, y, z)$  is independent of two variables  $w$  and  $y$ .

## S.135 (d)

$$X^*Y = XY + X^*Y' \quad \dots\dots (i)$$

$$Z = X^*Z$$

$$Z = XZ + X^*Z' \quad [\text{from (i)}]$$

$$\begin{aligned}P : X &= YZ = YZ + Y^*Z' \quad [\text{by (i)}] \\&= Y(XZ + X^*Z') + Y^*Z' \\&= XYZ + X^*YZ' + Y^*Z'\end{aligned}$$

So, P is valid.

$$Q : Y = XZ$$

$$Y = XZ + X^*Z' \quad [\text{by (i)}]$$

$$Y = XZ + X^*Z'$$

So, Q is a valid expression.

R :  $X \cdot Y \cdot Z = 1$  is also valid.

If we will take the truth table for P, Q, R then all are valid formula.

Alternate:

| x | y | $z = x \odot y$ | P:<br>$x = y \odot z$ | Q:<br>$y = x \odot z$ | R:<br>$x \odot y \odot z = 1$ |
|---|---|-----------------|-----------------------|-----------------------|-------------------------------|
| 0 | 0 | 1               | $0 = 0$               | $0 = 0$               | $1 = 1$                       |
| 0 | 1 | 0               | $0 = 0$               | $1 = 1$               | $1 = 1$                       |
| 1 | 0 | 0               | $1 = 1$               | $0 = 0$               | $1 = 1$                       |
| 1 | 1 | 1               | $1 = 1$               | $1 = 1$               | $1 = 1$                       |

Hence all satisfies.

P, Q, R are valid.

## S.136 (c)

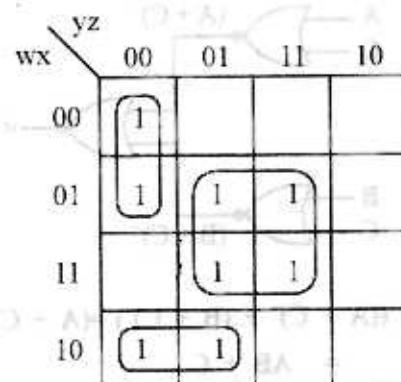
Let binary constant  $a_i$  be the value of  $f(x_1, x_2, \dots, x_n)$  for combination of variable whose decimal code is  $i$ . Then every switching function can be expressed in form,  $f(x_1, x_2, \dots, x_n) = a_0x_1x_2 \dots x_n + a_1x_1x_2 \dots x_n + \dots + a_{2^n-1}x_1x_2 \dots x_n$ . A factor  $a_2$  is either 0 or 1 if corresponding minterm is contained in canonical form of function. Then there are  $r = 2^n$  coefficient each of which can have values either 0 or 1.

Hence there are  $2^{2^n}$  possible assignment.

## S.137 (a, d)

$$f(w, x, y, z) = \Sigma(0, 4, 5, 7, 8, 9, 13, 15)$$

The K-map for f is:



$$\therefore f = w'y'z' + wx'y' + xz$$

Now, option:

(a)  $x'y'z' + w'xy' + wy'z + xz$

not equivalent to f.

(b)  $w'y'z' + wx'y' + xz$

equivalent to f.

(c)  $w'y'z' + wx'y' + xyz + xy'z$

$\Rightarrow w'y'z' + wx'y' + xz(y + y')$

$\Rightarrow w'y'z' + wx'y' + xz$

equivalent to f.

(d)  $x'y'z' + wx'y' + w'y$

not equivalent to f.

### S.138 (c)

This is SOP form and we require only 3 NAND gates.

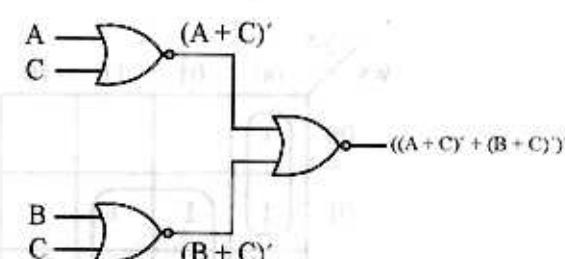
### S.139 (a)

From Karnaugh map, we get

|  |  | ab | 00 | 01 | 11 | 10 |
|--|--|----|----|----|----|----|
|  |  | cd | 00 | 1  |    | 1  |
|  |  | 01 | x  |    |    |    |
|  |  | 11 | x  |    |    |    |
|  |  | 10 | 1  | 1  |    | x  |

$$a'd' + b'd'$$

### S.140 (b)



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

### S.141 (c)

We have

$$[X + Z\{\bar{Y} + (\bar{Z} + X\bar{Y})\}][\bar{X} + \bar{Z}(X + Y)] = 1$$

Substituting  $X = 1$  and  $\bar{X} = 0$  we get

$$[1 + Z\{\bar{Y} + (\bar{Z} + 1\bar{Y})\}][0 + \bar{Z}(1 + Y)] = 1$$

$$\text{or } [1][\bar{Z}(1)] = 1$$

$\because 1 + A = 1$  and  $0 + A = A$

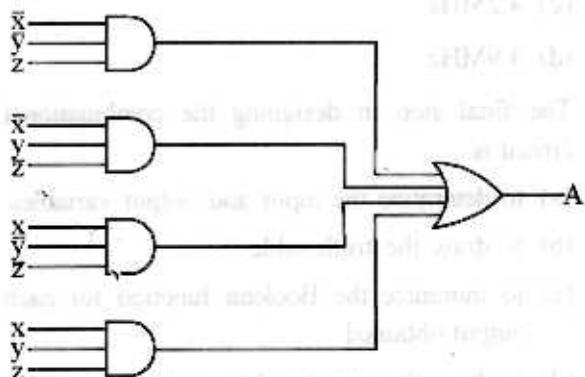
$$\text{or } \bar{Z} = 1 \rightarrow Z = 0$$

# 2.3

## COMBINATIONAL CIRCUITS

### LEVEL-1

**Q.1** Consider the following logic diagram.



In relation with full adder above logic diagram shows \_\_\_\_\_

- (a) Carry
- (b) Sum
- (c) Borrow
- (d) None

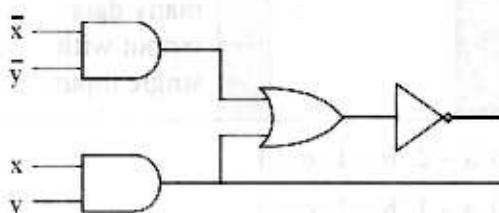
**Q.2** Full adder circuit can be implemented by

- (a) Multiplexers
- (b) Half adders
- (c) AND and OR gates
- (d) decoders

**Q.3** How many full adders are required to construct an  $m$ -bit parallel adder?

- (a)  $m$
- (b)  $m - 1$
- (c)  $m/2$
- (d)  $m + 1$

**Q.4** The following logic diagram is used as



- (a) Subtractor
- (b) Multiplier
- (c) Multiplexer
- (d) Adder

**Q.5** A combinational circuit consists of

- (a) Logic gates and a memory element
- (b) Memory elements only
- (c) Logic gates only
- (d) None of these

- Q.6** To convert a full adder into a full subtractor  
 (a) one input to carry is to be complemented  
 (b) carry is to be complemented  
 (c) sum is to be complemented  
 (d) can not be converted

- Q.7** In a 4 bit full adder, how many half adders and OR gates are required?  
 (a) 8 and 3  
 (b) 8 and 4  
 (c) 7 and 4  
 (d) 7 and 3

- Q.8** Match List I and List II and select the correct answer by using codes given below:

| List I |                | List II |                                               |
|--------|----------------|---------|-----------------------------------------------|
| (a)    | Multiplexer    | 1.      | Sequential memory                             |
| (b)    | De-multiplexer | 2.      | Converts decimal number to binary             |
| (c)    | Encoder        | 3.      | Data selector                                 |
|        |                | 4.      | Routes out many data output with single input |

- (a) a - 2, b - 1, c - 4  
 (b) a - 1, b - 2, c - 3  
 (c) a - 3, b - 4, c - 2  
 (d) a - 4, b - 3, c - 1

- Q.9** The fetching, decoding and executing of an instruction is broken down into several time intervals. Each of these intervals, involving one or more clock period, is called a  
 (a) Instruction cycle  
 (b) Process cycle  
 (c) Machine cycle  
 (d) None of these

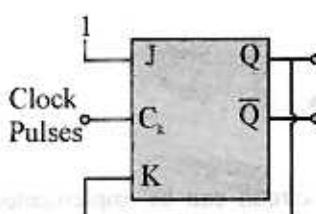
- Q.10** To implement Full adder using 8 : 1 MUX, the no. of 8 : 1 MUX required are  
 (a) 1  
 (b) 3  
 (c) 4  
 (d) 2

- Q.11** The address bus width of a memory of size  $1024 \times 8$  bits is:  
 (a) 8 bits  
 (b) 10 bits  
 (c) 13 bits  
 (d) 15 bits

- Q.12** A 4 bit modulo-6 ripple counter uses JK flip-flop. If the propagation delay of each FF is 70ns, the maximum clock frequency that can be used is equal to:  
 (a) 3.1MHz  
 (b) 3.6MHz  
 (c) 4.2MHz  
 (d) 4.9MHz

- Q.13** The final step in designing the combinational circuit is  
 (a) to determine the input and output variables  
 (b) to draw the truth table  
 (c) to minimize the Boolean function for each output obtained  
 (d) to draw the minimized logic diagram

- Q.14** In figure, assume that initially  $Q = 1$ , With clock pulses being given, the subsequent states of Q will be



- (a) 1, 0, 1, 0, 1, 0, 1, ....  
 (b) 0, 0, 1, 0, 0, 1, 0, ....  
 (c) 0, 1, 0, 1, 0, 1, 0, ....  
 (d) 1, 1, 0, 1, 0, 1, 0, ....

**Q.15** In regard to XNOR gate the following statements are made

- It has only two inputs
- It can have more than two inputs
- It is called as comparator

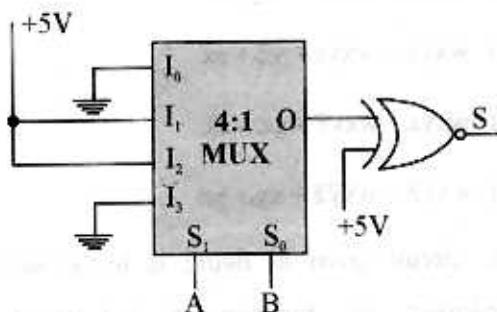
Which of the following is correct?

- 2 only
- 1, 3
- 1 only
- 2, 3

**Q.16** The operation of gate which is commutative but not associative is

- NOR
- EX-OR
- OR
- AND

**Q.17** The circuit shown below represents :



- $S(A, B) = \Pi m(0, 3)$
- $S(A, B) = \Sigma m(1, 2)$
- XOR gate with A, B as inputs
- Equality function

## LEVEL-2

**Q.18** Simplify the expression

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

- $BD$
- $\bar{B}D$
- $\bar{B}\bar{D}$
- $B\bar{D}$

**Q.19** For a binary half-subtractor having two inputs A and B, the correct set of logical expression for the output D (= A minimum B) and X (= borrow) are

- $D = \bar{A}\bar{B} + A\bar{B}, X = \bar{A}B$
- $D = \bar{A}\bar{B} + A\bar{B}, X = A\bar{B}$
- $D = AB + \bar{A}B, X = \bar{A}B$
- $D = AB + \bar{A}\bar{B}, X = A\bar{B}$

**Q.20** Implement the following Boolean function with a 4\*1 multiplexer and external gates. Connect inputs A and B to the selection lines. The input requirements for the four data lines will be function of variables C and D. These values are obtained by expressing F as function which may have to be implemented with external gates

$$F(A, B, C, D) = \Sigma (1, 3, 4, 11, 12, 13, 14, 15)$$

- $\Sigma F(A, B, C, D) = \Sigma (1, 3, 4, 11, 12, 13, 14, 15)$
- $\Sigma 1, 2, 7, 9$
- $\Sigma 1, 7, 11, 15$
- None

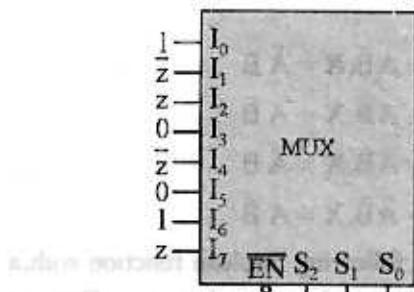
**Q.21** In the following question, match each of the items A, B and C on the left with an approximation item on the right

| Group I |                            | Group II |                                   |
|---------|----------------------------|----------|-----------------------------------|
| A       | Shift register can be used | 1.       | for code conversion               |
| B       | A multiplexer can be used  | 2.       | to generate memory slip to select |
| C       | A decoder can be used      | 3.       | for parallel to serial conversion |
|         |                            | 4.       | as many-to-one switch             |
|         |                            | 5.       | for analog to digital conversion  |

Codes:

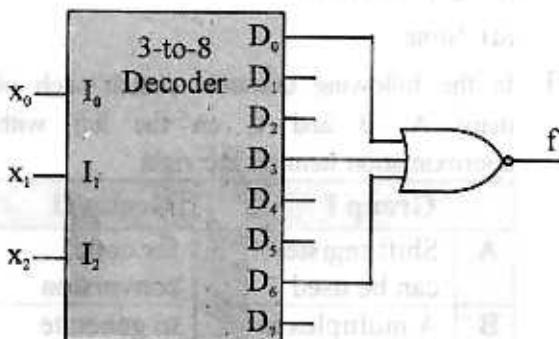
|     | A | B | C |
|-----|---|---|---|
| (a) | 1 | 2 | 3 |
| (b) | 1 | 3 | 5 |
| (c) | 5 | 4 | 2 |
| (d) | 3 | 4 | 1 |

**Q.22** The 8-to-1 multiplexer shown in figure, realize which of the following Boolean expression



- (a)  $w'x'z' + wy'z' + w'y'z + wxz$
- (b)  $wxz + w'xz + wxz' + w'x'y'$
- (c)  $wx'z + w'x'z' + wxz + xy'z'$
- (d) MUX is not enable

**Q.23**  $f(x_2, x_1, x_0) = ?$

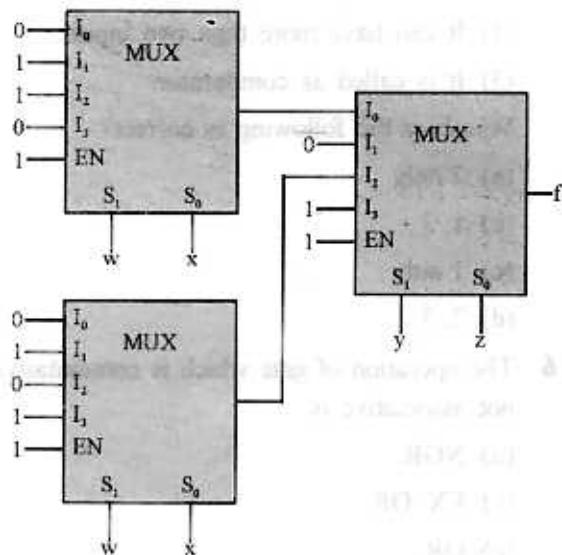


- (a)  $\Sigma_M(1, 2, 4, 5, 7)$
- (b)  $\Pi_M(1, 2, 4, 5, 7)$
- (c)  $\Sigma_M(0, 3, 6)$
- (d) None of above

**Q.24** In three-bit subtractor, the difference is \_\_\_\_\_ where x, y, z are inputs of the subtractor.

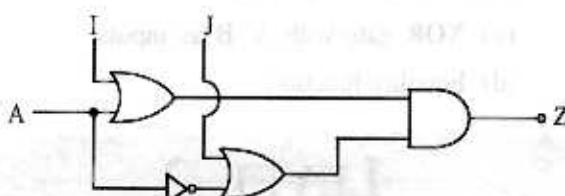
- (a)  $D = xyz + xy\bar{z} + \bar{x}yz + x\bar{y}z$
- (b)  $D = xyz + \bar{x}\bar{y}z + x\bar{y}\bar{z} + \bar{x}y\bar{z}$
- (c)  $D = \bar{x}\bar{y}z + xy\bar{z} + \bar{x}yz + x\bar{y}z$
- (d) None of these

**Q.25**  $f = ?$



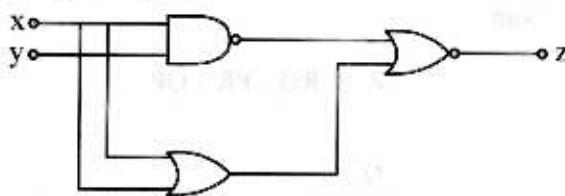
- (a)  $wxyz + \bar{w}xyz + \bar{xy} + \bar{yz}$
- (b)  $wxyz + \bar{w}xyz + y\bar{z} + zx$
- (c)  $\bar{w}xyz + wxy\bar{z} + gz + zx$
- (d)  $\bar{w}\bar{xy}z + w\bar{xy}z + xy + yz$

**Q.26** The circuit given in figure is to be used to implement the function  $Z = f(A, B) = \bar{A} + B$ . What values should be selected for I and J?



- (a) I = 0, J = B
- (b) I = B, J = 1
- (c) I = 1, J = B
- (d) I =  $\bar{B}$ , J = 0

**Q.27** Which of the following is the truth table of the given logic?



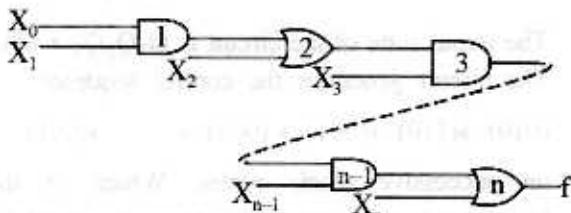
| x | y | z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| x | y | z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| x | y | z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

| x | y | z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Q.28** In the given network of AND and OR gates, it can be written as:



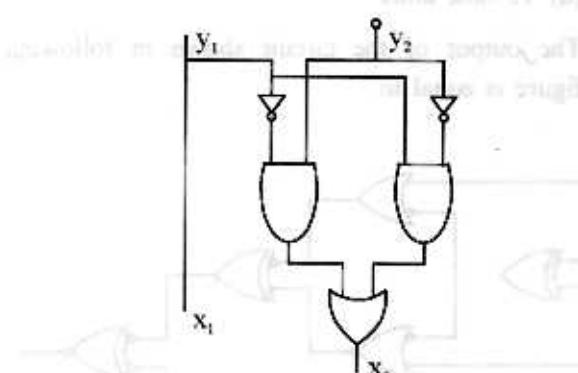
- (a)  $x_0 x_1 x_2 \dots x_n + x_1 x_2 \dots x_n + x_2 x_3 \dots x_n \dots x_n$
- (b)  $x_0 x_1 + x_2 x_3 + \dots + x_{n-1} \dots x_n$
- (c)  $x_0 x_1 x_3 \dots x_{n-1} + x_2 x_3 x_5 \dots x_{n-1} + \dots + x_{n-2} x_{n-1} + x_n$
- (d)  $x_0 + x_1 + x_2 + \dots + x_n$

**Q.29** The following k-map implements

| WX |  | YZ | 00             | 01             | 11             | 10             |
|----|--|----|----------------|----------------|----------------|----------------|
|    |  | 00 | D <sub>0</sub> | D <sub>1</sub> | D <sub>3</sub> | D <sub>2</sub> |
| 01 |  |    | D <sub>4</sub> | D <sub>5</sub> | D <sub>7</sub> | D <sub>6</sub> |
| 11 |  | x  | x              | x              | x              | x              |
| 10 |  |    | D <sub>8</sub> | D <sub>9</sub> | x              | x              |

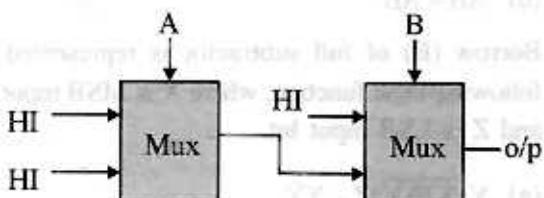
- (a) BCD to Decimal Decoder
- (b) Decimal to BCD Encoder
- (c) 1 of 10 Decoder
- (d) Choice (a) and (c) both

**Q.30** The logic circuit shown converts  $y_1 - y_2$  into



- (a) Gray code
- (b) BCD
- (c) error detecting code
- (d) excess - 3 code

**Q.31** How is the operation dependent on A and B?

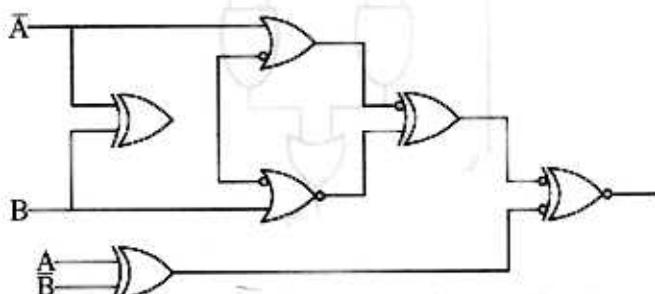


- (a) LO for A and B is LO
- (b) Independent of A and B
- (c) HI for A and B is HI
- (d) LO for A and B is HI

## LEVEL-3

- Q.32** A 4-bit carry look ahead adder, which adds two 4-bit numbers, is designed using AND, OR, NOT, NAND, NOR gates only. Assuming that all the inputs are available in both complemented and uncomplemented forms and the delay of each gate is one time unit, what is the overall propagation delay of the adder? Assume that the carry network has been implemented using two-level AND-OR logic.
- 4 time units
  - 6 time units
  - 10 time units
  - 12 time units

- Q.33** The output of the circuit shown in following figure is equal to



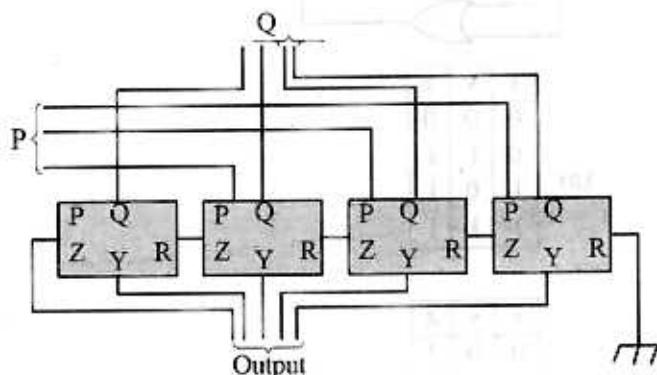
- 0
- 1
- $\bar{A}B + A\bar{B}$
- $AB + A\bar{B}$

- Q.34** Borrow (B) of full subtractor is represented by following logic function, where X is MSB input bit and Z is LSB input bit.

- $Y(\bar{X} \oplus Y)Z + X\bar{Y}$
- $Y(X \oplus \bar{Y})Z + \bar{X}\bar{Y}$
- $Y(X \oplus Y)Z + \bar{X}\bar{Y}$
- $(\bar{X} \oplus Y)Z + \bar{X}\bar{Y}$

- Q.35** The circuit shown in figure has 4 boxes each described by inputs P, Q, R and outputs Y, Z with

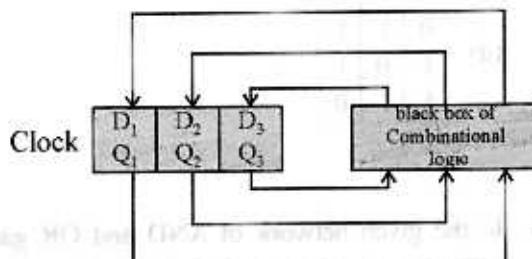
$$Z = R\bar{Q} + PR + \bar{Q}P$$



The circuit acts as a

- 4 bit adder giving  $P + Q$
- 4 bit subtractor giving  $P - Q$
- 4 bit subtractor giving  $Q - P$
- 4 bit adder giving  $P + Q + R$

- Q.36** Consider the following control circuit which contains a 3-bit register and a black box with some combinational logic



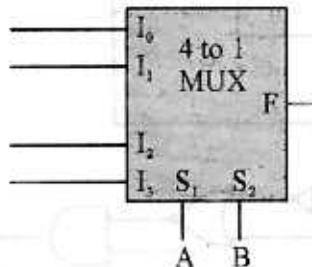
The initial state of the circuit is  $Q_1Q_2Q_3 = 000$ . The circuit generates the control sequence.

$$(010) \rightarrow (110) \rightarrow (001) \rightarrow (001) \rightarrow \dots \rightarrow (001)$$

on successive clock cycles. Which of the following sets of equations are implemented by the combinational logic in the black box?

- $D_1 = Q_1'Q_2'Q_3'$ ,  $D_2 = Q_1'$ ,  $D_3 = Q_2$
- $D_1 = Q_2'Q_3'$ ,  $D_2 = Q_1'Q_2'$ ,  $D_3 = Q_1Q_2Q_3'$
- $D_1 = Q_1'Q_2$ ,  $D_2 = Q_1'Q_3'$ ,  $D_3 = Q_1 \vee Q_3$
- $D_1 = Q_1 \vee Q_2$ ,  $D_2 = Q_1'$ ,  $D_3 = Q_1Q_2'$

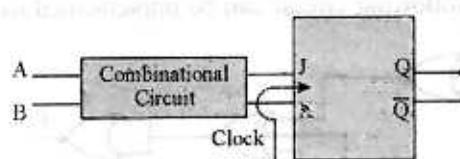
- Q.37** In the figure shown a 4 to 1 MUX to be used to implement the sum S of a 1-bit full adder with input bits P and Q and the carry input  $C_{in}$ . Which of the following combinations of inputs to  $I_0$ ,  $I_1$ ,  $I_2$  and  $I_3$  of the MUX will realize the sum S?



- (a)  $I_0 = I_1 = C_{in}$ ;  $I_2 = I_3 = \bar{C}_{in}$
- (b)  $I_0 = I_1 = \bar{C}_{in}$ ;  $I_2 = I_3 = C_{in}$
- (c)  $I_0 = I_3 = \bar{C}_{in}$ ;  $I_1 = I_2 = C_{in}$
- (d)  $I_0 = I_3 = C_{in}$ ;  $I_1 = I_2 = \bar{C}_{in}$

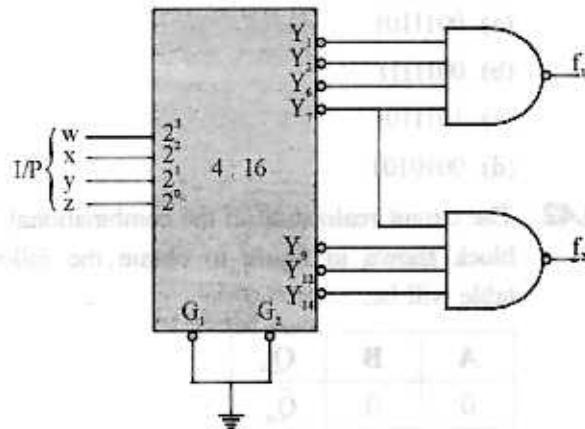
- Q.38** To realize the given truth table from the circuit in the figure, then input to J in terms of A and B would have to be

| A | B | $Q_{n+1}$ | J |
|---|---|-----------|---|
| 0 | 0 | $Q_n$     | 0 |
| 0 | 1 | 1         | 1 |
| 1 | 0 | $Q_n$     | 0 |
| 1 | 1 | 0         | x |



- (a)  $\overline{AB}$
- (b) B
- (c)  $\overline{A}$
- (d) AB

- Q.39** Consider circuit of 4 : 16 Demux below:



Consider the following expressions

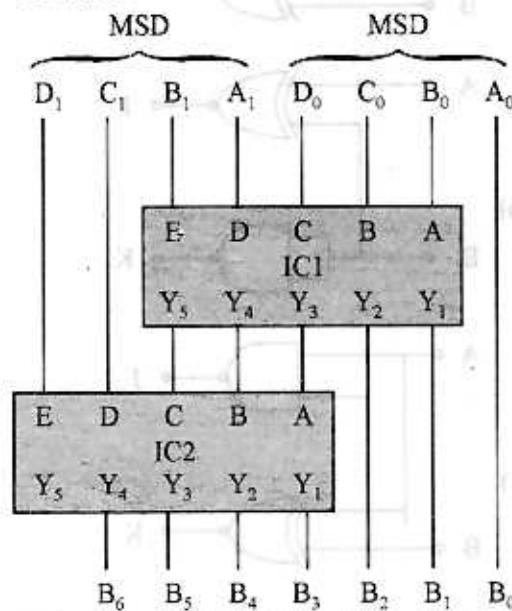
- (i)  $f_1 = \Sigma m(1, 2, 6, 7)$
- (ii)  $f_2 = \Sigma m(7, 9, 12, 14)$

State which among the following options is true

- (a) (i) only
- (b) (ii) only
- (c) both incorrect
- (d) (i) and (ii) both

#### Common Data For Questions 40 & 41:

The figure shows a 2-decade BCD-to-Binary converter.



- Q.40** The output of IC1 for an input of 29 will be,

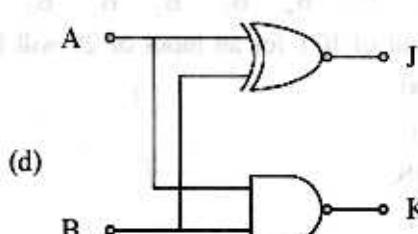
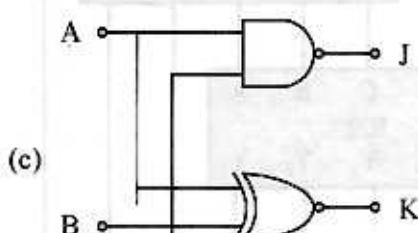
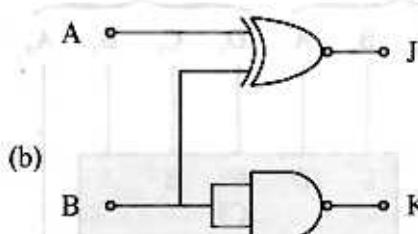
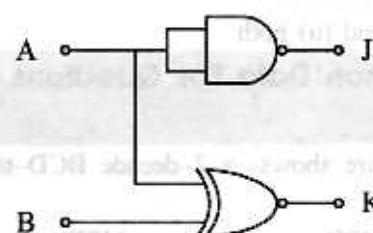
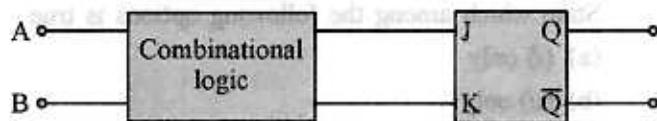
- (a) 01100
- (b) 01110
- (c) 01010
- (d) 01111

**Q.41** For the above part(a), the full binary output is,

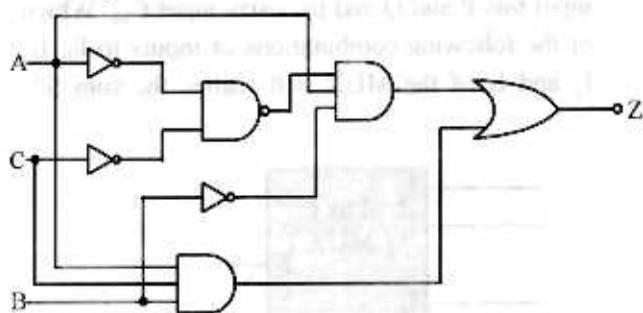
- (a) 0011101
- (b) 0011111
- (c) 1011101
- (d) 0010101

**Q.42** The circuit realization of the combinational logic block shown in figure to obtain the following table will be,

| A | B | $\bar{Q}_n$ |
|---|---|-------------|
| 0 | 0 | $\bar{Q}_n$ |
| 0 | 1 | 1           |
| 1 | 0 | $\bar{Q}_n$ |
| 1 | 1 | 0           |

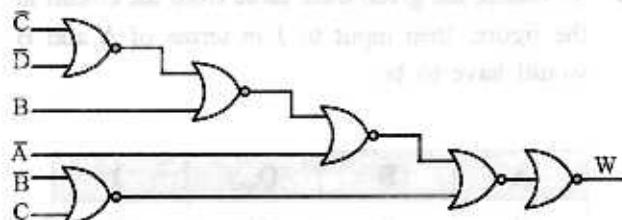


**Q.43** The simplified form of the logic circuit shown in figure below is,



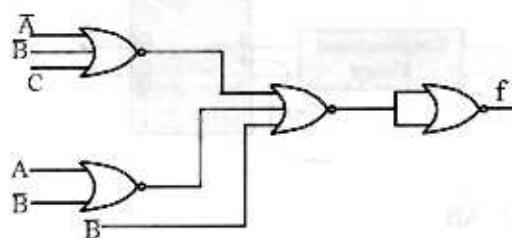
- (a)
- (b)
- (c)
- (d)

**Q.44** The following circuit represents :



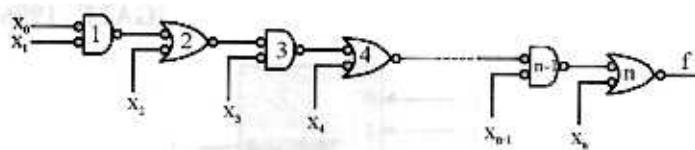
- (a)  $W = \overline{AB + C} + DA$
- (b)  $W = AB + B\bar{C} + ACD$
- (c)  $W = A(B + CD)BC$
- (d) None of these

**Q.45** The following circuit can be implemented using :



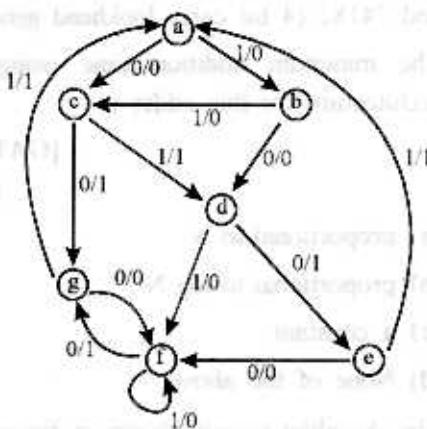
- (a) a single AND gate
- (b) a NAND gate
- (c) an EXOR gate
- (d) a NAND gate and a NOT gate

**Q.46** The output 'y' of the circuit is :



- (a)  $x_0x_1x_2x_3\dots x_n + x_1x_2x_3\dots x_n + x_2x_3\dots x_n + \dots + x_{n-1}x_n + x_n$
- (b)  $x_0x_2x_4\dots x_n + x_1x_3x_5\dots x_n + x_3x_4x_5x_6\dots x_n + \dots + x_{n-1}x_n$
- (c)  $x_0x_2x_4\dots x_n + x_1x_3x_5\dots x_{n-1} + x_3x_4\dots x_n + \dots + x_{n-2}x_n$
- (d)  $x_0x_2x_4\dots x_n + x_1x_2x_4x_6\dots x_n + x_3x_4x_6x_8\dots x_n + \dots + x_{n-1}x_n$

**Q.47** Following state diagram shows clocked sequential circuit:

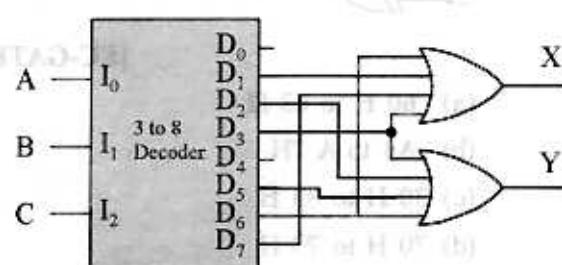


How many states the sequential circuit has?

- (a) 4
- (b) 5
- (c) 6
- (d) 7

#### Common Data For Questions 48 & 49:

The building block shown in figure is a active high output decoder:



**Q.48** The output X is:

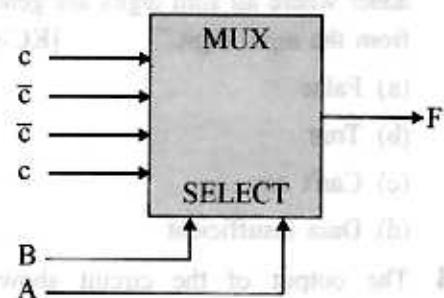
- (a)  $A\bar{B} + \bar{B}C$
- (b)  $BC + CA$
- (c)  $\bar{A}C + AB$
- (d)  $AB + BC + CA$

**Q.49** The output Y is:

- (a)  $AB + BC + CA$
- (b)  $ABC + \bar{B}C + AB$
- (c)  $\bar{A}B + C\bar{B} + AC$
- (d)  $\bar{A}B + B\bar{C} + A\bar{B}C$

## GATE QUESTIONS

**Q.50** The output F of the below multiplexer circuit can be represented by [GATE 1987]



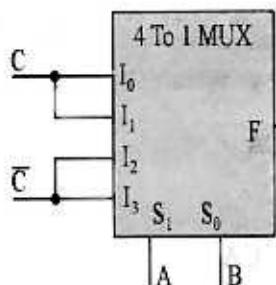
- (a)  $AB + B\bar{C} + \bar{C}A + \bar{B}\bar{C}$
- (b)  $A \oplus B \oplus C$
- (c)  $A \oplus B$
- (d)  $\bar{A}BC + \bar{A}\bar{B}\bar{C} + ABC$

**Q.51** All digital circuits can be realized using only

[GATE 1992]

- (a) Ex-Or gates
- (b) Multiplexers
- (c) Half adders
- (d) OR gates

- Q.52** The logic realized by the circuit shown in Figure below is



[EC-GATE 1992]  
[2-Marks]

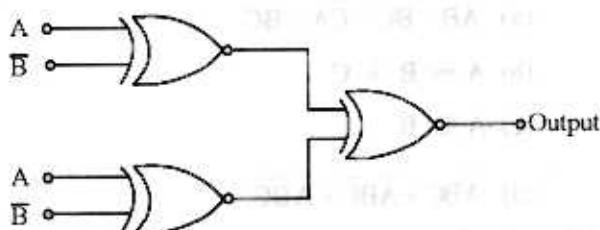
- (a)  $F = AC$
- (b)  $F = A \oplus C$
- (c)  $F = BC$
- (d)  $F = B \oplus C$

- Q.53** State whether True or False. Give reasons for your answer

"The look-ahead carry adder is a parallel carry adder where all sum digits are generated directly from the input digit." [EC-GATE 1994]

- (a) False
- (b) True
- (c) Can't say
- (d) Data insufficient

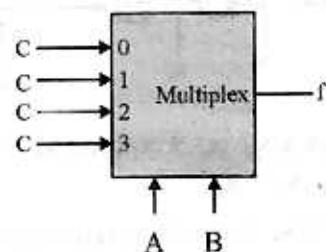
- Q.54** The output of the circuit shown (figure) is equal to



[EC-GATE 1995]

- (a) 0
- (b) 1
- (c)  $\bar{A}B + A\bar{B}$
- (d)  $(A * B) * (\bar{A} * \bar{B})$

- Q.55** Consider the circuit in the figure f implements [GATE 1996]



- (a)  $\bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC + A\bar{B}\bar{C}$
- (b)  $A + B + C$
- (c)  $A \oplus B \oplus C$
- (d)  $AB + BC + CA$

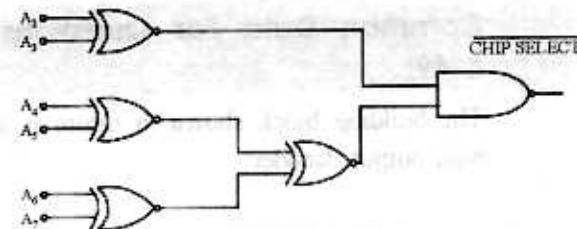
- Q.56** An N-bit carry lookahead adder, where N is a multiple of 4, employs ICs 74181 (4 bit ALU) and 74182 (4 bit carry lookahead generator).

The minimum addition time using the best architecture for this adder is

[GATE 1997]  
[1-Mark]

- (a) proportional to  $N$
- (b) proportional to  $\log N$
- (c) a constant
- (d) None of the above

- Q.57** The decoding circuit shown in figure has been used to generate the active low chip select signal for a microprocessor peripheral. (The address lines are designated as  $A_0$  to  $A_7$  for I/O addresses) The peripheral will correspond to I/O addresses in the range:



[EC-GATE 1997]

- (a) 60 H to 63 H
- (b) A4 to A 7H
- (c) 30 H to 33 H
- (d) 70 H to 73 H

**Q.58** A multiplexer with 4 bit data select input is a  
[GATE 1998]

[1-Mark]

- (a) 4 : 1 multiplexor
- (b) 2 : 1 multiplexor
- (c) 16 : 1 multiplexor
- (d) 8 : 1 multiplexor

**Q.59** The number of full and half-adders required to add 16-bit number is [GATE 1999]

[2-Marks]

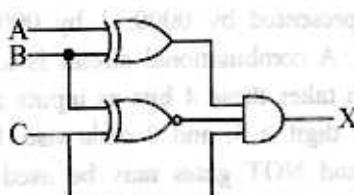
- (a) 8 half-adders, 8 full-adders
- (b) 1 half-adders, 15 full-adders
- (c) 16 half-adders, 0 full-adders
- (d) 4 half-adders, 12 full-adders

**Q.60** For a binary half-subtractor having two inputs A and B, the correct set of logical expressions for the outputs D (= A minus B) and X (=borrow) are [EC-GATE 1999]

[2-Marks]

- (a)  $D = AB + \bar{A}\bar{B}$ ,  $X = \bar{A}B$
- (b)  $D = \bar{A}B + A\bar{B}$ ,  $X = \bar{A}B$
- (c)  $D = \bar{A}B + A\bar{B} + AB$ ,  $X = A\bar{B}$
- (d)  $D = AB + \bar{A}\bar{B}$ ,  $X = A\bar{B}$

**Q.61** For the logic circuit shown in the figure, the required input condition (A, B, C) to make the output ( $X$ ) = 1 is

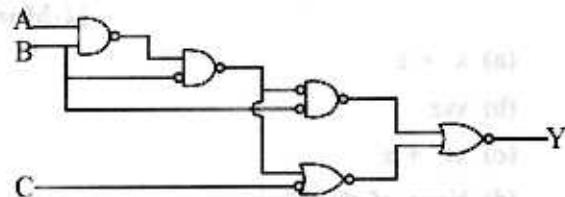


[EC-GATE 2000]

[1-Mark]

- (a) 0, 1, 1
- (b) 0, 0,
- (c) 1, 1, 1
- (d) 1, 0, 1

**Q.62** For the logic circuit shown in the figure, the simplified Boolean expression for the output Y is

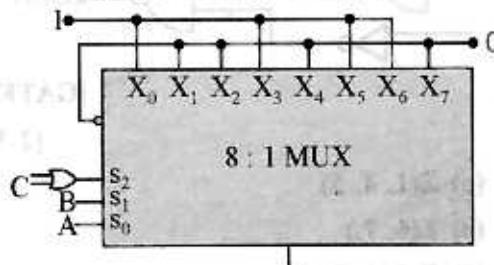


[EC-GATE 2000]

[2-Marks]

- (a)  $A+B+C$
- (b) B
- (c) C
- (d) A

**Q.63** In the TTL circuit in the figure,  $S_2$  and  $S_0$  are select lines and  $X_7$  and  $X_0$  are input lines.  $S_0$  and  $X_0$  are LSBs. The output Y is

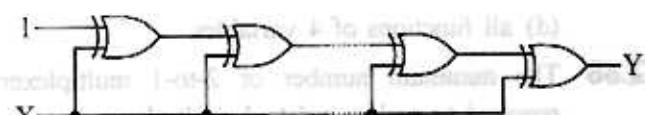


[EC-GATE 2001]

[2-Marks]

- (a) indeterminate
- (b)  $\bar{C}(A \oplus B) + C(A \oplus B)$
- (c)  $A \oplus B$
- (d)  $\overline{A \oplus B}$

**Q.64** If the input to the digital circuit (in the figure) consisting of a cascade of 20 XOR-gates is X, then the output Y is equal to



[EC-GATE 2002]

[1-Mark]

- (a) X
- (b)  $\bar{X}$
- (c) 0
- (d) 1

**Q.65** Let  $f(A, B) = A' + B$ . Simplified expression of function  $f(f(x + y, y), z)$  is [GATE 2002] [2-Marks]

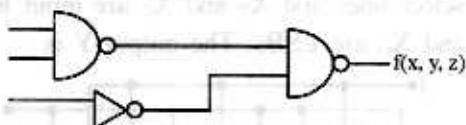
- (a)  $x' + z$
- (b)  $xyz$
- (c)  $xy' + z$
- (d) None of the above

**Q.66** Consider the following logic circuit whose inputs are functions  $f_1, f_2, f_3$  and output is  $f$ .

$$f(s, y, z) = \sum(0, 1, 3, 5)$$

$$f_2(s, y, z) = \sum(6, 7)$$

$$f_3(s, y, z) = \sum(1, 4, 5)$$



[GATE 2002]

[2-Marks]

- (a)  $\Sigma(1, 4, 5)$
- (b)  $\Sigma(6, 7)$
- (c)  $\Sigma(0, 1, 3, 5)$
- (d) None of the above

**Q.67** Without any additional circuitry, an 8:1 MUX can be used to obtain [EC-GATE 2003]

[1-Mark]

- (a) some but not all Boolean functions of 3 variables
- (b) all functions of 3 variables and some but not all of 4 variables
- (c) all functions of 3 variables but none of 4 variables
- (d) all functions of 4 variables

**Q.68** The minimum number of 2-to-1 multiplexers required to realize a 4-to-1 multiplexer is

[EC-GATE 2004]

[2-Marks]

- (a) 4
- (b) 3
- (c) 2
- (d) 1

**Q.69** A Boolean function of two variables  $x$  and  $y$  is defined as follows:

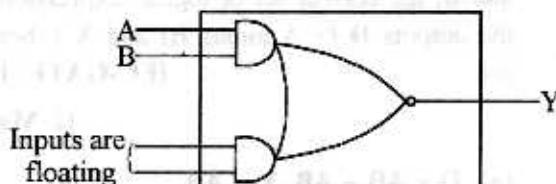
$$f(0, 0) = f(0, 1) = f(1, 1) = 1; f(1, 0) = 0$$

Assuming complements of  $x$  and  $y$  are not available, a minimum cost solution for realizing  $f$  using only 2-input NOR gates and 2-input OR gates (each having unit cost) would have a total cost of [EC-GATE 2004]

[2-Marks]

- (a) 4 unit
- (b) 3 unit
- (c) 2 unit
- (d) 1 unit

**Q.70** The figure shows the internal schematic of a TTL AND-OR-invert (AOI) gate. For the inputs shown in the figure, the output  $Y$  is



[EC-GATE 2004]

[1-Mark]

- (a) 1
- (b)  $AB$
- (c) 0
- (d)  $\overline{AB}$

**Q.71** A circuit outputs a digit in the form of 4 bits. 0 is represented by 0000, 1 by 0001, ..., 9 by 1001. A combinational circuit is to be designed which takes these 4 bits as inputs and outputs 1 if the digit  $\geq 5$ , and 0 otherwise. If only AND, OR and NOT gates may be used, what is the minimum number of gates required?

[GATE 2004]

[2-Marks]

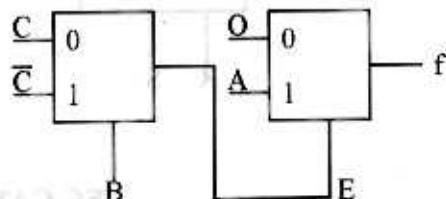
- (a) 2
- (b) 3
- (c) 4
- (d) 5

**Q.72** The switching expression corresponding to  $f(A, B, C, D) = \Sigma(1, 4, 5, 9, 11, 12)$

[GATE 2005]  
[1-Mark]

- (a)  $BC'D' + A'C'D + AB'D$
- (b)  $ABC' + ACD + B'C'D$
- (c)  $ACD' + A'BC' + A'C'D'$
- (d)  $A'BD + ACD' + BCD'$

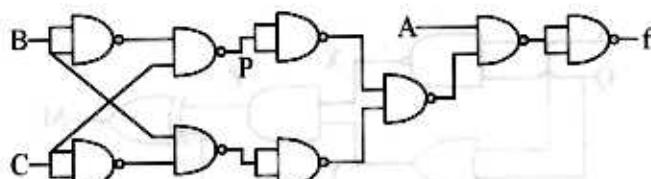
**Q.73** The Boolean function  $f$  implemented in the figure using two input multiplexers is



[EC-GATE 2005]  
[1-Mark]

- (a)  $ABC + A\bar{B}\bar{C}$
- (b)  $A\bar{B}C + A\bar{B}\bar{C}$
- (c)  $\bar{A}BC + \bar{A}\bar{B}\bar{C}$
- (d)  $\bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C}$

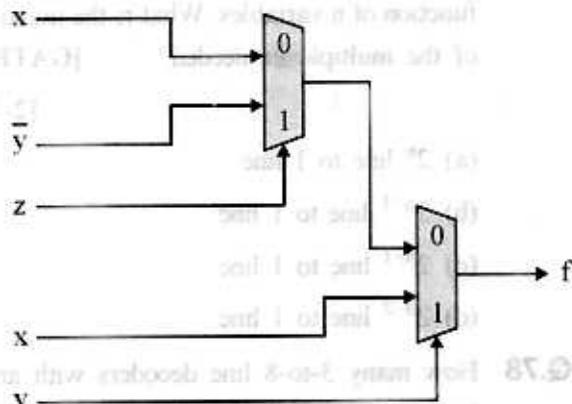
**Q.74** The point P in the following figure is stuck-at-1. The output  $f$  will be



[EC-GATE 2006]  
[2-Marks]

- (a) A
- (b)  $\bar{A}$
- (c)  $ABC$
- (d)  $\overline{ABC}$

**Q.75** Consider the circuit below. Which one of the following options correctly represents  $f(x, y, z)$ ?



[GATE 2006]  
[2-Marks]

- (a)  $x\bar{z} + xy + \bar{y}z$
- (b)  $x\bar{z} + xy + y\bar{z}$
- (c)  $xz + xy + \bar{y}z$
- (d)  $xz + x\bar{y} + \bar{y}z$

**Q.76** In a look-ahead carry generator, the carry generate function  $G_i$  and the carry propagate function  $P_i$  for inputs  $A_i$  and  $B_i$  are given by:

$$P_i = A_i \oplus B_i \text{ and } G_i = A_i B_i$$

The expressions for the sum bit  $S_i$  and the carry bit  $C_{i+1}$  of the look-ahead carry adder are given by:

$$S_i = P_i \oplus C_i \text{ and } C_{i+1} = G_i \oplus P_i C_i, \text{ where } C_0 \text{ is the input carry.}$$

Consider a two-level logic implementation of the look-ahead carry generator. Assume that all  $P_i$  and  $G_i$  are available for the carry generator circuit and that the AND and OR gates can have any number of inputs. The number of AND gates and OR gates needed to implement the look-ahead carry generator for a 4-bit adder with  $S_3, S_2, S_1, S_0$  and  $C_4$  as its output are respectively:

[GATE 2007]

- (a) 6, 3
- (b) 10, 4
- (c) 6, 4
- (d) 10, 5

**Q.77** Suppose only one multiplexer and one inverter are allowed to be used to implement any Boolean function of  $n$  variables. What is the minimum size of the multiplexer needed? [GATE 2007]

[2-Marks]

- (a)  $2^n$  line to 1 line
- (b)  $2^{n+1}$  line to 1 line
- (c)  $2^{n-1}$  line to 1 line
- (d)  $2^{n-2}$  line to 1 line

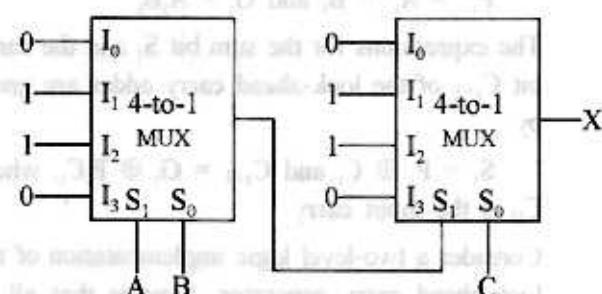
**Q.78** How many 3-to-8 line decoders with an enable input are needed to construct a 6-to-64 line decoder without using any other logic gates?

[GATE 2007]

[1-Mark]

- (a) 7
- (b) 8
- (c) 9
- (d) 10

**Q.79** In the following circuit, X is given by



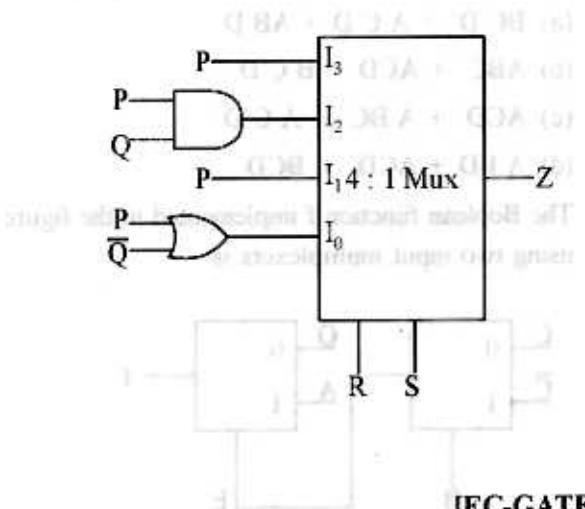
[EC-GATE 2007]

[2-Marks]

- (a)  $X = \bar{A}BC + A\bar{B}C + ABC + A\bar{B}\bar{C}$
- (b)  $X = A\bar{B}\bar{C} + \bar{A}BC + \bar{A}\bar{B}C + ABC$
- (c)  $X = A B + B C + A C$
- (d)  $X = \bar{A}\bar{B} + \bar{B}\bar{C} + \bar{A}\bar{C}$

**Q.80** For the circuit shown in the following figure,  $I_0 - I_3$  are inputs to the 4 : 1 multiplexer. R (MSB) and S are control bits.

[2-Marks]



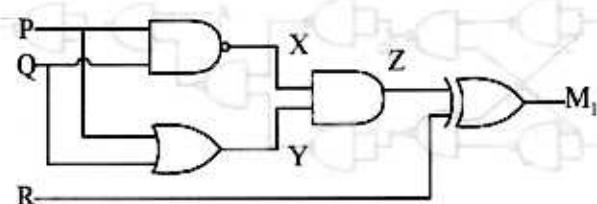
[EC-GATE 2008]

[2-Marks]

The output  $Z$  can be represented by

- (a)  $P\bar{Q} + P\bar{Q}R + \bar{P}\bar{Q}\bar{S}$
- (b)  $PQ + P\bar{Q}S + \bar{Q}RS$
- (c)  $P\bar{Q}\bar{R} + \bar{P}QR + PQRS + \bar{Q}RS$
- (d)  $PQR + PQRS + \bar{P}QRS + \bar{Q}RS$

**Q.81** Which of the following Boolean Expressions correctly represents the relation between P, Q, R and  $M_1$ ?

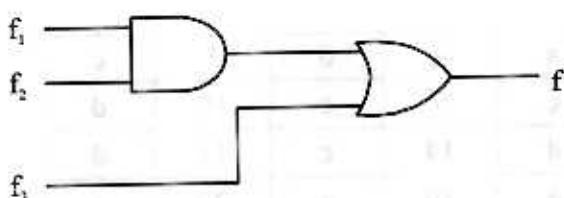


[EC-GATE 2008]

[2-Marks]

- (a)  $M_1 = (P \text{ OR } Q) \text{ XOR } R$
- (b)  $M_1 = (P \text{ AND } Q) \text{ XOR } R$
- (c)  $M_1 = (P \text{ XOR } Q) \text{ XOR } R$
- (d)  $M_1 = (P \text{ NOR } Q) \text{ XOR } R$

- Q.82** Given  $f_1$ ,  $f_2$  and  $f$  in canonical sum of products form (in decimal) for the circuit:



$$f_1 = \Sigma_m(4, 5, 6, 7, 8)$$

$$f_3 = \Sigma_m(1, 6, 15)$$

$$f = \Sigma_m(1, 6, 8, 15)$$

then  $f_2$  is:

[GATE 2008]

[1-Mark]

- (a)  $\Sigma_m(4, 6)$
- (b)  $\Sigma_m(4, 8)$
- (c)  $\Sigma_m(6, 8)$
- (d)  $\Sigma_m(4, 6, 8)$

- Q.83** If P, Q, R are Boolean variables, then

$$(P + \bar{Q})(P\bar{Q} + P.R)(\bar{P}.R + \bar{Q})$$

Simplifies to:

[GATE 2008]

[2-Marks]

- (a)  $P\bar{Q}$
- (b)  $P\bar{R}$
- (c)  $P\bar{Q} + R$
- (d)  $P\bar{R} + \bar{Q}$

- Q.84** What are the minimum number of 2-to-1 multiplexers required to generate a 2-input AND gate and a 2-input Ex-OR gate?

[EC-GATE 2009]

[2-Marks]

- (a) 1 and 1
- (b) 1 and 2
- (c) 1 and 3
- (d) 2 and 2

### Common Data For Questions 85 & 86:

Two products are sold from a vending machine, which has two push buttons  $P_1$  and  $P_2$ . When a button is pressed, the price of the corresponding product is displayed in a 7-segment display.

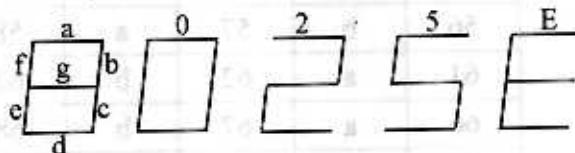
If no buttons are pressed, '0' is displayed, signifying 'Rs. 0'

If only  $P_1$  is pressed, '2' is displayed, signifying 'Rs. 2'

If only  $P_2$  is pressed, '5' is displayed, signifying 'Rs. 5'

If both  $P_1$  and  $P_2$  are pressed, 'E' is displayed, signifying 'Error'

The names of the segments in the 7-segment display, and the glow of the display for '0', '2', '5', and 'E' are shown below.



Consider

1. push buttons pressed/not pressed is equivalent to logic 1/0 respectively.
2. a segment glowing/not glowing in the display is equivalent to logic 1/0 respectively.

- Q.85** If segments a to g are considered as functions of  $P_1$  and  $P_2$ , then which of the following is correct? [EC-GATE 2009]

[2-Marks]

- (a)  $g = \bar{P}_1 + P_2$ ,  $d = c + e$
- (b)  $g = \bar{P}_1 + P_2$ ,  $e = b + c$
- (c)  $g = P_1 + P_2$ ,  $d = c + e$
- (d)  $g = P_1 + P_2$ ,  $e = b + c$

- Q.86** What are the minimum number of NOT gates and 2-input OR gates required to design the logic of the driver for this 7-segment display?

[EC-GATE 2009]

[2-Marks]

- (a) 3 NOT and 4 OR
- (b) 2 NOT and 3 OR
- (c) 1 NOT and 3 OR
- (d) 2 NOT and 4 OR

# ANSWER KEY

|    |   |    |   |    |   |    |   |    |   |
|----|---|----|---|----|---|----|---|----|---|
| 1  | b | 2  | a | 3  | a | 4  | d | 5  | c |
| 6  | a | 7  | d | 8  | c | 9  | c | 10 | d |
| 11 | b | 12 | b | 13 | d | 14 | c | 15 | d |
| 16 | a | 17 | b | 18 | d | 19 | a | 20 | a |
| 21 | d | 22 | a | 23 | a | 24 | b | 25 | d |
| 26 | c | 27 | b | 28 | c | 29 | d | 30 | a |
| 31 | b | 32 | b | 33 | b | 34 | d | 35 | c |
| 36 | d | 37 | d | 38 | b | 39 | d | 40 | b |
| 41 | a | 42 | a | 43 | b | 44 | b | 45 | b |
| 46 | d | 47 | b | 48 | c | 49 | d | 50 | b |
| 51 | c | 52 | b | 53 | a | 54 | b | 55 | a |
| 56 | b | 57 | a | 58 | c | 59 | b | 60 | b |
| 61 | a | 62 | b | 63 | b | 64 | d | 65 | c |
| 66 | a | 67 | b | 68 | b | 69 | c | 70 | c |
| 71 | b | 72 | a | 73 | b | 74 | a | 75 | a |
| 76 | b | 77 | c | 78 | c | 79 | b | 80 | b |
| 81 | c | 82 | c | 83 | a | 84 | b | 85 | c |
| 86 | b |    |   |    |   |    |   |    |   |

## SOLUTIONS

**S.1 (b)**

The expression of three input Ex-OR gate is

$$A \oplus B \oplus C = \bar{ABC} + \bar{AB}\bar{C} + \bar{A}\bar{B}\bar{C} + ABC$$

Hence, the given diagram shows the sum of expression for 3-input Ex-OR gate.

**S.2 (a)**

Multiplexers can implement full adder using proper control signals.

Neither half adder nor decoders can alone do this. Also AND and OR gates require NOT gate for implementation.

**S.3 (a)**

All the bits of the allgnd and addend are fed into the adder circuits simultaneously and the additions in each position are taking place at the same time. this circuit is known as parallel adder.

Hence for m-bit parallel adder we require M-full adders.

**S.4 (d)**

From the logic diagram, we have

$$\overline{(xy)} + (xy) = (\bar{x}\bar{y})(\bar{x}y)$$

$$\Rightarrow (x+y)(\bar{x}+\bar{y})$$

$$\Rightarrow xy + y\bar{x}$$

This is the equation of half adder.

Hence the given logic diagram is adder.

**S.5 (c)**

Combinational circuits are circuits in which the output at any time depends upon the combination of the input signals present at that instant only and doesn't depend upon past values.

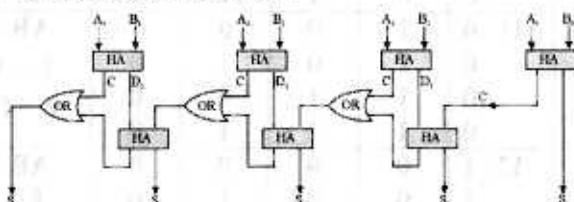
That's why, the combinational circuits are made up logic gates only.

**S.6 (a)**

The logic function for difference output in full subtractor is exactly the same as the output sum in full adder. Also the final borrow bit output of full subtractor is nearly similar to the final carry bit output of the full adder, with the only difference that the input variable is complemented in the full subtractor.

**S.7 (d)**

The circuit of 4-bit 4 full adder using half adders and OR gates is shown.

**S.8 (c)**

(a) Multiplexer → It is also called a data selector since it selects one of many inputs and stores the information to the output.

(b) De-multiplexer → A demultiplexer is a logic circuit that receives information on a single input and transmit the same information over one of several ( $2^n$ ) outputs lines.

(c) Encoder → An encoder is a combinational logic circuit that converts an active input signal into a coded output signal i.e., it converts decimal to binary.

Hence we have,

$$a = 3, b = 4, c = 2.$$

**S.9 (c)**

The necessary steps carried out to perform the operation of accessing either memory or I/O device contribute a machine cycle. In other words steps carried out to perform either a read or write operation constitute a machine cycle.

## (a) DS 8

**S.10 (d)**

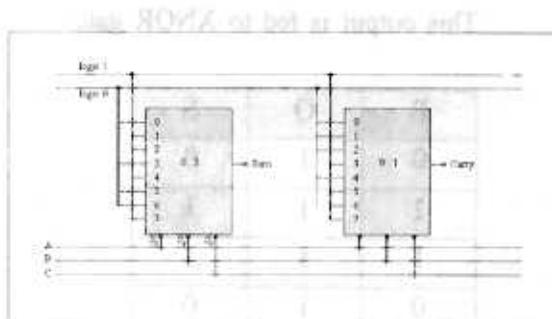
From the truth table of full adder the output equations in terms of decimal form

$$\text{Sum} = \Sigma m\{1, 2, 4, 7\}$$

$$\text{Carry} = \Sigma m\{3, 5, 6, 7\}$$

To implement using MUX

No. of outputs = No. of MUX

**S.11 (b)**

There are 1024 memory location  $1024 = 2^{10}$

Hence address bus width = 10 bits.

**S.12 (b)**

4 bit uses 4 FF

$$\text{Total delay } Nt_d = 4 \times 70 = 280 \times 10^{-9}$$

$$f = \frac{1}{280 \times 10^{-9}} = 3.6 \text{ MHz}$$

**S.14 (c)**

Initially  $J = K = 1$ , therefore the next output will be 0.

Now,  $J = 1$  and  $K = 0$ , therefore output will be 1.

Again  $J = K = 1$  and next output will be 0.

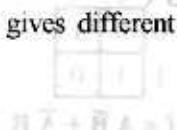
This cycle continues and the output will be of the form 0, 1, 0, 1, 0, 1, 0, ....

**S.16 (a)**

$$x \text{ NOR}(y \text{ NOR } z) \neq (x \text{ NOR } y) \text{ NOR } z.$$

**S.17 (b)**

For different combinations of select lines MUX gives different outputs.



| <b>S<sub>1</sub></b> | <b>S<sub>0</sub></b> | <b>P</b>           |
|----------------------|----------------------|--------------------|
| 0                    | 0                    | I <sub>0</sub> (0) |
| 0                    | 1                    | I <sub>1</sub> (1) |
| 1                    | 0                    | I <sub>2</sub> (1) |
| 1                    | 1                    | I <sub>3</sub> (0) |

This output is fed to XNOR gate.

| <b>P</b> | <b>Q</b> | <b>S</b> |
|----------|----------|----------|
| 0        | 1        | 0        |
| 1        | 1        | 1        |
| 1        | 1        | 1        |
| 0        | 1        | 0        |

$$\text{So, } S(A, B) = \Sigma m(1, 2)$$

### S.18 (d)

The expression can be put into sum-of-products form by multiplying all the terms.

$$\therefore x = \bar{A}A\bar{D} + \bar{A}B\bar{D} + \bar{A}D\bar{D} + BA\bar{D} + BD\bar{D} + BB\bar{D}$$

$$\text{As, } \bar{A}A = 0, BB = B \text{ and } D\bar{D} = 0$$

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

$$= \bar{B}D(\bar{A} + A + 1)$$

The term inside the parentheses is 1.

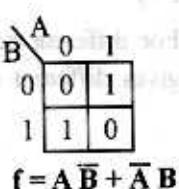
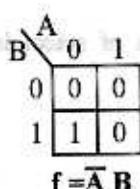
$$\therefore x = \bar{B}D$$

### S.19 (a)

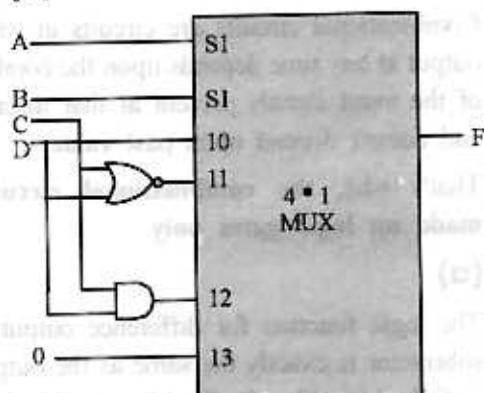
Truth-table for half-subtractor:

| <b>A</b> | <b>B</b> | <b>X</b> | <b>D</b> |
|----------|----------|----------|----------|
| 0        | 0        | 0        | 0        |
| 0        | 1        | 1        | 1        |
| 1        | 0        | 0        | 1        |
| 1        | 1        | 0        | 0        |

Drawing K-maps:



### S.20 (a)



Truth Table

| <b>A</b> | <b>B</b> | <b>C</b> | <b>D</b> | <b>F</b> |                           |
|----------|----------|----------|----------|----------|---------------------------|
| 10 0     | 0        | 0        | 0        | 0        | AB = 00<br>F = D          |
| 0        | 0        | 0        | 1        | 1        |                           |
| 0        | 0        | 1        | 0        | 0        |                           |
| 0        | 0        | 1        | 1        | 1        |                           |
| 11 0     | 1        | 0        | 0        | 1        | AB = 01<br>F = $\bar{C}D$ |
| 0        | 1        | 0        | 1        | 0        |                           |
| 0        | 1        | 1        | 0        | 0        | F = C + D                 |
| 0        | 1        | 1        | 1        | 0        |                           |
| 12 1     | 0        | 0        | 0        | 0        | AB = 10<br>F = CD         |
| 1        | 0        | 0        | 1        | 0        |                           |
| 1        | 0        | 1        | 0        | 0        |                           |
| 1        | 0        | 1        | 1        | 1        |                           |
| 13 1     | 1        | 0        | 0        | 1        | AB = 11<br>F = 1          |
| 1        | 1        | 0        | 1        | 1        |                           |
| 1        | 1        | 1        | 0        | 1        |                           |
| 1        | 1        | 1        | 1        | 1        |                           |

### S.21 (d)

(a) Shift registers are used to transfer the contents of one register to another register. It may have with serial and parallel input and outputs.

(b) Multiplexer is also called a data selector since it selects one of many inputs and stores the information to the output.

(c) An encoder is a combinational logic circuit that converts an active input signal into a coded output signal i.e., it converts decimal to binary.

Hence we have,

$$a = 3, b = 4, c = 1.$$

**S.22 (a)**

Let  $z = 0$

Then  $f = w'x' + wy'$

then put

$z = 0$  in all options. Since MUX is enable.

**S.23 (a)**

$$f = \overline{\sum m(0, 3, 6)} = \sum m(1, 2, 4, 5, 7)$$

**S.24 (b)**

The truth table of subtractor is

| Input |   |   | Output         |                |
|-------|---|---|----------------|----------------|
| X     | Y | Z | Difference (D) | Borrow Out (B) |
| 0     | 0 | 0 | 0              | 0              |
| 0     | 0 | 1 | 1              | 1              |
| 0     | 1 | 0 | 1              | 1              |
| 0     | 1 | 1 | 0              | 1              |
| 1     | 0 | 0 | 1              | 0              |
| 1     | 0 | 1 | 0              | 0              |
| 1     | 1 | 0 | 0              | 0              |
| 1     | 1 | 1 | 1              | 1              |

The output can be written as

$$\begin{aligned} D &= \bar{X}\bar{Y}Z + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XYZ \\ &= XYZ + \bar{X}\bar{Y}Z + X\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} \end{aligned}$$

**S.25 (d)**

The output from the upper first level multiplexer is  $f_a$  and from the lower first level multiplexer is  $f_b$

$$f_a = \bar{W}X + W\bar{X}$$

$$f_b = \bar{W}X + WX = X$$

$$\begin{aligned} f &= f_a\bar{Y}Z + f_bY\bar{Z} + YZ = (\bar{W}X + W\bar{X})\bar{Y}Z + XY\bar{Z} + YZ \\ &= \bar{W}XYZ + WXY\bar{Z} + XY + YZ \end{aligned}$$

**S.26 (c)**

$$\text{Here, } Z = (I+A)(J+\bar{A})$$

$$= I\bar{A} + AJ + IJ$$

Since we need  $\bar{A}$  at output, we take  $I = 1$ .

$$\text{Therefore } Z = \bar{A} + AJ + J$$

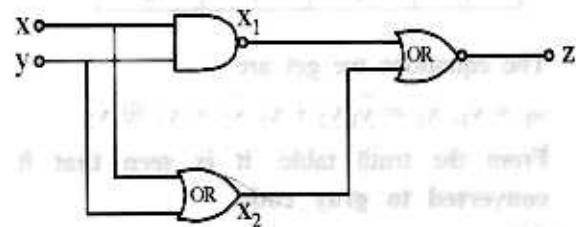
$$= \bar{A} + J$$

Therefore we take  $J = B$ .

**S.27 (b)**

| x | y | x <sub>1</sub> | x <sub>2</sub> | z |
|---|---|----------------|----------------|---|
| 0 | 0 | 1              | 0              | 1 |
| 0 | 1 | 1              | 1              | 1 |
| 1 | 0 | 1              | 1              | 1 |
| 1 | 1 | 0              | 1              | 1 |

| x | y | z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**S.28 (c)**

In terms of Boolean operations,

Output of 1 is  $x_0 x_1$

Output of 2 is  $(x_0 x_1 + x_2) x_3 = x_0 x_1 x_3 + x_2 x_3$

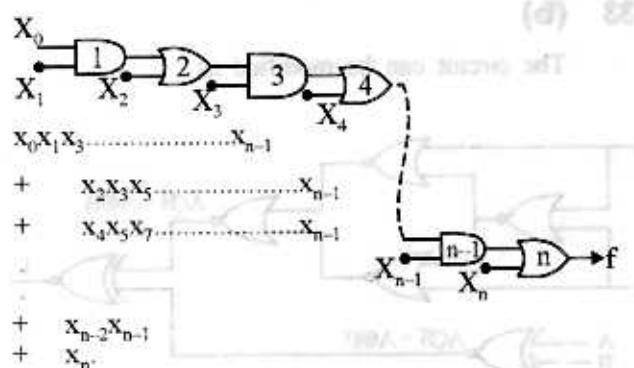
Output of 3 is  $(x_0 x_1 + x_2) x_3 = x_0 x_1 x_3 + x_2 x_3$

Output of 4 is  $x_0 x_1 x_3 + x_2 x_3 + x_4$

Output of 5 would be  $x_0 x_1 x_3 x_5 + x_2 x_3 x_5 + x_4 x_5$

Output of 6 would be  $x_0 x_1 x_3 x_5 + x_2 x_3 x_5 + x_4 x_5 + x_6$

Thus for n gates connected as shown, the output would be

**S.29 (d)**

The K-map reveals W, X, Y, Z are the input lines to the device and D<sub>0</sub> to D<sub>9</sub> are the output lines which get activated depending upon the input combinations. As seen from K-map for W, X, Y, Z from 0000 to 1001 we have D<sub>0</sub> and D<sub>9</sub>

### 2.3.20

### DIGITAL LOGIC

outputs activated respectively. This operation is performed by BCD to Decimal decoder. Also it is called as 1 of 10 decoder, as only 1 of 10 output line is high.

#### S.30 (a)

The truth table of circuit is

| $y_1$ | $y_2$ | $x_1$ | $x_2$ |
|-------|-------|-------|-------|
| 0     | 0     | 0     | 0     |
| 0     | 1     | 0     | 1     |
| 1     | 0     | 1     | 1     |
| 1     | 1     | 1     | 0     |

The equations we get are

$$x_1 = y_1, x_2 = \bar{y}_1 y_2 + y_1 \bar{y}_2 = y_1 \oplus y_2$$

From the truth table, it is seen that it is converted to gray code.

#### S.31 (b)

As both lines of A are high so its output irrespective of control input A and now second multiplexer has both high inputs so its output is independent of control input B. Hence, output is independent of both A and B.

#### S.32 (b)

Here the network is implemented using two-level AND-OR logic.

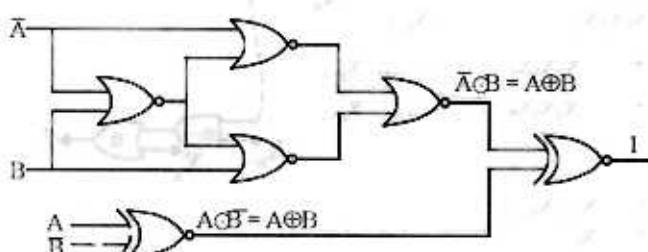
We know that delay is given as  $2N + 2$ .

Here N = No. of bit

then Delay =  $2 \times 2 + 2 = 6$  time unit

#### S.33 (b)

The circuit can be modified as



Inputs to the final EX-NOR gate are same ( $A \oplus B$ ). Hence output will always be 1.

#### S.34 (d)

The truth table for full-subtractor is

| X | Y | Z | Difference D | Borrow B |
|---|---|---|--------------|----------|
| 0 | 0 | 0 | 0            | 0        |
| 0 | 0 | 1 | 1            | 1        |
| 0 | 1 | 0 | 1            | 1        |
| 0 | 1 | 1 | 0            | 1        |
| 1 | 0 | 0 | 1            | 0        |
| 1 | 0 | 1 | 0            | 0        |
| 1 | 1 | 0 | 0            | 0        |
| 1 | 1 | 1 | 1            | 1        |

$$B = 001 + 010 + 011 + 111$$

$$= \bar{X}YZ + \bar{X}Y\bar{Z} + \bar{X}YZ + XYZ$$

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

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

#### S.35 (c)

Now the output Z of the box at the right is

$$Z = R \bar{Q} + PR + \bar{Q}P$$

But R = 0, therefore Z =  $\bar{Q}P$

Therefore, output Z of the next box (2<sup>nd</sup> from right) is

$$Z = R \bar{Q} + PR + \bar{Q}P$$

$$\text{But here } R = \bar{Q}P, \text{ therefore } Z = \bar{Q}P \bar{Q} + \bar{Q}PP + \bar{Q}P = \bar{Q}P$$

Similarly output Z of next boxes (3<sup>rd</sup> and 4<sup>th</sup> from right) will also be  $\bar{Q}P$ .

Now, of the four options the output  $\bar{Q}P$  can only be the borrow output of Q - P.

#### S.37 (d)

Truth-table is as shown below

| $S_1(P)$ | $S_2(Q)$ | $C_{in}$ | $S$ |
|----------|----------|----------|-----|
| 0        | 0        | 0        | 0   |
| 0        | 0        | 1        | 1   |
| 0        | 1        | 0        | 1   |
| 0        | 1        | 1        | 0   |
| 1        | 0        | 0        | 1   |
| 1        | 0        | 1        | 0   |
| 1        | 1        | 0        | 0   |
| 1        | 1        | 1        | 1   |

Now, Reading the truth-table as

| $S_1$ | $S_2$ | $S$            |       |
|-------|-------|----------------|-------|
| 0     | 0     | $C_{in}$       | $I_0$ |
| 0     | 1     | $\bar{C}_{in}$ | $I_1$ |
| 1     | 0     | $\bar{C}_{in}$ | $I_2$ |
| 1     | 1     | $C_{in}$       | $I_3$ |

Thus,

$$I_0 = I_3 = C_{in}$$

$$\bar{I}_1 = \bar{I}_2 = C_{in}$$

$$\text{or } I_1 = I_2 = \bar{C}_{in}$$

### S.38 (b)

| A | B | $Q_{n+1}$ | J |
|---|---|-----------|---|
| 0 | 0 | $Q_n$     | 0 |
| 0 | 1 | 1         | 1 |
| 1 | 0 | $Q_n$     | 0 |
| 1 | 1 | 0         | x |

$$J = \bar{A}B + AB$$

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

### S.39 (d)

Bubbled NAND gate is equivalent to OR gate.

$$\therefore f_1 = Y_1 + Y_2 + Y_6 + Y_7 = \sum m(1, 2, 6, 7)$$

$$\& f_2 = Y_7 + Y_9 + Y_{12} + Y_{14} = \sum m(7, 9, 12, 14)$$

### S.40 (b)

The input is

$$D_1 C_1 B_1 A_1 D_o C_o B_o A_o = 00101001$$

$$\text{The binary output } B_o = A_o = 1$$

$$\text{The inputs of IC1 are } E = B_1 = 1$$

$$D = A_1 = 0$$

$$C = D_o = 1$$

$$B = C_o = 0$$

$$A = B_o = 0$$

$$\begin{array}{l}
 10100 \\
 \swarrow \quad \searrow \\
 \rightarrow 0000100 \\
 + 0001010 \\
 \hline 0001110
 \end{array}$$

The outputs of IC1 are

$$Y_5 = 0$$

$$Y_4 = 1$$

$$Y_3 = 1$$

$$Y_2 = 1$$

$$Y_1 = 0$$

### S.41 (a)

The binary outputs  $B_1$  and  $B_2$  are,

$$B_1 = Y_1 = 0$$

$$B_2 = Y_2 = 1$$

The inputs of IC2 are  $E = D_1 = 0$

$$D = C_1 = 0$$

$$C = Y_5 \text{ of IC1} = 0$$

$$B = Y_4 \text{ of IC1} = 1$$

$$A = Y_3 \text{ of IC1} = 1$$

$$\begin{array}{r}
 00011 \\
 \swarrow \quad \searrow \\
 \rightarrow 0000001 \\
 + 0000010 \\
 \hline 0000011
 \end{array}$$

The inputs of IC2 are  $Y_5 = 0$

$$Y_4 = 0$$

$$Y_3 = 0$$

$$Y_2 = 1$$

$$Y_1 = 1$$

The binary outputs  $B_3$ ,  $B_4$ ,  $B_5$  and  $B_6$  are

$$B_3 = Y_1 = 1$$

$$B_4 = Y_2 = 1$$

$$B_5 = Y_3 = 0$$

$$B_6 = Y_4 = 0$$

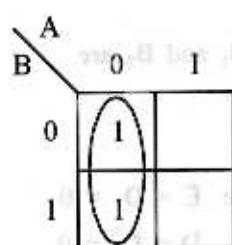
The full binary output is 0011101

### S.42 (a)

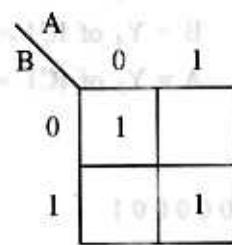
The state table is as follows:

| A | B | J | K | $Q_n$ | $Q_{n+1}$ |
|---|---|---|---|-------|-----------|
| 0 | 0 | 1 | 1 | $Q_n$ | $Q_n$     |
| 0 | 1 | 1 | 0 | X     | 1         |
| 1 | 0 | 0 | 0 | $Q_n$ | $Q_n$     |
| 1 | 1 | 0 | 1 | X     | 0         |

The k-map for J and K are:

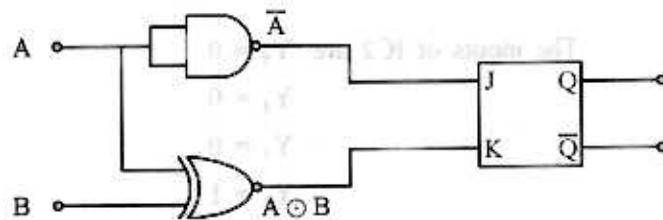


$$J = \bar{A}$$



$$K = AB + \bar{A}\bar{B} = A \oplus B$$

The circuit implementation is:



### S.43 (b)

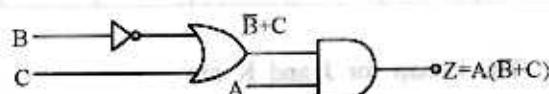
The output of the logic circuit given is,

$$Z = ABC + A\bar{B}(\bar{A}\bar{C})$$

The above expression can be simplified as follows:

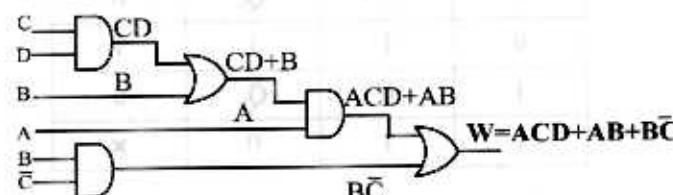
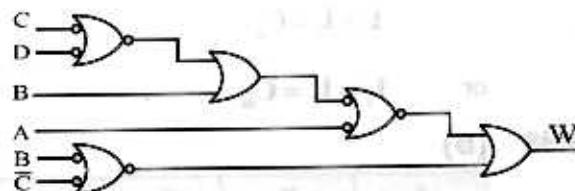
$$\begin{aligned} Z &= ABC + A\bar{B}(A + \bar{C}) \\ &= ABC + A\bar{B}(A + C) \\ &= ABC + A\bar{B}A + A\bar{B}C \\ &= ABC + A\bar{B} + A\bar{B}C \\ &= AC(B + \bar{B}) + A\bar{B} \\ &= AC + A\bar{B} \\ &= A(\bar{B} + C) \end{aligned}$$

The circuit implementation will be,

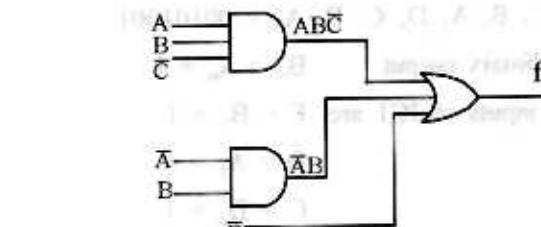
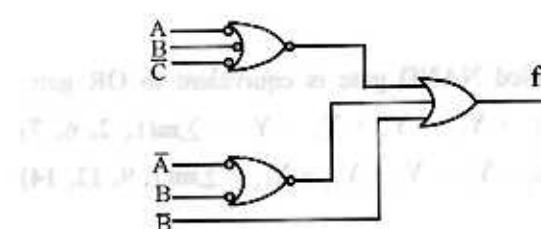


(d) 5b.2

### S.44 (b)



### S.45 (b)

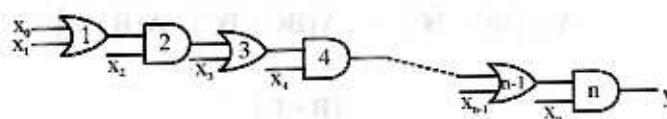


$$\begin{aligned} &= AB\bar{C} + \bar{A}B + \bar{B} \\ &= AB\bar{C} + \bar{A} + \bar{B} \\ &= \bar{A} + \bar{B} + A\bar{C} \\ &= \bar{A} + \bar{B} + \bar{C} \\ &= \overline{ABC} \end{aligned}$$

$\therefore f = \overline{ABC}$  can be realized using a NAND gate.

### S.46 (d)

Bubbled NAND is equivalent to OR gate and NOR gate is equivalent to AND gate. Therefore the equivalent circuit is :



$n$  is even

Take  $n = 4$

$\therefore$  Expression becomes

$$((x_0 + x_1)x_2) + x_3 \cdot x_4$$

$$x_0x_2x_4 + x_1x_2x_4 + x_3x_4 \quad (I)$$

Options (a) and (c) are ruled out.

Generalizing, we observe that  $x_5$  term will not come with the term starting with  $x_3$ .

In every term, maximum one odd term can be there.

So option (b) is ruled out.

### S.47 (b)

Drawing a state table : (Let input be X)

| Present State | Next Step |       | Output State |       |
|---------------|-----------|-------|--------------|-------|
|               | X = 0     | X = 1 | X = 0        | X = 1 |
| a             | c         | b     | 0            | 0     |
| b             | d(f)      | c     | 0            | 0     |
| c             | g(e)      | d(f)  | 1            | 1     |
| d             | e         | f     | 1            | 0     |
| e             | f         | a     | 0            | 1     |
| f             | g(e)      | f     | 1            | 0     |
| g             | f         | a     | 0            | 1     |

State e and g are equivalent because the next states and outputs are same. We can eliminate any one of them, say 'g'. Thus 'g' is replaced by 'e' wherever it occurs. Now states d and f are equivalent. Thus one of them say d, can be eliminated. Thus the reduced state table :-

| Present State | Next Step |       | Output State |       |
|---------------|-----------|-------|--------------|-------|
|               | X = 0     | X = 1 | X = 0        | X = 1 |
| a             | c         | b     | 0            | 0     |
| b             | f         | c     | 0            | 0     |
| c             | e         | f     | 1            | 1     |
| e             | f         | a     | 0            | 1     |
| f             | e         | f     | 1            | 0     |

### S.48 (c)

From trith table of 3 to 8 decoder.

$$D_0 = \bar{ABC}, D_1 = \bar{ABC}, D_2 = \bar{ABC}$$

$$D_3 = \bar{ABC}, D_4 = \bar{ABC}, D_2 = \bar{ABC}$$

$$D_6 = ABC, D_7 = ABC$$

|   |   | BC |    |    |    |
|---|---|----|----|----|----|
|   |   | 00 | 01 | 11 | 10 |
| A | 1 | 1  | 1  |    |    |
|   |   |    |    | 1  | 1  |

$$X = \sum_m(1, 3, 6, 7)$$

$$X = \bar{AC} + AB$$

### S.49 (d)

From trith table of 3 to 8 decoder

$$D_0 = \bar{ABC}, D_1 = \bar{ABC}, D_2 = \bar{ABC}$$

$$D_3 = \bar{ABC}, D_4 = \bar{ABC}, D_2 = \bar{ABC}$$

$$D_6 = ABC, D_7 = ABC$$

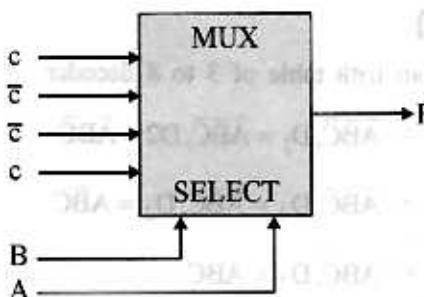
|   |   | BC |    |    |    |
|---|---|----|----|----|----|
|   |   | 00 | 01 | 11 | 10 |
| A | 0 | 1  | 1  |    |    |
|   |   |    |    | 1  | 1  |

$$Y = \sum_m(2, 3, 5, 6)$$

$$Y = \bar{AB} + BC + \bar{ABC}$$

### S.50 (b)

$$\begin{aligned} F &= \bar{ABC} + \bar{ABC} + \bar{ABC} + ABC \\ &= \bar{A}(B \oplus C) + A(B \oplus C) \\ &= A \oplus B \oplus C \end{aligned}$$



S.51 (c)

Since half address can implement product as well as sum terms, therefore all digital circuit can be realized with half-adder.

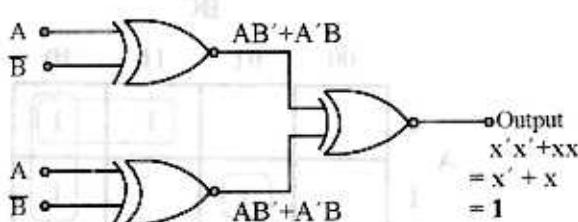
S.52 (b)

$$\begin{aligned} F &= \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}\bar{C} + ABC \\ &= \bar{B}(\bar{A}C + A\bar{C}) + B(\bar{A}C + A\bar{C}) \\ &= \bar{B}(A \oplus C) + B(A \oplus C) \\ &= (\bar{A}C + A\bar{C})(\bar{B} + B) = A \oplus C \end{aligned}$$

S.53 (a)

False; in look ahead carry adder the sum bit of 'n' position is generated in accordance of the carry generated by (n-1)th bit ahead of the nth bit addition.

S.54 (b)



Where  $x = AB' + A'B$ .

S.55 (a)

|   |   |   |   |
|---|---|---|---|
| A | B | 0 | 1 |
| 0 | C | C | C |
| 1 | C | C | C |

| A | B | C |
|---|---|---|
| 0 | 0 | C |
| 0 | 1 | C |
| 1 | 0 | C |
| 1 | 1 | C |

$$F = \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}\bar{C} + ABC$$

$$A \oplus (\bar{B}C + \bar{B}C) = \bar{A}(\bar{B}C + \bar{B}C) + A(\bar{B} + C)$$

$$(B + \bar{C})$$

$$= \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}\bar{C} + ABC$$

S.56 (b)

It depends on  $\log N$ .

S.57 (a)

At the output we require a zero so we substitute the options in the circuit and check.

S.58 (c)

4 select lines can select  $2^4 = 16$  line i.e. 16 : 1 MUX.

S.59 (b)

The one half-adder can add the least significant bit of the two numbers. Full adders are required to add the remaining 15 bits as they all involve adding carries.

S.60 (b)

| A | B | D(difference) | X (borrow) |
|---|---|---------------|------------|
| 0 | 0 | 0             | 0          |
| 0 | 1 | 1             | 1          |
| 1 | 0 | 1             | 0          |
| 1 | 1 | 0             | 0          |

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

$$X = \bar{A}B$$

S.61 (a)

As the output of AND is  $X = 1$ , then all input of this AND must be 1. Thus

$$\bar{A}B + A\bar{B} = 1 \quad \dots(1)$$

$$\bar{B}\bar{C} + BC = 1 \quad \dots(2)$$

$$C = 1 \quad \dots(3)$$

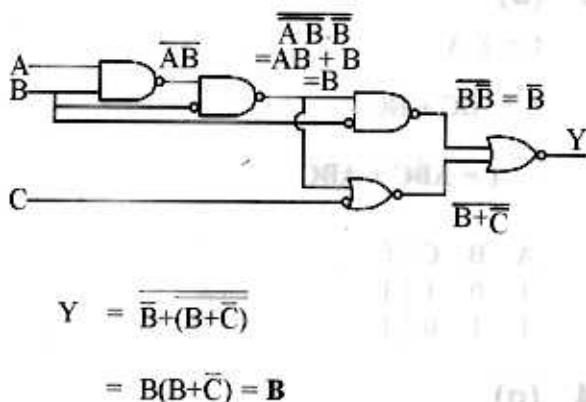
From (2) and (3), if  $C = 1$ , then  $B = 1$

If  $B = 1$ , then from (1)  $A = 0$

Thus  $A = 0$ ,  $B = 1$  and  $C = 1$

S.62 (b)

The circuit is as shown below:

**S.63 (b)**

| BA |    |    |    |
|----|----|----|----|
| C  | 00 | 01 | 11 |
| 0  | 1  |    | 1  |
| 1  |    | 1  | 1  |

$$Y = I_0 + I_3 + I_5 + I_6$$

$$Y = \overline{C}\overline{B}\overline{A} + \overline{C}AB + C\overline{B}A + CBA$$

$$Y = \overline{C}(\overline{B}\overline{A} + AB) + C(\overline{B}A + \overline{A}B)$$

$$Y = \overline{C}(A \oplus B) + C(A \oplus B)$$

**S.64 (d)**

$$\text{Output of 1st XOR} = \overline{X} \cdot 1 + X \cdot \overline{1} = \overline{X}$$

$$\text{Output of 2nd XOR} = \overline{X} \cdot \overline{X} + X \cdot X = 1$$

So after 4, 6, 8, .... 20 XOR output will be 1.

**S.65 (c)**

$$f(A, B) = A' + B$$

$$f(x+y, y) = \overline{x+y} + y = M$$

$$\begin{aligned} f(M, z) &= \overline{x+y} + y + z \\ &= (\overline{x+y})\overline{y} + z \\ &= x\overline{y} + z \end{aligned}$$

**S.66 (a)**

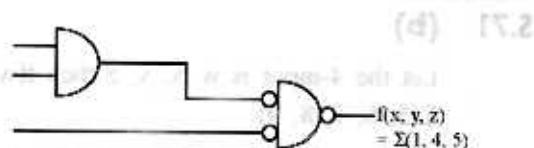
$$f_1 = \Sigma(0, 1, 3, 5)$$

$$f_2 = \Sigma(6, 7)$$

$$f = f_3 + f' = f_3 + 0 = f_3$$

$$\therefore f_3 = f = \Sigma(1, 4, 5)$$

(d) 2.3.25

**S.67 (b)**

n variables are implemented by  $2^{n-1} : 1$  MUX.

**S.68 (b)**

$$\text{No. of MUX} = \frac{4}{2} = 2$$

$$\text{and } \frac{2}{2} = 1$$

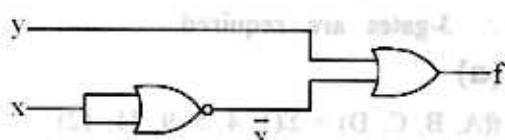
$$\therefore \text{Total} = 3$$

Thus 3 multiplexer are required.

**S.69 (c)**

| x | y | f |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Since complements are not available.



$$f(x, y) = \overline{x} + y$$

$$\therefore 2 \text{ units}$$

Here complements are not available so to get x we use NOR gate. Thus desired circuit require 1 unit OR and 1 unit NOR gate giving total cost 2 units.

**S.70 (c)**

For the TTL family if terminal is floating, then it is at logic 1.

$$\text{Thus } Y = (AB+1)$$

$$= AB \cdot 0 = 0$$

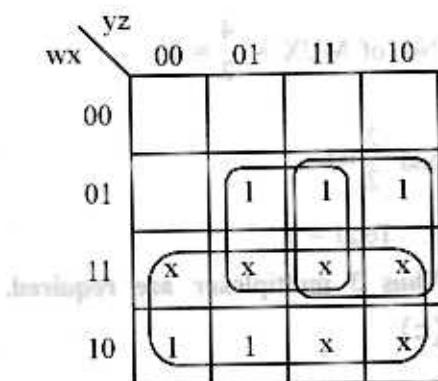
## S.71 (b)

Let the 4-input is  $w, x, y, z$  then  $f(w, x, y, z) = \Sigma(5, 6, 7, 8, 9)$ .

The decimal value of 10, 11, 12, 13, 14, 15 and don't care conditions. So,

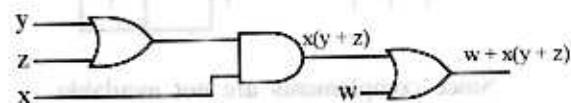
$$f(w, x, y, z) = \Sigma_0(10, 11, 12, 13, 14, 15)$$

K-map:



$$f(w, x, y, z) = w + xy + xz$$

$$f(w, x, y, z) = w + x(y + z)$$

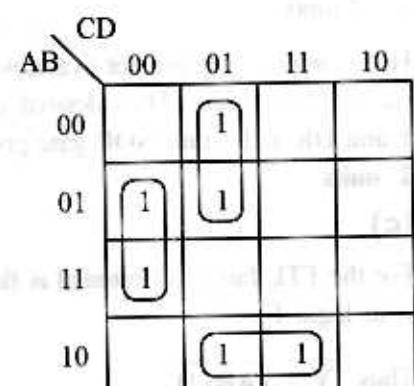


$\therefore$  3-gates are required.

## S.72 (a)

$$f(A, B, C, D) = \Sigma(1, 4, 5, 9, 11, 12)$$

The K-map for  $f$  is:



$$BC'D' + A'C'D + AB'D$$

## S.73 (b)

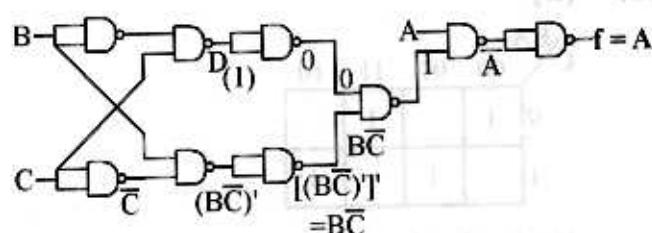
$$f = EA$$

$$E = \bar{B}C + B\bar{C}$$

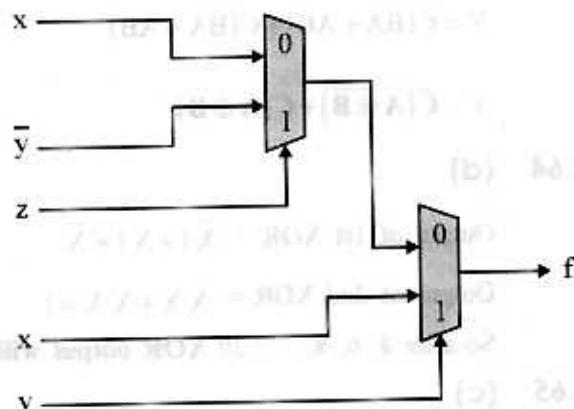
$$\therefore f = \bar{A}\bar{B}C + A\bar{B}\bar{C}$$

| A | B | C | f |
|---|---|---|---|
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |

## S.74 (a)



## S.75 (a)



Let output of 1<sup>st</sup> uppermost Mux is  $f_1$ .

$$f_1 = xz' + y'z$$

The output of 2<sup>nd</sup> Mux is:

$$\begin{aligned}
 f &= (xz' + y'z)y' + xy \\
 &= xy'z' + y'y'z + xy \\
 &= xy'z' + y'z + xy \quad [\because a \cdot a' = 0] \\
 &= y'z + xy'z' + xy(z + z') \quad [\because a + a' = 1] \\
 &= y'z + xy'z' + xyz + xyz' \\
 &= y'z + xy'z' + xyz' + xyz' + xyz \\
 &= y'z + xz'(y' + y) + xy(z' + z) \\
 &= y'z + xz' + xy \\
 &= xz' + xy + y'z.
 \end{aligned}$$

**S.76 (b)**

A 4-bit adder with  $S_3, S_2, S_1, S_0$  and  $C_4$  as its output require 2-1 bit adder and a single one-bit adder require 5 AND gates 2 OR gates in two level logic. So, the required gates are 10 AND gate and 4-OR gate.

**S.77 (c)**

To implement  $n$ -variable function we require data sector with  $n-1$  select inputs &  $2^{n-1}$  data inputs. So minimum size of MUX needed is  $2^{n-1}$  line to 1 line.

**S.78 (c)**

For construction of a  $6 \times 64$  decoder by using  $3 \times 8$  line decoder:

$$\frac{64}{8} = 8+1 \rightarrow \text{extra decoder for combining the result.}$$

$$3 \times 8 \xrightarrow{8+1} 6 \times 6$$

**S.79 (b)**

Let the O/P of first MUX be Y

$$\text{So } Y = \bar{A}B + A\bar{B} = A \oplus B$$

$$X = \bar{Y}C + Y\bar{C} = Y \oplus C$$

$$\text{So } X = A \oplus B \oplus C$$

$$= \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

**S.80 (b)**

$$Z = I_0\bar{R}\bar{S} + I_1\bar{R}S + I_2R\bar{S} + I_3RS$$

$$= (P + \bar{Q})\bar{R}\bar{S} + P\bar{R}S + PQ\bar{R}\bar{S} + PRS$$

$$= P\bar{R}\bar{S} + Q\bar{R}\bar{S} + P\bar{R}S + PQ\bar{R}\bar{S} + PRS$$

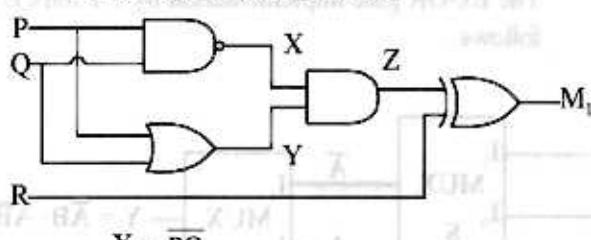
The k-map is as shown below

| PQ \ RS | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00      | 1  |    |    |    |
| 01      |    |    |    |    |
| 11      | 1  | 1  | 1  | 1  |
| 10      | 1  | 1  | 1  |    |

$$Z = PQ + P\bar{Q}S + Q\bar{R}\bar{S}$$

**S.81 (c)**

The circuit is shown below:



$$X = \bar{P}Q$$

$$Y = (P + Q)$$

$$\text{So, } Z = \bar{P}\bar{Q}(P + Q)$$

$$= (\bar{P} + \bar{Q})(P + Q)$$

$$= \bar{P}Q + P\bar{Q} = P \oplus Q$$

and,

$$M_1 = Z \oplus R$$

$$= (P \oplus Q) \oplus R$$

**S.82 (c)**

$f_1$  and  $f_2$  are connected with AND gates. So when  $f_2 = \Sigma_m(6, 8)$  then f is true  $\Sigma_m(1, 6, 8, 15)$ .

**S.83 (a)**

$$(P + Q')(PQ' + PR)(P'R' + Q')$$

$$= (PQ' + PR + PQ' + PRQ')(P'R' + Q')$$

$$= (PQ' + PR + PRQ')(P'R' + Q')$$

$$= (PQ' + PR(1 + Q'))(P'R' + Q')$$

$$= (PQ' + PR)(P'R' + Q')$$

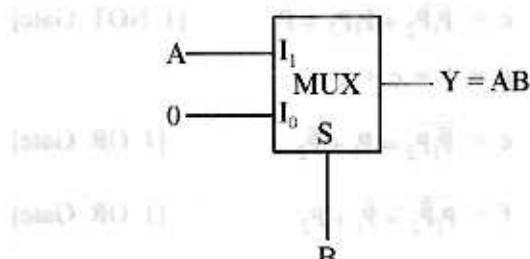
$$= PQ' + PRQ'$$

$$= PQ'(1 + R)$$

$$= PQ'$$

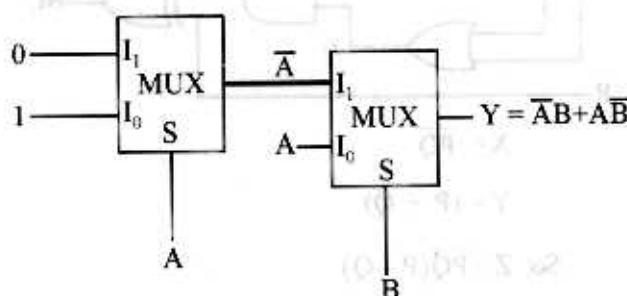
**S.84 (b)**

The AND gate implementation by 2-1 mux is as follows:



$$Y = \bar{B}I_0 + BI_1 = AB$$

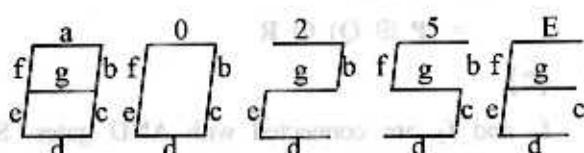
The Ex-OR gate implementation by 2:1 mux is as follows:



$$Y = \bar{B}I_0 + BI_1 = A\bar{B} + B\bar{A}$$

### S.85 (c)

The given situation is as follows:



$$P_1 = 0, P_2 = 0 \quad P_1 = 1 \quad P_2 = 1 \quad P_1 = 1, P_2 = 1$$

The truth table is as shown below:

| P <sub>1</sub> | P <sub>2</sub> | a | b | c | d | e | f | g |
|----------------|----------------|---|---|---|---|---|---|---|
| 0              | 0              | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0              | 1              | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1              | 0              | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1              | 1              | 1 | 0 | 0 | 1 | 1 | 1 | 1 |

From truth table we can write

$$a = 1$$

$$b = \bar{P}_1\bar{P}_2 + P_1\bar{P}_2 = \bar{P}_2 \quad [1 \text{ NOT Gate}]$$

$$c = \bar{P}_1\bar{P}_2 + \bar{P}_1P_2 = \bar{P}_1 \quad [1 \text{ NOT Gate}]$$

$$d = 1 = c + e$$

$$\text{and } c = \bar{P}_1P_2 = P_1 + \bar{P}_2 \quad [1 \text{ OR Gate}]$$

$$f = \bar{P}_1\bar{P}_2 = \bar{P}_1 + P_2 \quad [1 \text{ OR Gate}]$$

$$g = \bar{P}_1\bar{P}_2 = P_1 + P_2$$

[1 OR Gate]

Thus we have  $g = P_1 + P_2$  and  $d = 1 = c + e$ . It may be observed easily from figure that

Led g does not glow only when both  $P_1$  and  $P_2$  are 0. Thus

$$g = P_1 + P_2$$

LED d is 1 in all condition and also it depends on  $d = c + e$ .

### S.86 (b)

As shown in previous solution 2-NOT gates and 3-OR gates are required.

# 2.4

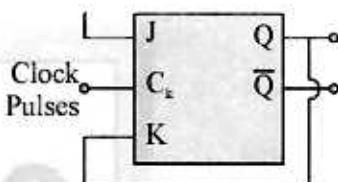
## SEQUENTIAL CIRCUITS

### LEVEL-1

- Q.1** S-R flip-flop can be converted to T-type flip-flop. If  
 (a) S and R are connected to Q and  $\bar{Q}$  respectively.  
 (b) S is connected to Q  
 (c) R is connected to Q  
 (d) Both S and R are shortened
- Q.2** The main difference between JK and RS flip-flop is that  
 (a) JK flip does not need a clock pulse  
 (b) JK flip-flop accepts both inputs as 1  
 (c) there is a feedback in JK flip-flop  
 (d) JK flip-flop is acronym of Junction cathode multivibrator
- Q.3** T flip-flop is commonly used as  
 (a) a digital counter and frequency divider  
 (b) a digital counter only  
 (c) a delay switch  
 (d) none of these
- Q.4** Which of the following conditions must be met to avoid race around problem?  
 (a)  $T > \Delta t > t_p$   
 (b)  $\Delta t < t_p < T$   
 (c)  $2t_p < \Delta t < T$   
 (d) none of these

- Q.5** In clocked SR flip flop if Preset and Clear are low signals applied then \_\_\_\_\_  
 (1) output will not be complementary of each other  
 (2) output will enter infinite loop under presence of clock pulse  
 (3) output will be an uncertain state  
 (a) 1 - False, 2 - False, 3 - True  
 (b) 1 - True, 2 - True, 3 - True  
 (c) 1 - True, 2 - False, 3 - True  
 (d) 1 - False, 2 - True, 3 - True
- Q.6** The difference between the output states of J-K flip flop and S-R flip flop is  
 (a) The S-R flip-flop has invalid state  
 (b) The J-K flip-flop has invalid state  
 (c) The S-R flip-flop has race-around condition  
 (d) The J-K flip-flop has race-around condition
- Q.7** A switch-tail ring counter is made by using a single D-FF. The resulting circuit is  
 (a) SR flip-flop  
 (b) JK flip-flop  
 (c) T - FF  
 (d) D - FF

- Q.8** In figure, assume that initially  $Q = 1$ . With clock pulses being given, the subsequent states of  $Q$  will be?



- (a) 1, 0, 1, 0, 1, 0, 1, ....
- (b) 0, 1, 0, 1, 0, 1, 0, ....
- (c) 0, 0, 1, 0, 0, 1, 0, ....
- (d) 1, 1, 0, 1, 1, 0, 1, ....

- Q.9** In a sequential circuit, the outputs at any instant of time depends
- (a) only on the inputs present at that instant of time
  - (b) on past outputs as well as present inputs
  - (c) only on the past inputs
  - (d) only on the present outputs

- Q.10** The S and R inputs of S-R flip flop are called synchronous inputs because
- (a) When  $S = 1$ ,  $\bar{Q} = 0$
  - (b) When  $S = 1$ ,  $Q = 0$
  - (c) The data on these inputs are transferred on the output
  - (d) The data is transferred to output only when clock signal is applied to it.

- Q.11** For a divide by 4 counter, the preset input P is equal to
- (a) 4
  - (b) 12
  - (c) 8
  - (d) 16

- Q.12** Suppose input to J-K flip flop is  $J = 0$ ,  $K = 1$  with applied clock pulse of 20 ms and the propagation delay time of 2 nsec. Then the operation performed is
- (a) flip flop race around condition
  - (b) flip flop toggle
  - (c) flip flop normal operation
  - (d) Both (a) and (b)

- Q.13** In general block diagram of sequential logic circuit, the clock is given to
- (a) Time delay device
  - (b) Output logic
  - (c) Memory element
  - (d) Both (a) and (c)

## LEVEL-2

- Q.14** Match the following:

| Flip-flop          | Clock pulse         |
|--------------------|---------------------|
| (i) S-R flip-flop  | (1) Edge triggered  |
| (ii) J-K flip-flop | (2) Level triggered |
| (iii) D flip-flop  |                     |
| (iv) T flip-flop   |                     |

Choose the correct option:

- (a) (i)-1, (ii)-1, (iii)-2, (iv)-2,
- (b) (i)-1, (ii)-2, (iii)-2, (iv)-1
- (c) (i)-2, (ii)-2, (iii)-1, (iv)-1
- (d) None of the above

- Q.15** For a sequence generator to generate a sequence .....101010111..... The maximum number of flip flops required are \_\_\_\_\_
- (a) 3
  - (b) 4
  - (c) 5
  - (d) None of the these

- Q.16** D flip flop is formed by combining the inputs of
- (a) J-K F/F
  - (b) S-R flip flop
  - (c) Master-slave J-K F/F
  - (d) T F/F

- Q.17** How many flip-flops are required to build a binary counter circuit to count from 0 to 2048?
- (a) 10
  - (b) 9
  - (c) 11
  - (d) 8

- Q.18** The master slave JK flip-flop is effectively a combination of  
 (a) a T flip-flop and a D flip-flop  
 (b) an SR flip-flop and a D flip-flop  
 (c) a SR flip-flop and a T flip-flop  
 (d) two T flip-flop

- Q.19** The Q-output of J-K flip-flop is "1". The output does not change when a clock-pulse is applied. The inputs J and K will be respectively (x-don't care state)  
 (a) 0 and x  
 (b) 0 and 1  
 (c) 1 and 0  
 (d) x and 0

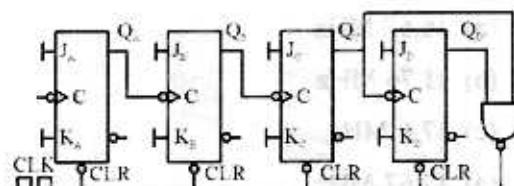
- Q.20** A divide by 78 counter can be realized by using  
 (a) 6 numbers of mod-13 counters  
 (b) one mod-13 counter followed by one mod-6 counter  
 (c) 13 numbers of mod-6 counters  
 (d) 13 numbers and mod-13 counters

- Q.21** The characteristic equation of the T-FF is given by  
 (a)  $Q^+ = T\bar{Q} + Q\bar{T}$   
 (b)  $Q^+ = \bar{T}Q + TQ$   
 (c)  $Q^+ = TQ$   
 (d)  $Q^+ = T\bar{Q}$

- Q.22** List the condition of the flag in the status register after the DAD B instruction has been executed  
 (a) Reset to 1  
 (b) Set to 1  
 (c) Set to zero  
 (d) None

### Common Data For Questions 23 & 24:

A counter is shown below:



- Q.23** The counter shown is  
 (a) Mod-12  
 (b) Mod-9  
 (c) Mod-14  
 (d) None.

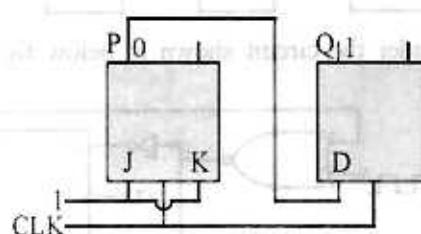
- Q.24** Frequency of output  $Q_D$  is  
 (a) 633 kHz  
 (b) 833 kHz  
 (c) 733 kHz  
 (d) None.

- Q.25** Which is the most appropriate match for the items in the first column with the items in the second column?

|   | Group I                  | Group II                        |
|---|--------------------------|---------------------------------|
| X | Indirect Addressing      | I. Array implementation         |
| Y | Indexed Addressing       | II. Writing relocatable code    |
| Z | Base Register Addressing | III. Passing array as parameter |

- (a) (X, II), (Y, III), (Z, I)  
 (b) (X, III), (Y, I), (Z, II)  
 (c) (X, III), (Y, II), (Z, I)  
 (d) (X, I), (Y, III), (Z, II)

- Q.26** The following arrangement of master-slave flip flops has the initial state of P, Q as 0, 1 respectively.

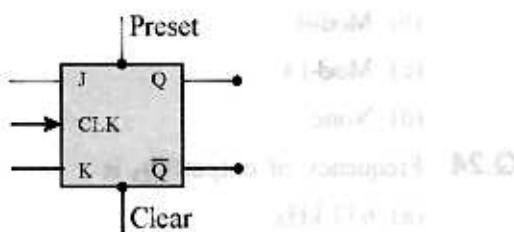


After three clock cycles, the output state P, Q respectively are

- (a) 1, 1  
 (b) 0, 0  
 (c) 0, 1  
 (d) 1, 0

**Q.27** The inputs of the J-K flip-flop, shown below are:

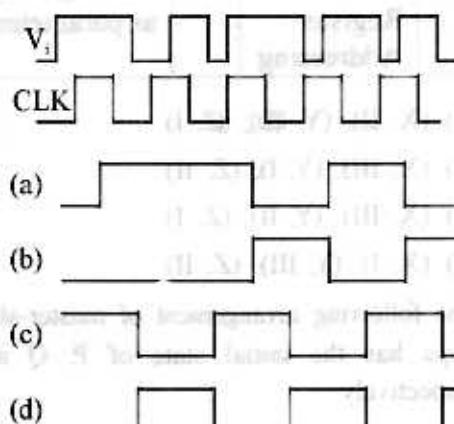
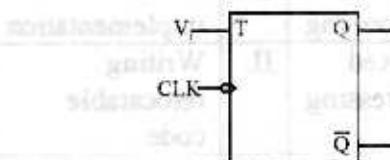
$$\text{PRESET} = \text{CLEAR} = 1; J = K = 0$$



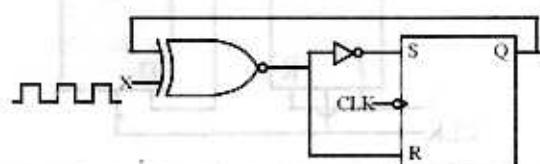
If a single clock pulse is applied, then device will

- (a) toggle
- (b) set
- (c) reset
- (d) not change states

**Q.28** The input signal  $V_i$ , shown in below fig., is applied to the FF, when initially in 0 state. Assume all timing constraints are satisfied. The output Q is



**Q.29** Consider the circuit shown in below fig.



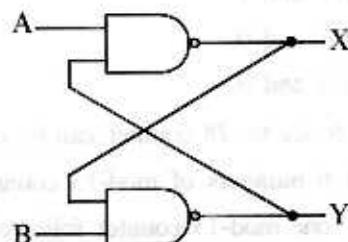
The expression for the next state  $Q^+$  is

- (a)  $xQ$
- (b)  $x\bar{Q}$
- (c)  $x \cup Q$
- (d)  $x \oplus Q$

**Q.30** Let the clock pulses be numbered 1, 2, 3..... after the point at which the FF is reset ( $Q_0 = 0$ ). The circuit is a

- (a) even parity checker
- (b) odd parity generator
- (c) Both A and B
- (d) None of the above

**Q.31** In below fig. initially  $A = 1$  and  $B = 1$ , the input B is now replaced by a sequence 1 0 1 0 1 0..... the outputs X and Y will be



- (a) Fixed at 1 and 0, respectively
- (b) Fixed at 0 and 1, respectively
- (c)  $X = 1010\dots$  while  $Y = 1010\dots$
- (d)  $X = 1010\dots$  while  $Y = 0101\dots$

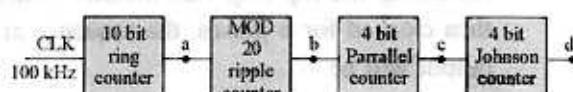
**Q.32** A 4 bit ripple counter and a 4 bit synchronous counter are made by flip-flops having a propagation delay of 10 ns each. If the worst case delay in the ripple counter and the synchronous counter be R and S respectively, then

- (a)  $R = 10 \text{ ns}, S = 40 \text{ ns}$
- (b)  $R = 10 \text{ ns}, S = 30 \text{ ns}$
- (c)  $R = 30 \text{ ns}, S = 10 \text{ ns}$
- (d)  $R = 40 \text{ ns}, S = 10 \text{ ns}$

**Q.33** A 3-stage ripple counter, with propagation delay of flip flop as 25 nsec and pulse width of strobe input is 10 nsec. Then the maximum operating frequency at which counter operates reliably is

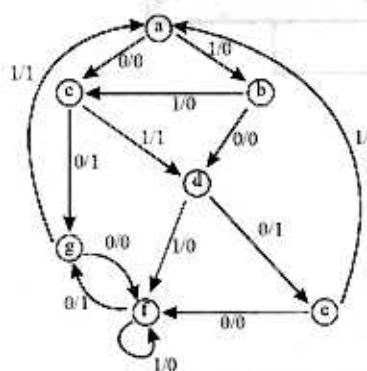
- (a) 16.67 MHz
- (b) 11.76 MHz
- (c) 17.6 MHz
- (d) 12.67 MHz

**Q.34** What is the frequency of the pulses at the point a, b, c and d in the circuit?



- (a) 100 kHz, 10 kHz, 500 Hz, 31.25 Hz
- (b) 10 kHz, 1 kHz, 62.5 Hz, 7.81 Hz
- (c) 10 kHz, 500 Hz, 31.25 Hz, 3.9 Hz
- (d) 100 kHz, 1 kHz, 31.25 Hz, 3.9 Hz

**Q.35** Following state diagram shows clocked sequential circuit:



How many states the sequential circuit has?

- (a) 6
- (b) 7
- (c) 4
- (d) 5

**Q.36** Maximum propagation delay for a synchronous counter and an asynchronous counter, each with 4 flip-flops is \_\_\_\_\_ and \_\_\_\_\_ respectively. The propagation delay of one flip-flop is 20 nsec.

- (a) 20 nsec, 20 nsec
- (b) 80 nsec, 80 nsec
- (c) 20 nsec, 80 nsec
- (d) 80 nsec, 20 nsec

**Q.37** If a counter having 10 FFs is initially at 0, what count will it hold after 2060 pulses?

- (a) 000 000 1110
- (b) 000 001 1100
- (c) 000 001 1000
- (d) 000 000 1100

**Q.38** A gate draws  $2.8\mu A$  when its output is HIGH and  $3.6\mu A$  when its output is LOW. Find out the power dissipation when the voltage supply  $V_{CC}$  is 5V and the gate is operated on 50% duty cycle?

- (a)  $12\mu W$
- (b)  $13\mu W$
- (c)  $16\mu W$
- (d)  $18\mu W$

**Q.39** To convert any counter into count down, the gate that can be used is

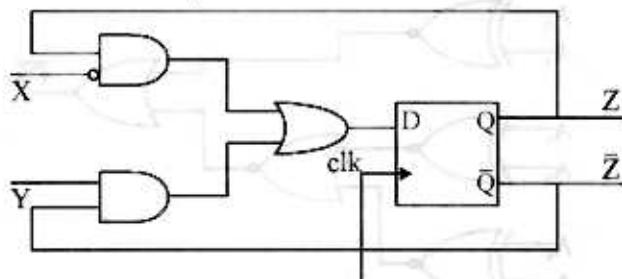
- (a) XNOR
- (b) AND
- (c) XOR
- (d) OR

**Q.40** A 4 bit shift register shifts 1 bit to the right at every clock pulse. The D input is derived from  $Q_0$ ,  $Q_2$  and  $Q_3$  through two XOR gates. To what values should the shift register be initialized so that the pattern (1001) occurs after the first clock pulse?

- (a) 0101
- (b) 0001
- (c) 1000
- (d) 0010

### LEVEL-3

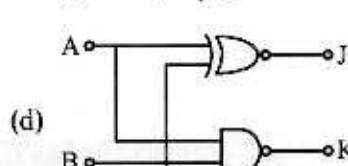
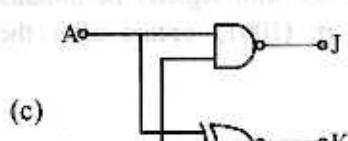
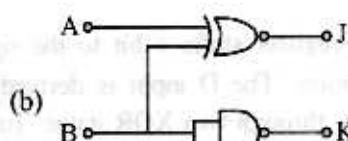
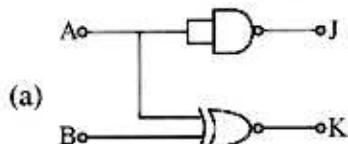
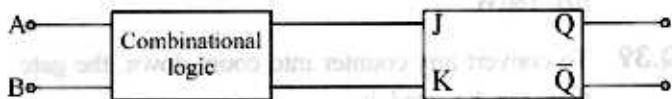
**Q.41** A sequential circuit using D flip-flop and logic gates is shown where X and Y are inputs and Z is the output. Then the circuit is



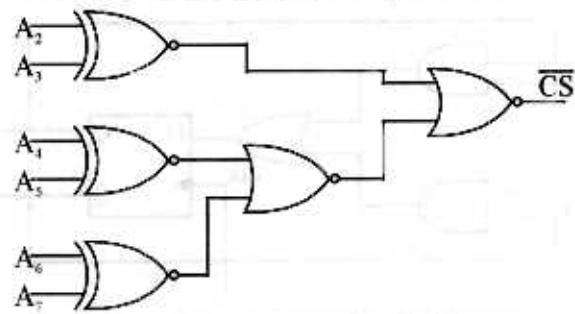
- (a) RS flip-flop with  $X = R$  and  $Y = S$
- (b) JK flip-flop with  $X = J$  and  $Y = K$
- (c) RS flip-flop with  $X = Z$  and  $Y = R$
- (d) JK flip-flop with  $X = K$  and  $Y = J$

- Q.42** The circuit realization of the combination logic block shown in figure to obtain the following truth table will be,

| A | B | Q           |
|---|---|-------------|
| 0 | 0 | $\bar{Q}_n$ |
| 0 | 1 | 1           |
| 1 | 0 | $Q_n$       |
| 1 | 1 | 0           |

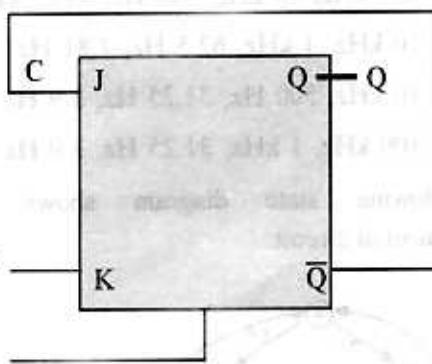


- Q.43** The decoding circuit has been used to generate the active low chip select signal for a microprocessor peripheral (The address line are designated as  $A_0$  to  $A_7$  for I-O address)



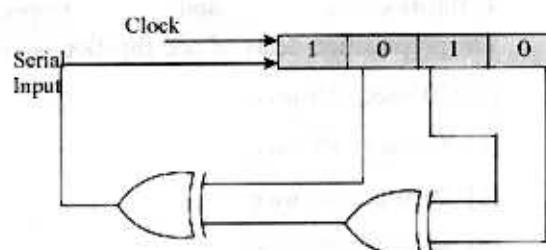
- (a)  $A_4$  to  $A_7$  H
- (b) 60 H to 63 H
- (c) 70 H to 73 H
- (d) 50 H to 53 H

- Q.44** In a JK flip-flop we have  $I = \bar{Q}$  and  $K = 1$ . Assuming the flip-flop was initially cleared and then clocked for 6 pulses, the sequence at the Q output will be



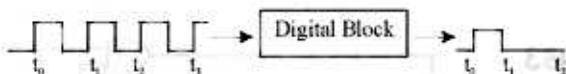
- (a) 010000
- (b) 010101
- (c) 010010
- (d) 011001

- Q.45** The shift register shown in the given figure is initially loaded with the bit pattern 1010. Subsequently the shift register is clocked, and with each clock pulse the pattern gets shifted by one bit position to the right. With each shift, the bit at the serial input is pushed to the left most position (msb). After how many clock pulses will the content of the shift register become 1010 again?

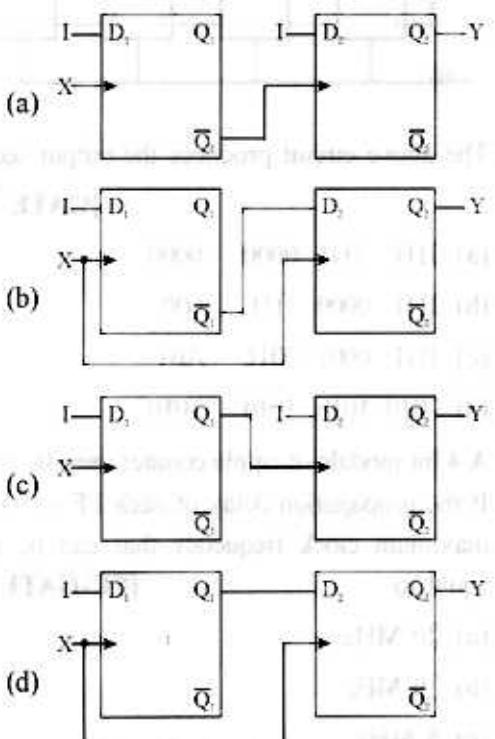


- (a) 3
- (b) 11
- (c) 7
- (d) 10

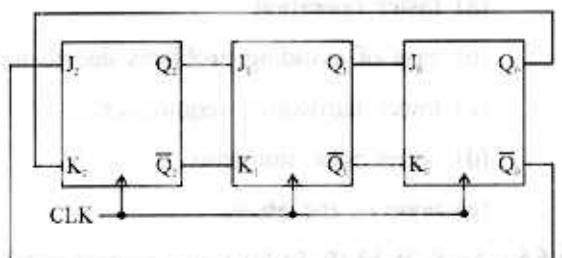
- Q.46** The digital block in figure realized using two positive edge triggered D- flip-flop. Assume that for  $t < t_0$ ,  $Q_1 = Q_2 = 0$ .



The circuit in the digital block is given by

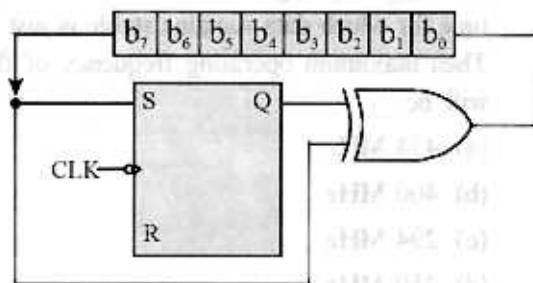


- Q.47** The three-stage Johnson counter as shown in figure is clocked at a constant frequency of  $f_c$  from the starting state of  $Q_2\ Q_1\ Q_0 = 101$ . The frequency of output  $Q_2\ Q_1\ Q_0$  will be



- (a)  $\frac{f_c}{2}$
- (b)  $\frac{f_c}{6}$
- (c)  $\frac{f_c}{3}$
- (d)  $\frac{f_c}{8}$

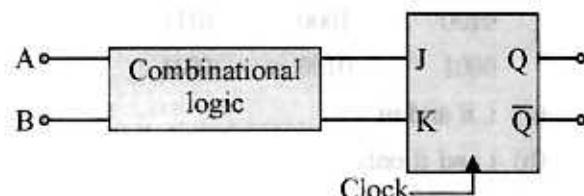
- Q.48** The 8-bit left shift register and D-flip-flop shown in below fig is synchronized with same clock. The D-flip-flop is initially cleared.



The circuit act as

- (a) Binary to Gray code converter
- (b) Binary to 2's complement converter
- (c) Binary to 1's complement converter
- (d) Binary to Excess-3 code converter

- Q.49** To realize the given truth table from the circuit shown in the figure, the input to J in terms of A and B would have to be



Truth Table

| A | B | $Q_{n+1}$ | J |
|---|---|-----------|---|
| 0 | 0 | $Q_n$     | 0 |
| 0 | 1 | 1         | 1 |
| 1 | 0 | $Q_n$     | 0 |
| 1 | 1 | 0         | X |

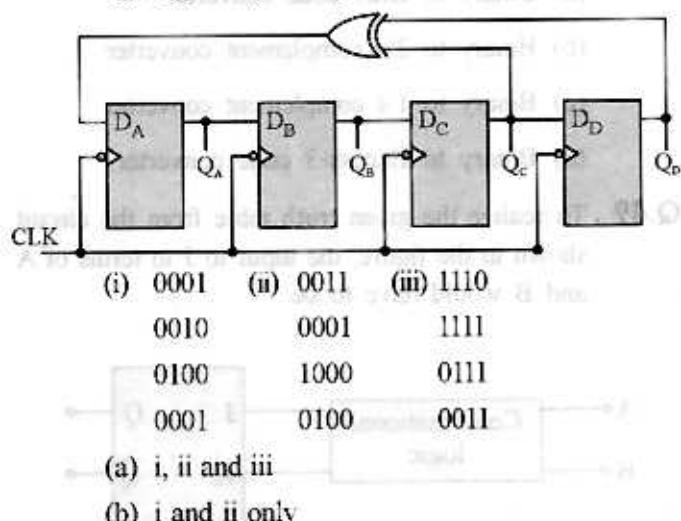
- (a)  $\bar{A}$
- (b) B
- (c)  $\bar{A}B$
- (d) AB

- Q.50** Suppose propagation delay time of flip flop is 0.2 nsec and it is followed by 2 decoders with a propagation delay time of 0.1 nsec each.

The time required for data input to settle before the triggering edge of clock is 2 nsec and the time for which data remains stable is just 1 nsec. Then maximum operating frequency of flip flop will be \_\_\_\_\_

- 434 MHz
- 400 MHz
- 294 MHz
- 450 MHz

- Q.51** Figure below shows D type Flip-Flops connected as shift register, then outputs  $Q_D$ ,  $Q_C$ ,  $Q_B$ ,  $Q_A$  are given by :



- Q.52** An X-Y flip flop, whose Characteristic Table is given below is to be implemented using a J-K flip flop

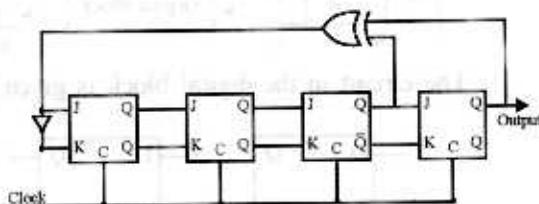
| X | Y | $Q_{n+1}$ |
|---|---|-----------|
| 0 | 0 | 1         |
| 0 | 1 | $Q_n$     |
| 1 | 0 | $Q_n$     |
| 1 | 1 | 0         |

This can be done by making

- $J = \bar{Y}$ ,  $K = X$
- $J = \bar{X}$ ,  $K = Y$
- $J = Y$ ,  $K = \bar{X}$
- $J = X$ ,  $K = \bar{Y}$

## GATE QUESTIONS

**Q53**



The above circuit produces the output sequence:

[GATE 1987]

- 1111 1111 0000 0000
- 1111 0000 1111 000
- 1111 0001 0011 010
- 1010 1010 1010 1010

- Q.54** A 4 bit modulo-6 ripple counter uses JK flip flop. If the propagation delay of each FF is 50 ns, the maximum clock frequency that can be used is equal to

[EC-GATE 1990]

- 20 MHz
- 10 MHz
- 5 MHz
- 4 MHz

- Q.55** Advantage of synchronous sequential circuits over asynchronous ones is: [GATE 1991]

- faster operation
- ease of avoiding problems due to hazards
- lower hardware requirement
- better noise immunity
- none of the above

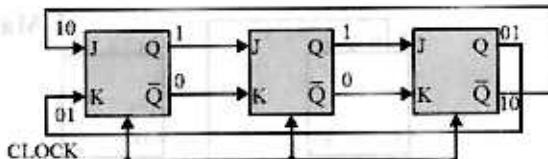
- Q.56** An S-R FLIP-FLOP can be converted into a T FLIP-FLOP by connecting \_\_\_\_\_ to Q and \_\_\_\_\_ to  $\bar{Q}$ .

[EC-GATE 1991]

- $S^+$ ,  $R^-$
- $S^-$ ,  $R^+$
- $S^+$ ,  $R$
- $S$ ,  $R$

**Q.57** For the initial state of 000, the function performed by the arrangement of the J-K flip-flops in the figure [GATE 1993]

[2-Marks]



- (a) Shift Register
- (b) Mod-3 Counter
- (c) Mod-6 Counter
- (d) Mod-2 Counter
- (e) None of the above

**Q.58** An R-S latch is [EC-GATE 1995]

- (a) combinational circuit
- (b) synchronous sequential circuit
- (c) one bit memory element
- (d) one clock delay element

**Q.59** Consider the following state table, for a sequential machine. The number of states in the minimized machine will be [GATE 1996]

| Present state | Input |    | Next State Output |
|---------------|-------|----|-------------------|
|               | 0     | 1  |                   |
| A             | D0    | B0 |                   |
| B             | A0    | C1 |                   |
| C             | A0    | B1 |                   |
| D             | A1    | C1 |                   |

- (a) 4
- (b) 3
- (c) 2
- (d) 1

**Q.60** Match the pair:

| Group I |                              | Group II |                                   |
|---------|------------------------------|----------|-----------------------------------|
| A.      | A shift register can be used | 1.       | for code conversion               |
| B.      | A multiplexer can be used    | 2.       | to generate memory chip select    |
| C.      | A decoder can be used        | 3.       | for parallel to serial conversion |
|         |                              | 4.       | as a many to one switch           |
|         |                              | 5.       | for analog to digital conversion  |

[EC-GATE 1996]

[2-Marks]

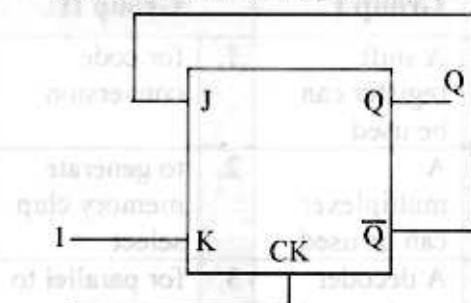
- (a) (A)  $\rightarrow$  (3), (B)  $\rightarrow$  (4), (C)  $\rightarrow$  (2)
- (b) (A)  $\rightarrow$  (5), (B)  $\rightarrow$  (4), (C)  $\rightarrow$  (1)
- (c) (A)  $\rightarrow$  (4), (B)  $\rightarrow$  (2), (C)  $\rightarrow$  (3)
- (d) (A)  $\rightarrow$  (3), (B)  $\rightarrow$  (4), (C)  $\rightarrow$  (5)

**Q.61** A pulse train can be delayed by a finite number of clock periods using [EC-GATE 1996]

[1-Mark]

- (a) a serial-in-serial-out shift register
- (b) a serial-in-parallel-out shift register
- (c) a parallel-in-serial-out shift register
- (d) a parallel-in-parallel-out shift register

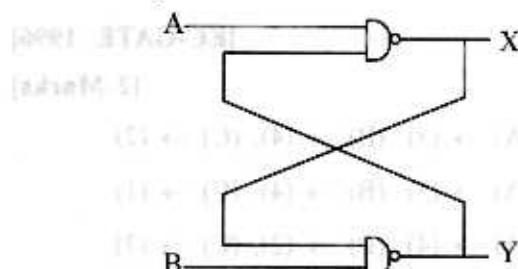
- Q.62** In a J-K flip-flop we have  $J = Q'$  and  $K = \bar{I}$ . (Figure shown). Assuming that the flip flop was initially cleared and then clocked for 6 pulses, the sequence at the Q output will be



[EC-GATE 1997]

- (a) 010000
- (b) 011001
- (c) 010010
- (d) 010101

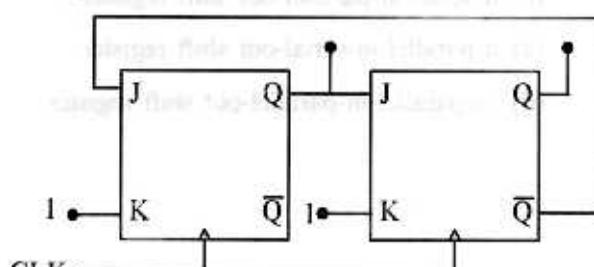
- Q.63** In Figure,  $A = 1$  and  $B = 1$ , the input B is now replaced by a sequence 101010.... the output x and y will be



[EC-GATE 1998]

- (a) fixed at 0 and 1, respectively
- (b)  $x = 1010\dots$  while  $y = 0101\dots$
- (c)  $x = 1010\dots$  while  $y = 1010\dots$
- (d) fixed at 1 and 0 respectively

- Q.64** Figure shows a mod-K counter, here K is equal to

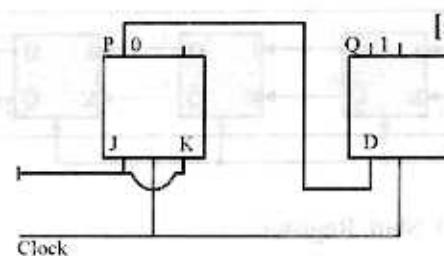


[EC-GATE 1998]

- (a) 1
- (b) 2
- (c) 3
- (d) 4

- Q.65** The following arrangement of master-slave flip flops has the initial state of P, Q as 0, 1 (respectively). After three clock cycles the output state P, Q is (respectively),

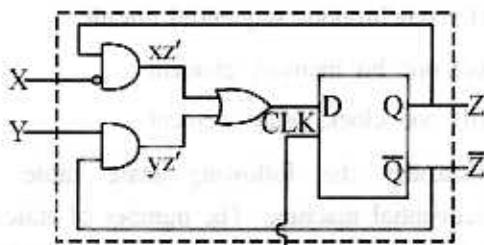
[GATE 2000]



[2-Marks]

- (a) 1, 0
- (b) 1, 1
- (c) 0, 0
- (d) 0, 1

- Q.66** A sequential circuit using D Flip-Flop and logic gates is shown in the figure, where X and Y are the inputs and Z is the output. The circuit is

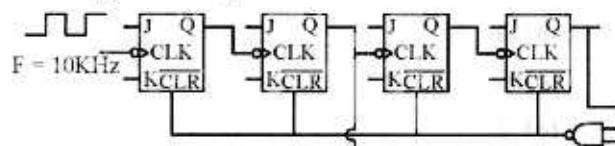


[EC-GATE 2000]

[2-Marks]

- (a) S-R Flip-Flop with inputs X = S and Y = R
- (b) S-R Flip-Flop with inputs X = R and Y = S
- (c) J-K Flip-Flop with inputs X = K and Y = J
- (d) J-K Flip-Flop with inputs X = J and Y = K

- Q.67** In the figure, the J and K inputs of all the four Flip-Flops are made high. The frequency of the signal at output Y is

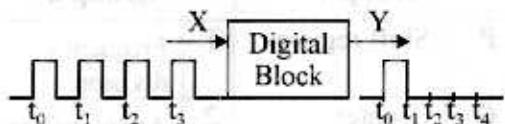


[EC-GATE 2001]

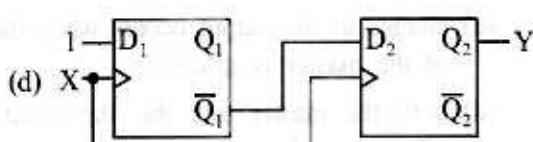
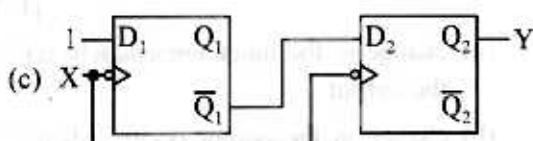
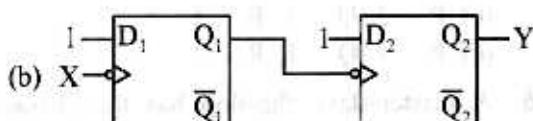
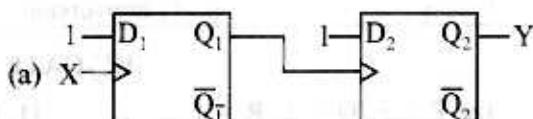
[2-Marks]

- (a) 0.833 KHz
- (b) 0.77 KHz
- (c) 0.91 KHz
- (d) 1.0 KHz

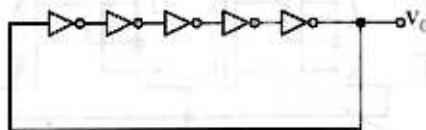
- Q.68** The digital block in the figure is realized using two positive edge triggered D-flip-flops. Assume that for  $t < t_0$ ,  $Q_1 = Q_2 = 0$ . The circuit in the digital block is given by:



[EC-GATE 2001]  
[2-Marks]



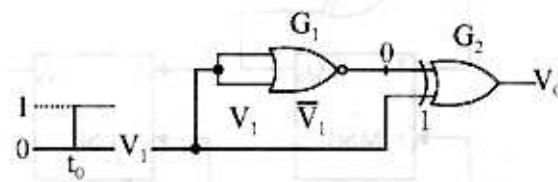
- Q.69** For the ring oscillator shown in the figure, the propagation delay of each inverter is 100 pico sec. What is the fundamental frequency of the oscillator output?



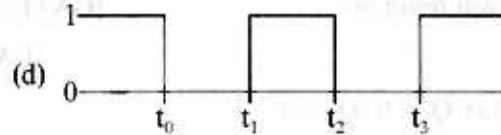
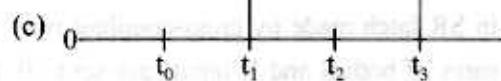
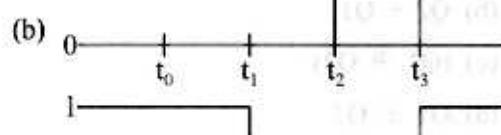
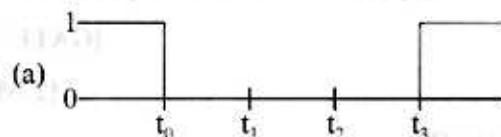
[EC-GATE 2001]  
[1-Mark]

- (a) 1 GHz
- (b) 2 GHz
- (c) 10 MHz
- (d) 100 MHz

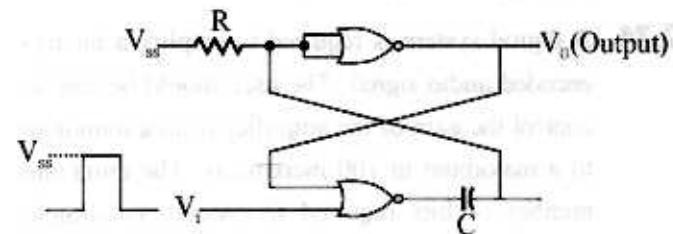
- Q.70** The gates  $G_1$  and  $G_2$  in the figure have propagation delays of 10 nsec and 20 nsec respectively. If the input  $V_i$  makes an abrupt change from logic 0 to 1 at time  $t = t_0$ , then the output waveform  $V_o$  is



[EC-GATE 2002]  
[2-Marks]



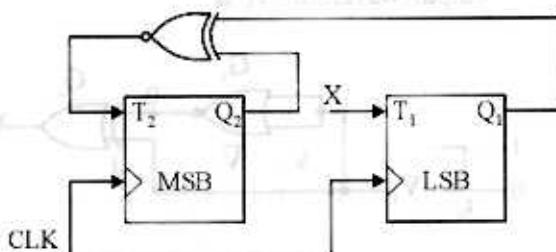
- Q.71** The circuit in the figure has two CMOS NOR-gates. This circuit functions as a:



[EC-GATE 2002]  
[2-Marks]

- (a) flip-flop
- (b) Schmitt trigger
- (c) astable multivibrator
- (d) monostable multivibrator

- Q.72** Consider the partial implementation of a 2-bit counter using T flip-flops following the sequence 0-2-3-1-0, as shown below.



To complete the circuit, the input X should be

[GATE 2004]

[2-Marks]

- (a)  $Q_2'$
- (b)  $Q_2 + Q_1$
- (c)  $(Q_1 \oplus Q_2)$
- (d)  $Q_1 \oplus Q_2$

- Q.73** In SR latch made by cross-coupling two NAND gates, if both S and R inputs are set to 0, then it will result in:

[GATE 2004]

[1-Mark]

- (a)  $Q = 0, Q' = 1$
- (b)  $Q = 1, Q' = 0$
- (c)  $Q = 1, Q' = 1$
- (d) Indeterminate states

- Q.74** A digital system is required to amplify a binary-encoded audio signal. The user should be able to control the gain of the amplifier from a minimum to a maximum in 100 increments. The minimum number of bits required to encode, in straight binary, is

[EC-GATE 2004]

[1-Mark]

- (a) 8
- (b) 7
- (c) 6
- (d) 5

- Q.75** Choose the correct one from among the alternatives A, B, C, D after matching an item from Group 1 with the most appropriate item in Group 2.

| Group 1 |                | Group 2 |                                    |
|---------|----------------|---------|------------------------------------|
| P.      | Shift register | 1.      | Frequency division                 |
| Q.      | Counter        | 2.      | Addressing in memory chips         |
| R.      | Decoder        | 3.      | Serial to parallel data conversion |

[EC-GATE 2004]

- (a) P - 3, Q - 2, R - 1
- (b) P - 1, Q - 2, R - 2
- (c) P - 2, Q - 1, R - 3
- (d) P - 3, Q - 1, R - 2

[1-Mark]

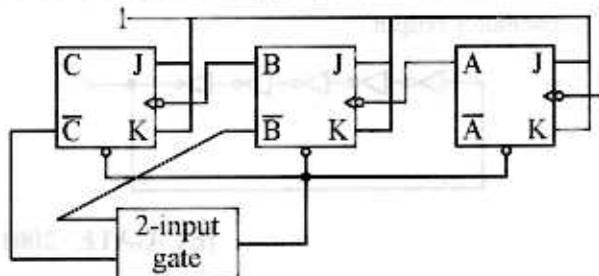
- Q.76** A master-slave flip-flop has the characteristic that

[EC-GATE 2004]

[1-Mark]

- (a) change in the input immediately reflected in the output
- (b) change in the output occurs when the state of the slave is affected
- (c) change in the output occurs when the state of the master is affected
- (d) both the master and the slave states are affected at the same time

- Q.77** In the modulo-6 ripple counter shown in the figure, the output of the 2-input gate is used to clear the J-K flip-flops.



The 2-input gate is

[EC-GATE 2004]

[2-Marks]

- (a) an OR gate
- (b) a NOR gate
- (c) a NAND gate
- (d) an AND gate

**Q.78** A CPU has only three instructions I1, I2 and I3, which use the following signals in time steps T1-T5:

I1: T1: Ain, Bout, Cin

T2: PCout, Bin

T3: Zout, Ain

T4: PCin, Bout

T5: End

I2: T1: Cin, Bout, Din

T2: Aout, Bin

T3: Zout, Ain

T4: Bin, Cout

T5: End

I3: T1: Din, Aout

T2: Din, Bout

T3: Zout, Ain

T4: Dout, Ain

T5: End

Which of the following logic functions will generate the hardwired control for the signal Ain?

[IT-GATE 2004]

[2-Marks]

(a)  $T_1.I_1 + T_2.I_3 + T_4.I_3 + T_5$

(b)  $(T_1 + T_2 + T_3).I_3 + T_1.I_1$

(c)  $(T_1 + T_2).I_1 + (T_2 + T_4).I_3 + T_3$

(d)  $(T_1 + T_2).I_2 + (T_1 + T_3).I_1 + T_3$

**Q.79** How many pulses are needed to change the contents of a 8 bit up-counter from 10101100 to 00100111 (rightmost bit is the LSB)?

[IT-GATE 2005]

[1 Mark]

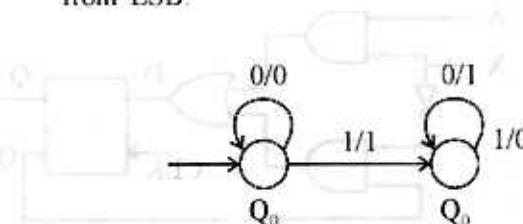
(a) 134

(b) 133

(c) 124

(d) 123

**Q.80** The following diagram represents a finite state machine which takes as input a binary number from LSB.



Which of the following is TRUE? [GATE 2005]

[2-Marks]

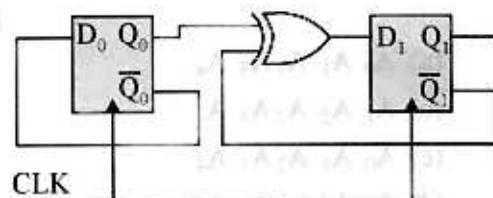
(a) It computes 1's complement of the input number.

(b) It computes 2's complement of the input number.

(c) It increments the input number.

(d) It decrements the input number.

**Q.81** Consider the following circuit.



The flip-flops are positive edge triggered D FFs. Each state is designated as a two bit string  $Q_0Q_1$ . Let the initial state be 00. The state transition sequence is

[GATE 2005]

[2-Marks]

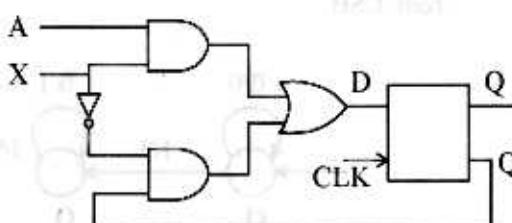
(a)  $00 \rightarrow 11 \rightarrow 01$

(b)  $00 \rightarrow 11$

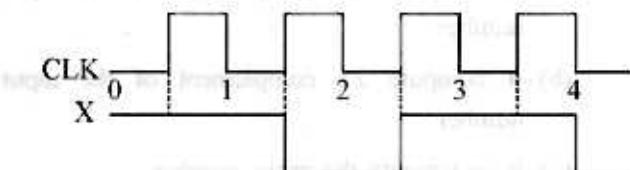
(c)  $00 \rightarrow 10 \rightarrow 01 \rightarrow 11$

(d)  $00 \rightarrow 11 \rightarrow 01 \rightarrow 10$

- Q.82** Consider the following circuit involving a positive edge triggered D flip-flop.



Consider the following timing diagram. Let A represent the logic level on the line a in the  $i^{\text{th}}$  clock period.



Let  $A'$  represent the complement of A. The correct output sequence of Y over the clock periods 1 through 5 is

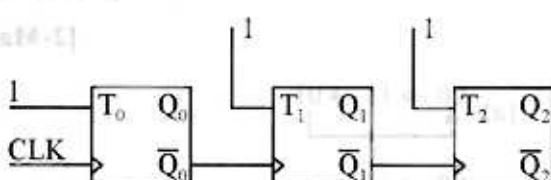
[GATE 2005]

[2-Marks]

- (a)  $A_0 \ A_1 \ A'_1 \ A_3 \ A_4$
- (b)  $A_1 \ A_2 \ A'_2 \ A_3 \ A_4$
- (c)  $A_0 \ A_1 \ A_2 \ A'_3 \ A_4$
- (d)  $A_1 \ A_2 \ A'_3 \ A_4 \ A_5$

- Q.83** The given figure shows a ripple counter using positive edge triggered flip-flops.

If the present state of the counter is  $Q_2Q_1Q_0 = 011$ , then its next state ( $Q_2Q_1Q_0$ ) will be

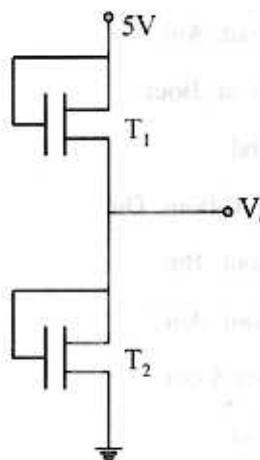


[EC-GATE 2005]

[2-Marks]

- (a) 111
- (b) 101
- (c) 100
- (d) 010

- Q.84** Both transistors  $T_1$  and  $T_2$  shown in the figure, have a threshold voltage of 1 Volts. The device parameters  $K_1$  and  $K_2$  of  $T_1$  and  $T_2$  are, respectively,  $36 \mu\text{A}/\text{V}^2$ . The output voltage  $V_0$  is



[EC-GATE 2005]

[2-Marks]

- (a) 4 V
- (b) 3 V
- (c) 2 V
- (d) 1 V

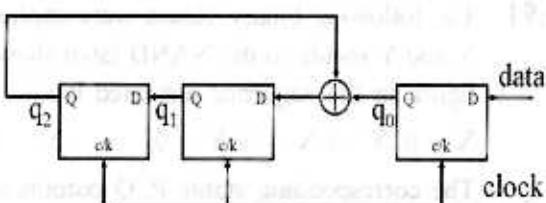
- Q.85** The present output  $Q_n$  of an edge triggered JK flip-flop is logic 0. If  $J = 1$ , then  $Q_{n+1}$

[EC-GATE 2005]

[2-Marks]

- (a) will be logic 0
- (b) will be logic 1
- (c) cannot be determined
- (d) will race around

- Q.86** Consider a Boolean function  $f(w, x, y, z)$ . Suppose that exactly one of its inputs is allowed to change at a time. If the function happens to be true for two input vectors  $1 = (w_1, x_1, y_1, z_1)$  and  $2 = (w_2, x_2, y_2, z_2)$ , we would like the function to remain true as the input changes from 1 to 2 ('1' and '2' differ in exactly one bit position), without becoming false.



momentarily. Let  $f(w, x, y, z) = (5, 7, 11, 12, 13, 15)$ . Which of the following cube covers of  $f$  will ensure that the required property is satisfied?

[GATE 2006]

- (a)  $\bar{w}xz, w\bar{x}\bar{y}, \bar{x}\bar{y}z, xyz, wyz$
- (b)  $wxy, wxz, wyz$
- (c)  $wxyz, xz, wxyz$
- (d)  $wzy, wyz, wxz, wxz, xyz, xyz$

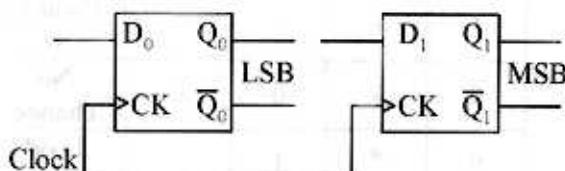
**Q.87** You are given a free running clock with a duty cycle of 50% and a digital waveform  $f$  which changes only at the negative edge of the clock. Which one of the following circuits (using clocked D flip-flops) will delay the phase of  $f$  by  $180^\circ$ ? [GATE 2006]

[1-Mark]

- (a)
- (b)
- (c)
- (d)

**Q.88** Two D-flip-flops, as shown below, are to be connected as a synchronous counter that goes through the following  $Q_1 Q_0$  sequence  
 $00 \rightarrow 01 \rightarrow 11 \rightarrow 10 \rightarrow 00 \dots$

The inputs  $D_0$  and  $D_1$  respectively should be connected as

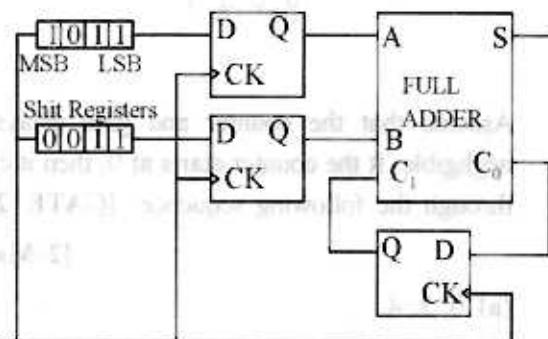


[EC-GATE 2006]

[2-Marks]

- (a)  $\overline{Q_1 Q_0}$  and  $Q_1 Q_0$
- (b)  $\overline{Q_0}$  and  $Q_1$
- (c)  $\overline{Q_1} Q_0$  and  $\overline{Q_1} Q_0$
- (d)  $\overline{Q_1}$  and  $Q_0$

**Q.89** For the circuit shown in the figure below, two 4-bit parallel-in serial-out shift registers loaded with the data shown are used to feed the data to a full adder. Initially, all the flip-flops are in clear state. After applying two clock pulses, the outputs of the full adder should be



[EC-GATE 2006]

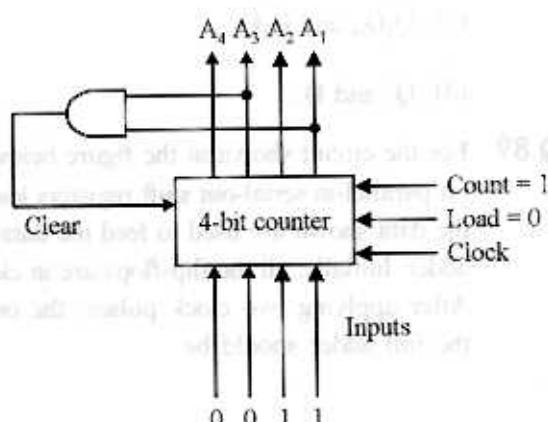
[2-Marks]

- (a)  $S = 0 \quad C_0 = 1$
- (b)  $S = 0 \quad C_0 = 0$
- (c)  $S = 1 \quad C_0 = 1$
- (d)  $S = 1 \quad C_0 = 0$

**Q.90** The control signal functions of a 4-bit binary counter are given below (where X is "don't care"):

| Clear | Clock | Load | Count | Function   |
|-------|-------|------|-------|------------|
| 1     | x     | x    | x     | Clear to 0 |
| 0     | x     | 0    | 0     | No change  |
| 0     | ↑     | 1    | x     | Load input |
| 0     | ↑     | 0    | 1     | Count next |

The counter is connected as follows:



Assume that the counter and gate delays are negligible. If the counter starts at 0, then it cycles through the following sequence: [GATE 2007]

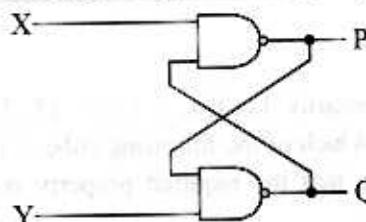
[2-Marks]

- (a) 0, 3, 4
- (b) 0, 3, 4, 5
- (c) 0, 1, 2, 3, 4
- (d) 0, 1, 2, 3, 4, 5

**Q.91** The following binary values were applied to the X and Y inputs to the NAND latch shown in the figure in the sequence indicated below:

$$X = 0, Y = 1; X = 0, Y = 0; \quad X = 1, Y = 1.$$

The corresponding stable P, Q outputs will be

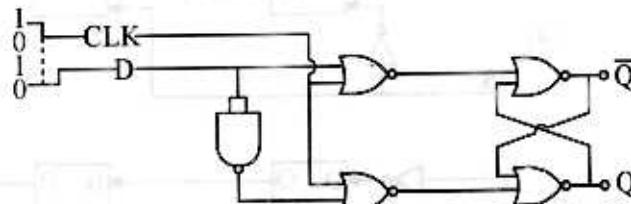


[EC-GATE 2007]

[2-Marks]

- (a)  $P = 1, Q = 0; P = 1, Q = 0;$   
 $P = 1, Q = 0$  or  $P = 0, Q = 1$
- (b)  $P = 1, Q = 0; P = 1, Q = 1;$   
 $P = 1, Q = 0$  or  $P = 0, Q = 1$
- (c)  $P = 1, Q = 0; P = 0, Q = 1$  or  
 $P = 0, Q = 1; P = 0, Q = 1$
- (d)  $P = 1, Q = 0; P = 1, Q = 1;$   
 $P = 1, Q = 1$

**Q.92** For the circuit shown in the figure, D has a transition from 0 to 1 after CLK changes from 1 to 0. Assume gate delays to be negligible



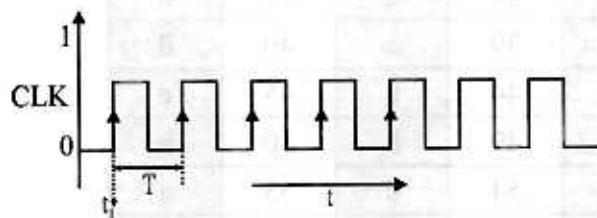
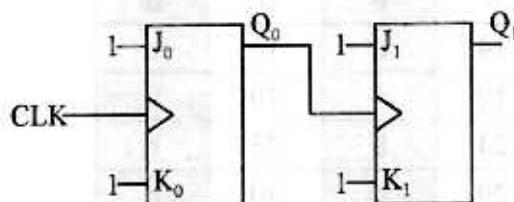
Which of the following statements is true?

[EC-GATE 2008]

[2-Marks]

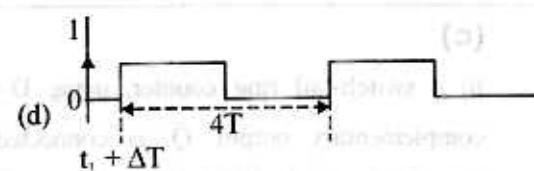
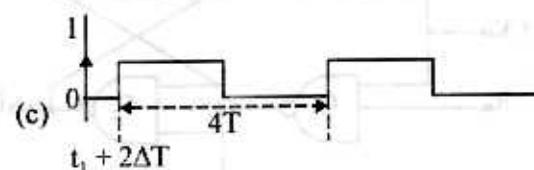
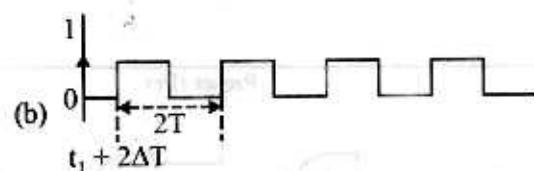
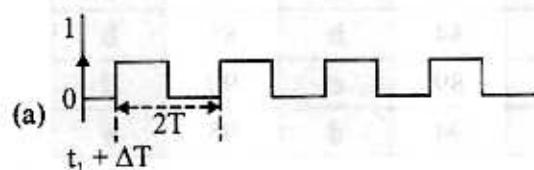
- (a) Q goes to 1 at the CLK transition and stays at 1.
- (b) Q goes to 0 at the CLK transition and stays at 0.
- (c) Q goes to 0 at the CLK transition and goes to 1 when D goes to 1.
- (d) Q goes to 1 at the CLK transition and goes to 0 when D goes to 1.

- Q.93** For each of the positive edge-triggered J-K flip flop used in the following figure, the propagation delay is  $\Delta T$

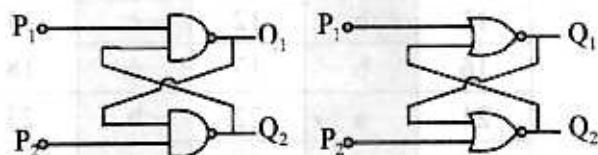


Which of the following waveforms correctly represents the output at  $Q_1$ ? [EC-GATE 2008]

[2-Marks]



- Q.94** Refer to the NAND and NOR latches shown in the figure. The inputs ( $P_1, P_2$ ) for both the latches are first made (0,1) and then, after a few seconds, made (1,1). The corresponding stable outputs ( $Q_1, Q_2$ ) are

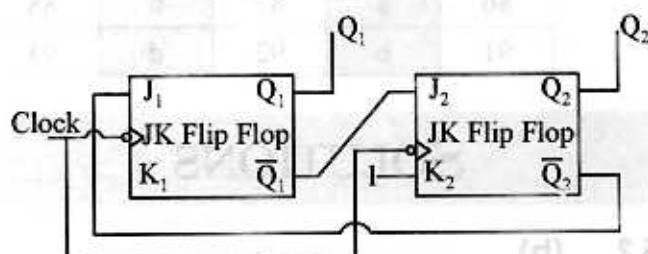


[EC-GATE 2009]

[2-Marks]

- NAND: first (0,1) then (0,1) NOR: first (1,0) then (0,0)
- NAND: first (1,0) then (1,0) NOR: first (1,0) then (1,0)
- NAND: first (1,0) then (1,1) NOR: first (0,1) then (0,1)
- NAND: first (1,0) then (1,0) NOR: first (1,0) then (0,0)

- Q.95** What are the counting states ( $Q_1, Q_2$ ) for the counter shown in the figure below?



[EC-GATE 2009]

[2-Marks]

- 01, 10, 11, 00, 01, ...
- 11, 10, 00, 11, 10, ...
- 00, 11, 01, 10, 00, ...
- 01, 10, 00, 01, 10, ...

# ANSWER KEY

|    |   |    |   |    |   |    |   |    |   |
|----|---|----|---|----|---|----|---|----|---|
| 1  | a | 2  | b | 3  | a | 4  | a | 5  | a |
| 6  | a | 7  | c | 8  | b | 9  | b | 10 | d |
| 11 | b | 12 | c | 13 | d | 14 | d | 15 | c |
| 16 | b | 17 | c | 18 | c | 19 | d | 20 | b |
| 21 | a | 22 | b | 23 | a | 24 | b | 25 | b |
| 26 | d | 27 | d | 28 | d | 29 | d | 30 | d |
| 31 | b | 32 | d | 33 | b | 34 | c | 35 | d |
| 36 | c | 37 | d | 38 | c | 39 | c | 40 | d |
| 41 | d | 42 | a | 43 | c | 44 | b | 45 | c |
| 46 | d | 47 | a | 48 | a | 49 | b | 50 | b |
| 51 | d | 52 | a | 53 | c | 54 | c | 55 | a |
| 56 | d | 57 | c | 58 | c | 59 | a | 60 | a |
| 61 | b | 62 | d | 63 | a | 64 | c | 65 | b |
| 66 | c | 67 | d | 68 | d | 69 | a | 70 | b |
| 71 | d | 72 | d | 73 | c | 74 | b | 75 | d |
| 76 | c | 77 | a | 78 | a | 79 | b | 80 | b |
| 81 | d | 82 | a | 83 | c | 84 | b | 85 | b |
| 86 | a | 87 | b | 88 | d | 89 | c | 90 | d |
| 91 | b | 92 | d | 93 | c | 94 | d | 95 | b |

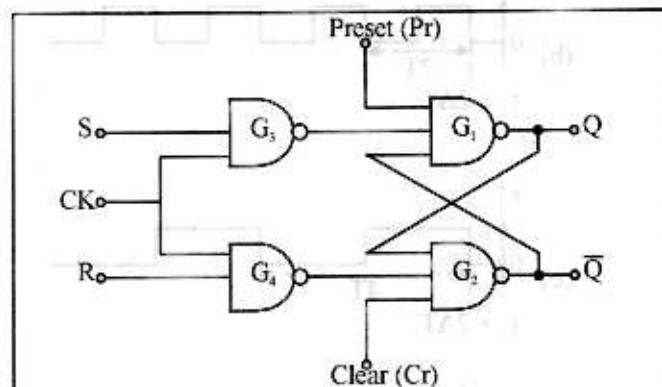
## SOLUTIONS

**S.2 (b)**

RS flip-flop results in indeterminate state at input 1, while JK flip-flop produces the output.

**S.5 (a)**

If preset and clear both are 0 then output  $Q$ ,  $\bar{Q}$  will be uncertain because preset and clear are active low signals. Hence zero to preset or clear indicates high input to gates as shown.


**S.7 (c)**

In a switch-tail ring counter, using D-FF, the complementary output  $\bar{Q}$  is connected to D input for a single D-FF it becomes a T-FF.

**S.8 (b)**

Initially  $J = K = 1$ , therefore the next output will be 0.

Now,  $J = 1$  and  $K = 0$ , therefore output will be 1.

Again  $J = K = 1$  and next output will be 0.

This cycle continues and the output will be of the form 0, 1, 0, 1, 0, 1, ....

**S.11 (b)**

The preset input is given by

$$\begin{aligned} P &= 16 - m \text{ for 4 bit binary counter} \\ &= 16 - 4 = 12 \end{aligned}$$

**S.12 (c)**

Race around condition is toggling of output if and only if  $J = K = 1$ . So this is the normal operation of flipflop.

**S.13 (d)**

Memory element is provided with clock signal and it is also referred to as Time delay device.

**S.14 (d)**

All the flip flops ( $S - R$ ,  $J - K$ ,  $D$ ,  $T$ ) are level triggered but only master slave flip flop is edge triggered.

**S.15 (c)**

To generate a sequence of length  $\ell$

$$\ell \leq 2^N - 1 \quad N: \text{Number of flip flops}$$

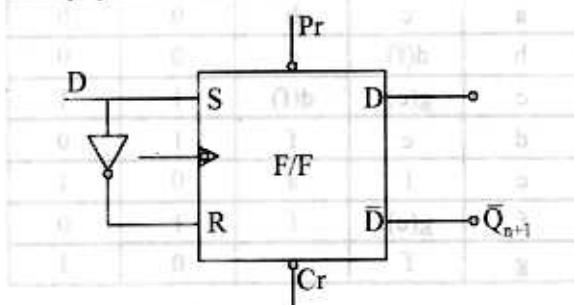
$\therefore$  Sequence here is of 9 bits

$$\therefore 9 \leq 2^N - 1$$

$$\therefore 10 \leq 2^N$$

$\therefore$  By trial and error method  $N = 4$

But 4 is the minimum number of flip flops required and not the maximum

**S.16 (b)**

Thus, D Flip-Flop can be formed by combining the inputs of SR Flip-Flop.

**S.17 (c)**

Number of FFs require  $= 2^n = 2048$

$$\therefore n = 11$$

**S.19 (d)**

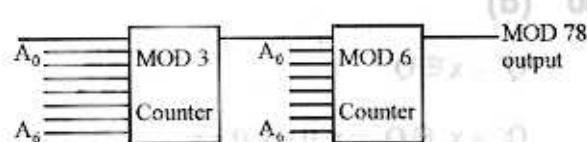
The value of J-K so that Q remains 1 can be found from the truth table of JK flip-flop

| Q | J | K | $Q^+$ |
|---|---|---|-------|
| 1 | 0 | 0 | 1     |
| 1 | 1 | 0 | 1     |

Here we see that K should be 0 and J can be either 0 or 1. i.e., don't care state.

**S.20 (b)**

One mod-13 counters followed by the mod-6 counter will result in mod-N counter where N is divisible by both 13 and 6 i.e., 78

**S.21 (a)**

| $Q_n$ | $Q_{n+1}$ | $T_n$ |
|-------|-----------|-------|
| 0     | 0         | 0     |
| 0     | 1         | 1     |
| 1     | 0         | 1     |
| 1     | 1         | 0     |

From the table  $Q^+ = T\bar{Q} + \bar{T}Q$

**S.22 (b)**

Only the carry flag is affected by the DAD B operation. Therefore, the sign, zero, auxiliary carry, and even parity flag conditions are not predictable. The carry flag (CY) will be set to 1 because of the overflow or carry during the addition ( $FFFF + 0001 = 1\ 0000H$ ).

**S.23 (a)**

All flip-flop in the counter will be cleared as soon as the states 1100 ( $12_{10}$ ) is reached. Hence, counter is allowed from 0000 to 1100 sequence of state and the reset states are skipped. It is Mod-12 counter.

**S.24 (b)**

Frequency of output  $Q_D$  is  $1 \text{ MHz}/12 = 833 \text{ kHz}$

**S.28 (d)**

Given FF is a negative edge triggered T flip-flop. So at the negative edge of clock  $v_i$  FF will invert the output if there is 1 at input.

**S.29 (d)**

| x | Q | S | R | $Q^+$ |
|---|---|---|---|-------|
| 0 | 0 | 0 | 1 | 0     |
| 0 | 1 | 1 | 0 | 1     |
| 1 | 0 | 1 | 0 | 1     |
| 1 | 1 | 0 | 1 | 0     |

$$\text{Hence } Q^+ = x \oplus Q$$

**S.30 (d)**

$$Q^+ = x \oplus Q$$

$$Q_1^+ = x_1 \oplus Q_0 = x_1 \bar{0} + \bar{x}_1 0 = x_1$$

$$Q_2^+ = x_2 \oplus x_1$$

$$Q_3^+ = x_3 \oplus x_2 \oplus x_1$$

$$Q_4^+ = x_4 \oplus x_3 \oplus x_2 \oplus x_1$$

So this generate the even parity and check odd parity.

**S.31 (b)**

| A | B | X | Y |
|---|---|---|---|
| 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |

X and Y are fixed at 0 and 1.

**S.32 (d)**

In ripple counter delay  $4T_d = 40 \text{ ns}$ .

The synchronous counter are clocked simultaneously, then its worst delay will be equal to  $10 \text{ ns}$ .

**S.33 (b)**

The maximum operating frequency is given by

$$\frac{1}{T} \geq N(t_d) + T_s$$

N : number of flip flop = 3

$t_d$  : propagation delay = 25 nsec

$T_s$  : strobe pulse width = 10 nsec

$$\therefore f_{\max} = \frac{1}{Nt_d + T_s}$$

$$= \frac{1}{3 \times 25 \times 10^{-9} + 10 \times 10^{-9}} \\ = 11.76 \text{ MHz}$$

**S.34 (c)**

Ring counter is  $N : 1$  divider

$$\therefore a = \frac{100 \times 10^3}{10} = 10 \text{ kHz}$$

MOD - 20 is a divide by 20 counter

$$\therefore b = \frac{10 \times 10^3}{20} = 0.5 \text{ kHz} = 500 \text{ Hz}$$

4 bit parallel counter is a  $2^4 = 16$ , MOD 16 counter

$$\therefore c = \frac{500}{16} = 31.25 \text{ Hz}$$

4 bit Johnson counter is a  $2N : 1$  divider

$$\therefore d = \frac{31.25}{8} = 3.9 \text{ Hz}$$

**S.35 (d)**

Drawing a stable table : (Let input be X)

| Present State | Next State |       | Output State |       |
|---------------|------------|-------|--------------|-------|
|               | X = 0      | X = 1 | X = 0        | X = 1 |
| a             | c          | b     | 0            | 0     |
| b             | d(f)       | c     | 0            | 0     |
| c             | g(e)       | d(f)  | 1            | 1     |
| d             | e          | f     | 1            | 0     |
| e             | f          | a     | 0            | 1     |
| f             | g(e)       | f     | 1            | 0     |
| g             | f          | a     | 0            | 1     |

State c and g are equivalent because the next states and outputs are same. We can eliminate any one of them, say 'g'. Thus 'g' is replaced by 'c' wherever it occurs. Now states d and f are equivalent. Thus one of them say d, can be eliminated. Thus the reduced state table :-

| Present State | Next State |       | Output State |       |
|---------------|------------|-------|--------------|-------|
|               | X = 0      | X = 1 | X = 0        | X = 1 |
| a             | c          | b     | 0            | 0     |
| b             | f          | c     | 0            | 0     |
| c             | e          | f     | 1            | 1     |
| e             | f          | a     | 0            | 1     |
| f             | e          | f     | 1            | 0     |

### S.36 (c)

The propagation delay of one flip-flop is 20 nsec. So for a synchronous counter delay is 20 nsec and for 4 flip-flops in asynchronous mode is  $20 \times 4 = 80$  nsec.

### S.37 (d)

It completes one cycle = 1024 pulses

$$\text{For } \frac{2048}{1024} = 2 \text{ cycles}$$

$$\begin{aligned} \text{Balance} &= 2060 - 2048 \\ &= 12 \text{ pulses} \end{aligned}$$

Binary 12 = 000 000 1100

### S.38 (c)

The given

$$V_{CC} = 5 \text{ V}$$

$$I_{CCL} = 3.6 \mu\text{A}$$

$$I_{CCH} = 2.8 \mu\text{A}$$

The average current drawn is

$$\begin{aligned} I_{CC(\text{avg})} &= \frac{I_{CCL} + I_{CCH}}{2} \\ &= \frac{3.6 + 2.8}{2} \end{aligned}$$

$$I_{CC(\text{avg})} = \frac{6.4}{2} = 3.2 \mu\text{A}$$

$$\begin{aligned} \text{So the power dissipation is } P_A &= V_{CC} I_{CC(\text{avg})} \\ &= (5 \text{ V})(3.2 \mu\text{A}) \\ &= 16 \mu\text{W} \end{aligned}$$

### S.39 (c)

Count down: The 2-bit count down will be 11, 10, 01, 00... This can be achieved by XORing with 1 output of 2 bit counter.

$$00 \oplus 11 = 11$$

For e.g.

$$01 \oplus 11 = 10$$

### S.40 (d)

Working backwards

| D <sub>in</sub> | XOR2 | Q <sub>0</sub> | XOR1 | Q <sub>2</sub> | Q <sub>3</sub> | Q <sub>1</sub> |
|-----------------|------|----------------|------|----------------|----------------|----------------|
| 1               | 1    | 0              | 1    | 1              | 0              | 0              |

∴ It should be initialized to 0010

| D <sub>in</sub>       | Q <sub>0</sub> | Q <sub>1</sub> | Q <sub>2</sub> | Q <sub>3</sub> | XOR1 | XOR2 | D <sub>in</sub> |
|-----------------------|----------------|----------------|----------------|----------------|------|------|-----------------|
| Initial               | 0              | 0              | 1              | 0              | 1    | 1    | 1               |
| 1 <sup>st</sup> clock | 1              | 0              | 0              | 1              | 1    | 0    | 0               |

### S.41 (d)

$$D = \bar{X}Q_n + \bar{Q}_nY$$

Truth table

| X | Y | Q <sub>n</sub> | D | Q <sub>n+1</sub> |
|---|---|----------------|---|------------------|
| 0 | 0 | 0              | 0 | 0                |
| 0 | 0 | 1              | 1 | 1                |
| 0 | 1 | 0              | 1 | 1                |
| 0 | 1 | 1              | 1 | 1                |
| 1 | 0 | 0              | 0 | 0                |
| 1 | 0 | 1              | 0 | 0                |
| 1 | 1 | 0              | 1 | 1                |
| 1 | 1 | 1              | 0 | 0                |

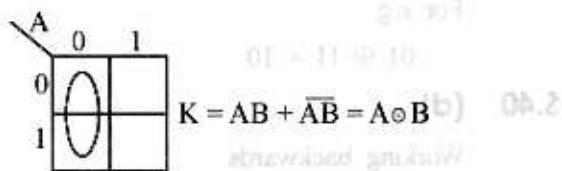
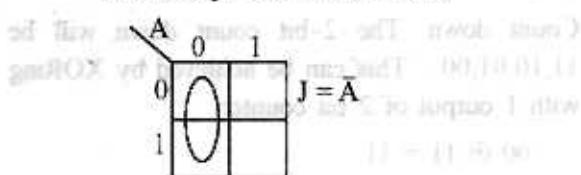
So we can see that circuit works as JK flip-flop with X = K and Y = J.

### S.42 (a)

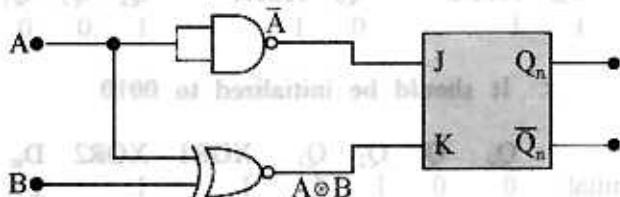
The state table is as follows.

| A | B | J | K | Q <sub>n</sub> | Q <sub>n+1</sub> |
|---|---|---|---|----------------|------------------|
| 0 | 0 | 1 | 1 | Q <sub>n</sub> | $\bar{Q}_n$      |
| 0 | 1 | 1 | 0 | X              | 1                |
| 1 | 0 | 0 | 0 | Q <sub>n</sub> | Q <sub>n</sub>   |
| 1 | 1 | 0 | 1 | X              | 0                |

The k-maps for J and K are.



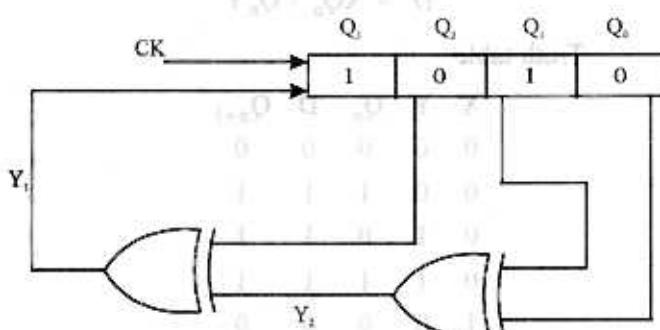
The circuit implementation is



S.45 (c)

$$Y_2 = Q_1 \oplus Q_0$$

$$Y_1 = Q_2 \oplus Y_2$$



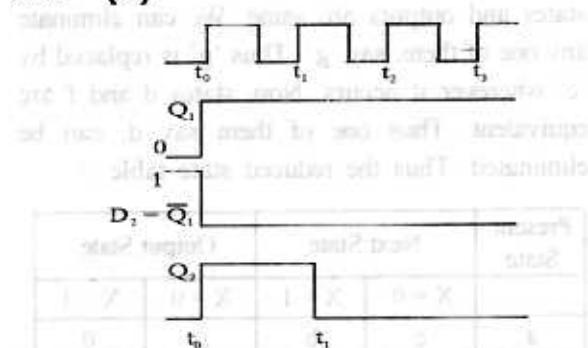
The sequences are as follows:

|   | $Q_3 (Y_1)$ | $Q_2$ | $Q_1$ | $Q_0$ | $Y_1$ | $Y_2$ |
|---|-------------|-------|-------|-------|-------|-------|
| 1 | 1           | 0     | 1     | 0     | 1     | 1     |
| 2 | 1           | 1     | 0     | 1     | 0     | 1     |
| 3 | 0           | 1     | 1     | 0     | 0     | 1     |
| 4 | 0           | 0     | 1     | 1     | 0     | 0     |
| 5 | 0           | 0     | 0     | 1     | 1     | 1     |
| 6 | 1           | 0     | 0     | 0     | 0     | 0     |
| 7 | 0           | 1     | 0     | 0     | 1     | 0     |
|   | 1           | 0     | 1     | 0     |       |       |

As we see from the table, sequence 1010 repeats after 7 cycles.

(d) 8E.2

S.46 (d)



S.47 (a)

| $\bar{Q}_0$ | $Q_0$ | $Q_2$ | $\bar{Q}_2$ | $Q_1$ | $\bar{Q}_1$ | Output |   |   |
|-------------|-------|-------|-------------|-------|-------------|--------|---|---|
| $J_2$       | $K_2$ | $J_1$ | $K_1$       | $J_0$ | $K_0$       | 1      | 0 | 1 |
| 0           | 1     | 1     | 0           | 0     | 1           | 0      | 1 | 0 |
| 1           | 0     | 0     | 1           | 1     | 0           | 1      | 0 | 1 |
| 0           | 1     | 1     | 0           | 0     | 1           | 0      | 1 | 0 |
| 1           | 0     | 0     | 1           | 1     | 0           | 1      | 0 | 1 |

We see that 101 repeat after every two cycles

Hence frequency will be  $\frac{f_c}{2}$

S.48 (a)

The output of XOR gate  $Z = b_{i+1} \oplus b_i$  and this output shift the register to left,

initially  $Z = 0$

After 1st clock  $Z = b_7 \oplus 0 = b_7$

After 2nd clock  $Z = b_7 \oplus b_6$

3rd clock  $Z = b_6 \oplus b_5$

4th clock  $Z = b_5 \oplus b_4$

S.49 (b)

| A | B | $Q_{n+1}$ | J |
|---|---|-----------|---|
| 0 | 0 | $Q_n$     | 0 |
| 0 | 1 | 1         | 1 |
| 1 | 0 | $Q_n$     | 0 |
| 1 | 1 | 0         | X |

$$J = \bar{A}B + AB$$

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

S.50 (b)

maximum operating frequency is

$$f_{\max} = \frac{1}{T_c} \leq \frac{1}{t_{\text{setup}} + t_{\text{pd(FF)}} + t_{\text{ns}}}$$

$t_{\text{setup}}$  : Time required for data input to settle before triggering edge

$t_{ns}$  : Next state i.e. Decoder propagation delay

$$\frac{1}{T_c} \leq \frac{1}{2+0.2+0.1+0.1}$$

$$\frac{1}{T_c} \leq \frac{1}{2.4 \text{ n sec}}$$

$$f_{\max} = \frac{1}{T_c} \leq \frac{1}{2.4 \times 10^{-9}}$$

$$\therefore f_{\max} = 0.416 \times 10^9 \approx 400 \text{ MHz}$$

S.51 (d)

Here  $D_A = Q_D \oplus Q_C$ ;  $D_B = Q_A$ ;  $D_C = Q_B$ ;  $D_D = Q_C$

For the given input string, outputs are  $Q_D$ ,  $Q_C$ ,  $Q_B$ ,  $Q_A$ . Therefore right most bit is  $Q_A$  and left most bit is  $Q_D$ . Now we determine the string sequence according to the above mentioned equations.

| (i) | $Q_D$ | $Q_C$ | $Q_B$ | $Q_A$ |
|-----|-------|-------|-------|-------|
|     | 0     | 0     | 0     | 1     |
|     | 0     | 0     | 1     | 0     |
|     | 0     | 1     | 0     | 0     |
|     | 1     | 0     | 0     | 1     |

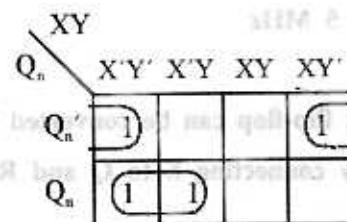
| (ii) | $Q_D$ | $Q_C$ | $Q_B$ | $Q_A$ |
|------|-------|-------|-------|-------|
|      | 0     | 0     | 1     | 1     |
|      | 0     | 1     | 1     | 0     |
|      | 1     | 1     | 0     | 1     |
|      | 1     | 0     | 1     | 0     |

| (iii) | $Q_D$ | $Q_C$ | $Q_B$ | $Q_A$ |
|-------|-------|-------|-------|-------|
|       | 1     | 1     | 1     | 0     |
|       | 1     | 1     | 0     | 0     |
|       | 1     | 0     | 0     | 0     |
|       | 0     | 0     | 0     | 1     |

S.52 (a)

Characteristic equation of JK Flip-flop is

| <b>Q<sub>n</sub></b> | <b>X</b> | <b>Y</b> | <b>Q<sub>n+1</sub></b> |
|----------------------|----------|----------|------------------------|
| 0                    | 0        | 0        | 1                      |
| 0                    | 0        | 1        | 0                      |
| 0                    | 1        | 0        | 1                      |
| 0                    | 1        | 1        | 0                      |
| 1                    | 0        | 0        | 1                      |
| 1                    | 0        | 1        | 1                      |
| 1                    | 1        | 0        | 0                      |
| 1                    | 1        | 1        | 0                      |



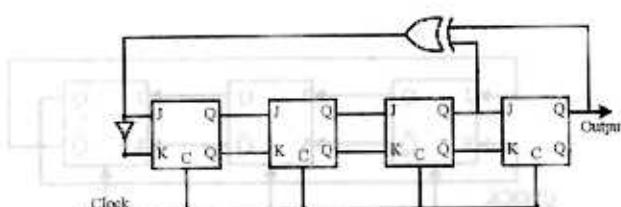
The characteristic equation becomes:

$$A = X'Q_a + Y'Q_b \quad \dots \dots \text{(ii)}$$

On comparing equation (i) and (ii)

We get  $\mathbf{J} = \mathbf{Y}'$

S.53 (c)



As can be seen, from the circuit the sequence at the output followed by the circuit is:

D C B A O/p

1 1 1 1 → 1

0 1 1 1 1

0 0 1 1 1

0 0 0 1 1

1 0 0 0 0

0 1 0 0 0

0 0 1 0 0

1 0 0 1 1

1 1 0 0 0

0 1 1 0 0

1 0 1 1 1

0 1 0 1 1

1 0 1 0 0

1 1 0 1 1

1 1 1 0 0

#### S.54 (c)

$$\text{Total delay} = N \cdot T_d = 4 \times 50 \times 10^{-9}$$

$$\therefore f = 5 \text{ MHz}$$

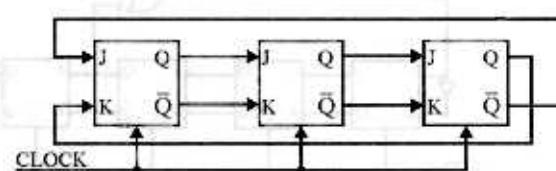
#### S.56 (d)

An SR flip-flop can be converted into T flip-flop by connecting S to Q and R to  $\bar{Q}$ .

#### S.57 (c)

The output states:-

000  
100  
110 } MOD - 6 Counter  
011  
001  
000



#### S.59 (a)

| Present State | $\Delta S$ |         | Output  |         |
|---------------|------------|---------|---------|---------|
|               | $x = 0$    | $x = 1$ | $x = 0$ | $x = 1$ |
| A             | D          | B       | 0       | 1       |
| B             | A          | C       | 0       | 1       |
| C             | A          | B       | 0       | 1       |
| D             | A          | C       | 1       | 1       |

Since no state reduction is possible and therefore four states are required.

#### S.60 (a)

(A)  $\rightarrow$  (3)

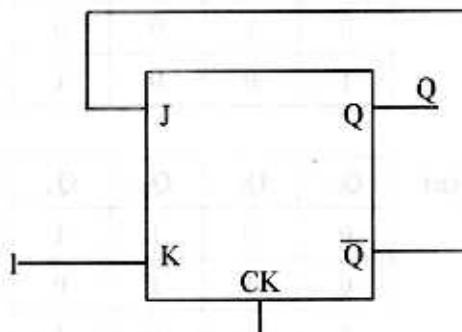
(B)  $\rightarrow$  (4)

(C)  $\rightarrow$  (2)

#### S.61 (b)

Since a pulse train is applied to the shift register it should be serial input shift register. Now we want an integral number of clock period delay, we should have parallel output so that any number of clock periods can be introduced.

#### S.62 (d)



| $J = \bar{Q}$ | $K = 1$ | $Q_n$ | $\bar{Q}_n$ | Clk |
|---------------|---------|-------|-------------|-----|
| -             | -       | 0     | 1           |     |
| 1             | 1       | 1     | 0           | 1   |
| 0             | 1       | 0     | 1           | 2   |
| 1             | 1       | 1     | 0           | 3   |
| 0             | 1       | 0     | 1           | 4   |
| 1             | 1       | 1     | 0           | 5   |
| 0             | 1       | 0     | 1           | 6   |

**S.64 (c)**

| $J_A = \bar{Q}_B$ | $K_A = 1$ | $J_B = Q_A$ | $K_B = 1$ | $Q_A$ | $Q_B$ |
|-------------------|-----------|-------------|-----------|-------|-------|
| X                 | X         | X           | X         | 0     | 0     |
| 1                 | 1         | 0           | 1         | 1     | 0     |
| 1                 | 1         | 1           | 1         | 0     | 1     |
| 0                 | 1         | 0           | 1         | 0     | 0     |

MOD -3

**S.65 (b)**

| CLK | J | K | D | P | Q |
|-----|---|---|---|---|---|
| -   | 1 | 1 | 0 | 0 | 1 |
| 1   | 1 | 1 | 1 | 1 | 1 |
| 2   | 1 | 1 | 0 | 0 | 0 |
| 3   | 1 | 1 | 1 | 1 | 1 |

**S.66 (c)**

The truth table is shown below:

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

Comparing from the truth table of J-K FF

$$Y = J, X = K$$

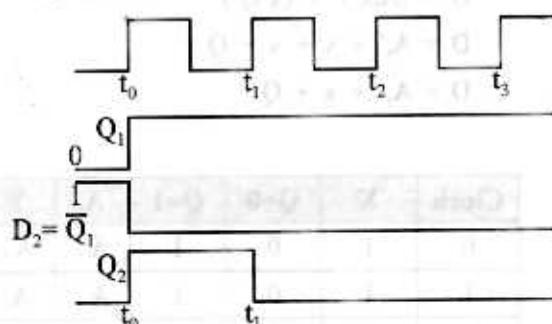
**S.67 (d)**

Given counter is Mod-10 up counter

$$\therefore \text{Frequency of O/P} = \frac{10K}{10} = 1K$$

**S.68 (d)**

The output of options (d) satisfy the given conditions

**S.69 (a)**The propagation delay of each inverter is  $t_{pd}$  then the fundamental frequency of oscillator output is

$$f = \frac{1}{2nt_{pd}}$$

$$= \frac{1}{2 \times 5 \times 100 \times 10^{-12}} = 1 \text{ GHz}$$

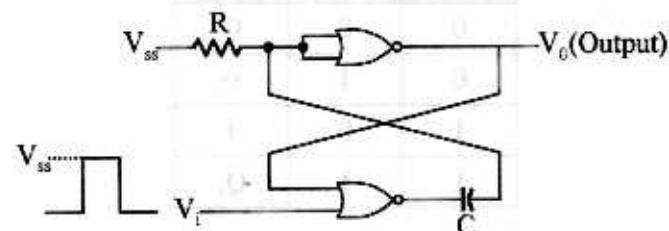
**S.70 (b)** $G_1$  has delay  $\rightarrow 10 \text{ nsec}$  $G_2$  has delay  $\rightarrow 20 \text{ nsec}$ 

Considering the delays of both gates the table will be

| Input | Output of $G_1$ | Output of $G_2$ |
|-------|-----------------|-----------------|
| 0     | 1               | 1               |
| 1     | 1               | 1               |
| 1     | 0               | 1               |
| 1     | 0               | 0               |
| 1     | 0               | 1               |

**S.71 (d)**

The circuit is as shown below



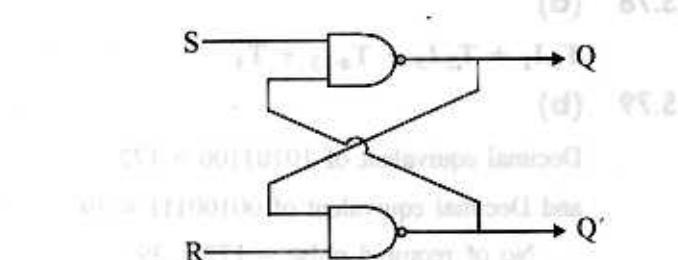
The circuit shown is monostable multivibrator as it requires an external triggering and it has one stable and one quasistable state.

**S.72 (d)**

Sequence is 0-2-3-1-0 are 00-10-11-01-00.

The input  $x = Q_1 \oplus Q_2$ , which generate the sequence 00-10-11-01-00.**S.73 (c)**

SR latch by cross-coupling two NAND gates



If the both S and R inputs are set 0.

Then the result will be:

Table for SR Latch using the NAND gates:

| S | R | Q                                                     |
|---|---|-------------------------------------------------------|
| 0 | 0 | Invalid state<br>$(Q = Q' = 1)$                       |
| 0 | 1 | 1                                                     |
| 1 | 0 | 0                                                     |
| 1 | 1 | Previous state $\therefore Q = 1 \text{ and } Q' = 1$ |

**S.74 (b)**

$$2^n \geq 100$$

$$\therefore n \geq 7$$

**S.76 (c)**

Master and slave flip-flops are isolated due to inverter at their clock input.

| X | Y | Z           |
|---|---|-------------|
| 0 | 0 | Q           |
| 0 | 1 | 0           |
| 1 | 0 | 1           |
| 1 | 1 | $\bar{Q}_1$ |

**S.77 (a)**

In the modulo-6 ripple counter at the end of sixth pulse (i.e., after 101 or at 110) all states must be cleared. Thus when CB is 110 then all states must be cleared. The input to 2-input gate is  $\bar{C}$  and  $\bar{B}$  and the desired output should be low since the CLEAR is active low.

Thus when  $\bar{C}$  and  $\bar{B}$  are 0, 0 then output must be 0. In all other case the output must be 1. OR gate can implement this function.

**S.78 (a)**

$$T_1.I_1 + T_2.I_3 + T_4.I_3 + T_3$$

**S.79 (b)**

Decimal equivalent of 10101100 = 172  
and Decimal equivalent of 00100111 = 39  
 $\therefore$  No of required pulse = 172 - 39

$$= 133$$

**S.80 (b)**

Transition table for the diagram is:

(d) 05.2

| PS       | 0        | 1        |
|----------|----------|----------|
| $Q_0(0)$ | $Q_0, 0$ | $Q_1, 1$ |
| $Q_1(1)$ | $Q_1, 1$ | $Q_0, 0$ |

Output function z is

| PS | 0 | 1 |
|----|---|---|
| 0  | 0 | 1 |
| 1  | 1 | 0 |

$\therefore$  The output z compute the 2's complement of the input number.

**S.81 (d)**

$$D = AX + X'Q'$$

| $Q_0$ | $Q_1$ | N | S |
|-------|-------|---|---|
| 0     | 0     | 1 | 1 |
| 1     | 1     | 0 | 1 |
| 0     | 1     | 1 | 0 |
| 1     | 0     | 0 | 0 |
| 0     | 0     | 1 | 1 |
| 1     | 1     | 0 | 1 |
| :     | :     | : | : |

**S.82 (a)**

The equation for D is

$$D = (A_i x)' + (x'Q)'$$

$$D = A_i' + x + x + Q$$

$$D = A_i' + x + Q$$

| Clock | X | $Q=0$ | $Q=1$ | $A_i$ | Y     |
|-------|---|-------|-------|-------|-------|
| 0     | 1 | 0     | 1     | $A_0$ | $A_0$ |
| 1     | 1 | 0     | 1     | $A_1$ | $A_0$ |
| 2     | 0 | 0     | 1     | $A_2$ | $A_1$ |
| 3     | 1 | 0     | 1     | $A_3$ | $A_1$ |
| 4     | 1 | 0     | 1     | $A_4$ | $A_3$ |
| 5     | 0 | 0     | 1     | $A_5$ | $A_4$ |

So, from clock 1 to 5 the output is  $A_0, A_1, A_1, A_3, A_4$ .

**S.83 (c)**

$$Q_2 Q_1 Q_0 = 011$$

$$\text{1st Clk} \rightarrow Q_2 Q_1 Q_0 = 100$$

$\bar{Q}_0 = 1$  (triggers  $T_1$ )

$\bar{Q}_1 = 1$  (triggers  $T_2$ )

**S.84 (b)**

$$I_{D_1} = I_{D_2}$$

$$K_1(V_{G_1} S_1 - V_1)^2 = K_2(V_{G_2} S_2 - V_1)^2$$

$$\Rightarrow 36(5 - V_0 - 1)^2 = 9(V_0 - 0 - 1)^2$$

$$\Rightarrow V_0 = 3V$$

**S.85 (b)**

Since  $J = 1$  and  $Q_n = 0$  So  $Q_{n+1} = 1$

as even if  $K = 0$ ,  $Q_{n+1} = 1$  (set)

and if  $K = 1$ ,  $Q_{n+1} = \overline{Q_n} = 1$  (toggle)

**S.86 (a)**

$$f(w, x, y, z) = \Sigma(5, 7, 11, 12, 13, 15)$$

Draw the K-map

|    | 10 | 01 | 11 | 10 |
|----|----|----|----|----|
| 00 |    |    |    |    |
| 01 |    | 1  | 1  |    |
| 11 | 1  | 1  | 1  |    |
| 10 |    |    | 1  |    |

$$w'xz \rightarrow 0101 \quad (5)$$

$$wxy' \rightarrow 1100 \quad (12)$$

$$xy'z \rightarrow 0101 \quad (5)$$

$$xyz \rightarrow 0111 \quad (7)$$

$$wyz \rightarrow 1011 \quad (11)$$

**S.87 (b)**

Clock with duty cycle 50%. In (b) there are 2 D flip-flop is leftmost is activated or clock value 1. The clock value of rightmost 1 D f/f is 0. So it delay the phase f by  $180^\circ$ .

**S.88 (d)**

Taking  $D_0 = Q'_1$  and  $D_1 = Q_0$  we get the desired sequence.

**S.89 (c)**

| A | B | C <sub>i</sub> | S | C <sub>0</sub> |                            |
|---|---|----------------|---|----------------|----------------------------|
| 1 | 1 | 0              | 0 | 1              | → after first clock pulse  |
| 1 | 1 | 1              | 1 | 1              | → after second clock pulse |

**S.90 (d)**

The given counter counts next,

Only when count = 1

Load = 0

Clear = 0

If clear input is '0'.

The counter get clear to zero.

Clear input is '1' when both input of and gate are 1 (i.e.,  $A_3 = A_1 = 1$ ).

Hence the count sequence of given circuit is

| A <sub>4</sub> | A <sub>3</sub> | A <sub>2</sub> | A <sub>1</sub> |                         |
|----------------|----------------|----------------|----------------|-------------------------|
| 0              | 0              | 0              | 0              | ⇒ [0 initially (given)] |
| 0              | 0              | 0              | 1              | ⇒ 1                     |
| 0              | 0              | 1              | 0              | ⇒ 2                     |
| 0              | 0              | 1              | 1              | ⇒ 3                     |
| 0              | 1              | 0              | 0              | ⇒ 4                     |
| 0              | 1              | 0              | 1              | ⇒ 5                     |

Now  $A_3 = A_1 = 1$ . Hence output of AND gate = 0.

∴ Counter get clear to 0.

Hence sequence becomes 0, 1, 2, 3, 4, 5.

**S.91 (b)**

For X = 0, Y = 1      P = 1, Q = 0

For X = 0, Y = 0      P = 1, Q = 1

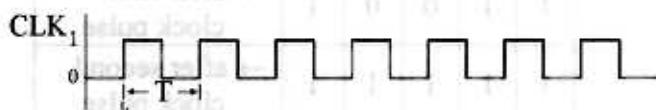
For X = 1, Y = 1      P = 1, Q = 0 or P = 0,  
Q = 1

## S.92 (d)

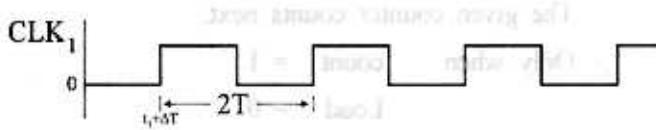
Begin with set condition i.e.,  $Q = 1$  and  $Q' = 0$ .

## S.93 (c)

Since the input to both JK flip-flop is 11, the output will change every time with clock pulse. The input to clock is:

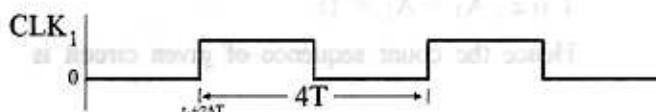


The output  $Q_0$  of first FF occurs after time  $\Delta T$  and



it is as shown below.

The output  $Q_1$  of second FF occurs after time  $\Delta T$  when it gets input (i.e., after  $\Delta T$  from  $t_1$ ) and it is as shown below:



Time period of waveform of output at

$$Q_1 = 2 \times 2 \times T = 4T$$

Delay time at output  $Q_1 = 2\Delta T$

Note:

1. In case of  $n$  flip-flops in such case, Time period of last output waveform

$$= 2^n T$$

where  $T$  = Time period for clock pulse

2. Delay time =  $n\Delta T$

where  $\Delta T$  = Propagation delay provided by one flip-flop

$$0 = Q_1, 1 = 1$$

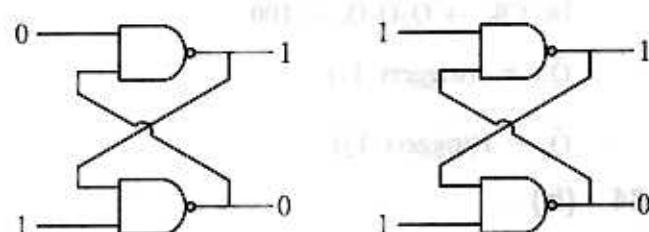
$$1 = Q_1, 0 = 0$$

$$0 = Q_1 \Rightarrow 0 = Q_1, 1 = 1$$

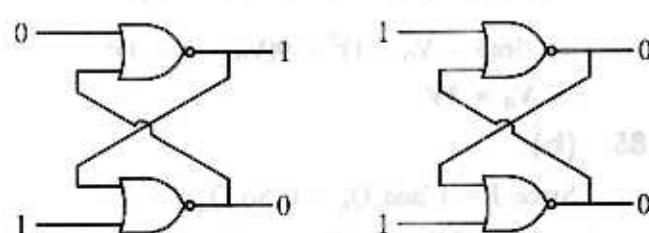
$$1 = 0$$

## S.94 (d)

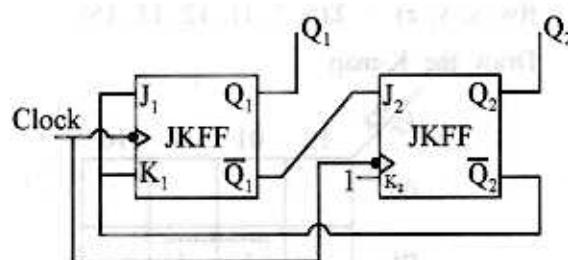
For the NAND latch the stable states are as follows:



For the NOR latches the stable states are as follows:



## S.95 (b)



| Clock | J <sub>1</sub> | K <sub>1</sub> | J <sub>2</sub> | K <sub>2</sub> | Q <sub>1</sub> | Q <sub>2</sub> |
|-------|----------------|----------------|----------------|----------------|----------------|----------------|
| 0     | 1              | 1              | 1              | 1              | 0              | 0              |
| 1     | 1              | 1              | 1              | 1              | 1              | 1              |
| 2     | 0              | 0              | 0              | 1              | 1              | 0              |
| 3     | 1              | 1              | 1              | 1              | 0              | 0              |



**UNIT-3**

**COMPUTER**

**ORGANIZATION AND**

**ARCHITECTURE**

**Unit-3**

**COMPUTER  
ORGANIZATION AND  
ARCHITECTURE**

In computer architecture, the term "level-1 cache" refers to a fast, small, local memory that sits between the CPU and main memory. It is used to store frequently accessed data and instructions. Level-1 cache is typically implemented using SRAM or DRAM technology. It is located directly on the CPU chip and has a very low access latency compared to main memory. The size of level-1 cache varies depending on the processor type and manufacturer.

# OVERVIEW OF COMPUTER ARCHITECTURE

**3.1**

## LEVEL-1

**Q.1** Which of the following statements is correct?

- (i) There are distinct representation for +0 and -0 in both the sign and magnitude and 1's complement system.
- (ii) 2's complement has only one representation for 0.
- (a) (i)
- (b) (ii)
- (c) (i) and (ii)
- (d) Neither (i) nor (ii)

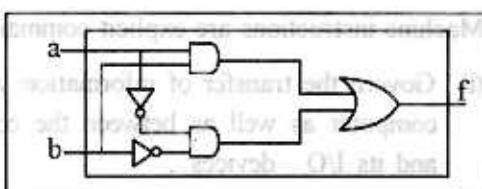
**Q.2** Coprocessor

- (i) is designed to provide fast, low cost hardware implementation for complicated arithmetic functions.
- (ii) is a separate instruction set processor that is closely coupled to the CPU.
- (iii) is a processor whose instructions and registers are direct extensions of the CPU.
- (a) (i), (iii)
- (b) (ii), (iii)
- (c) (i), (ii)
- (d) (i), (ii), (iii)

**Q.3** Which of the following statement is correct about Booth's algorithm in general form \_\_\_\_\_

- (a) In general, in the booth scheme -1 times the shifted multiplicand is selected when moving from 0 to 1
- (b) In general, in the booth scheme +1 time the shifted multiplicand is selected when moving from 1 to 0
- (c) Neither (a) nor (b)
- (d) Both (a) and (b)

**Q.4** A machine XYZ, performs \_\_\_\_\_ boolean operation with the help of following circuit?



- (a)  $\text{bool } f = (a \&\& b) || (\neg a \&\& \neg b)$
- (b)  $\text{bool } f = (a \&\& b) || (\neg a \&\& \neg b)$
- (c)  $\text{bool } f = (a \&\& b!) || (\neg a \&\& \neg b)$
- (d)  $\text{bool } f = (a \&\& b!) || (\neg a \&\& \neg b)$

**Q.5** Memory in which any location can be reached in a short and fixed amount of time specifying its address is called

- (a) Cache memory
- (b) RAM
- (c) ROM
- (d) Virtual memory

**Q.6** The following description is related to temporal ALU expansion

Use one copy of the  $m$ -bit ALU chip in the manner of a serial adder to perform an operation on  $km$ -bit words in  $k$  consecutive clock cycles. In each cycle, the ALU processes a separate  $m$ -bit slice of each operand.

Then, such a processing is called as \_\_\_\_\_

- (a) multicycle processing
- (b) multiple precision processing
- (c) Both (a) and (b)
- (d) Neither (a) nor (b)

**Q.7** \_\_\_\_\_ are used for the large scale numerical calculations required in applications such as weather forecasting and aircraft design and simulation.

- (a) mainframe
- (b) super computer
- (c) desktop computers
- (d) mini computer

**Q.8** Machine instructions are explicit commands that

- (i) Govern the transfer of information within a computer as well as between the computer and its I/O devices.
  - (ii) Specify the arithmetic and logic operations to be performed.
- (a) (i)
  - (b) (ii)
  - (c) Both (i) and (ii)
  - (d) Neither (i) nor (ii)

**Q.9** Which of the following is not an advantage of Booth's algorithm?

- (i) Booth's algorithms handles both positive and negative multipliers uniformly.
  - (ii) Booth's algorithm achieves some efficiency in the number of additions required when the multiplier has a few large blocks of 1's.
  - (iii) The speed of doing multiplication by booth's algorithm is more than the normal algorithm of average.
- (a) (ii), (i)
  - (b) (iii)
  - (c) (ii), (iii)
  - (d) (i), (ii), (iii)

**Q.10** In case of 2's complement representation expansion of bit length is \_\_\_\_\_

- (a) Not possible
- (b) Possible by adding additional bit position to the left and fill in with the value of the original sign bit.
- (c) Possible by adding additional bit position to the right and fill in with the value of the original sign bit.
- (d) All above

**Q.11** To preserve accuracy during floating point calculations one or more extra bits are temporally attached to the right end of the mantissa, such bits are called as \_\_\_\_\_

- (a) Guard bits
- (b) Denormalized bits
- (c) Equalized bits
- (d) Normalized bits

**Q.12** The Booth algorithm \_\_\_\_\_

- (i) generates a  $2n$  bit product
  - (ii) treats both positive and negative 2's complements  $n$  bit operand uniformly.
- (a) (i)
  - (b) (ii)
  - (c) (i) and (ii)
  - (d) Neither (i) nor (ii)

**Q.13** The bandwidth of memory system is given by

- (a) Number of bits/total time
- (b) Total time/number of bits
- (c) number of bits  $\times$  total time
- (d) All

**Q.14** A mini-computer has

- (a) limited storage capacity ranging from 8000 to 16000 locations
- (b) relatively small basic data unit size within the storage i.e. usually 8 to 16 digits for internal storage word.
- (c) relatively low speed and low capacity auxiliary storage devices
- (d) all of these

**Q.15** One of the main feature that distinguish micro-processors from micro-computers is

- (a) exactly the same as the machine cycle time
- (b) micro-processor does not contain I/O devices
- (c) words are usually larger in micro-processor
- (d) none of the above

**Q.16** Advantage of multiplexing Address Bus with data bus

- (a) No. of pins increased
- (b) No. of pins reduced
- (c) Both
- (d) None of the above

**Q.17** Negative numbers cannot be represented in

- (a) signed magnitude form
- (b) 1's complement form
- (c) 2's complement form
- (d) None of the above

**Q.18** The XOR operator  $\oplus$  is

- (a) commutative
- (b) associative
- (c) distributive over AND operator
- (d) None of the above

**Q.19** How many 2-input multiplexers are required to construct a  $2^{10}$ -input multiplexer?

- (a) 1023
- (b) 31
- (c) 10
- (d) 127

**Q.20** The clock of a microprocessor can be divided by 5 using a

- (a) 3 bit counter
- (b) 5 bit counter
- (c) mod 5 counter
- (d) mod 3 counter

**Q.21** Suppose the largest n-bit binary number requires 'd' digits in decimal representation. Which of the following relations between 'n' and 'd' is approximately correct?

- (a)  $d = 2^n$
- (b)  $n = 2^d$
- (c)  $d < n \log_{10} 2$
- (d)  $d > n \log_{10} 2$

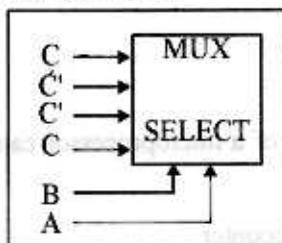
**Q.22** Floating point numbers in a computer are represented by a 10 bit mantissa (including sign bit) and a 6 bit exponent (including a sign bit). The approximate value of the maximum number that can be represented is (assume the mantissa is stored in normalized form)

- (a)  $2^{64}$
- (b)  $2^{63}$
- (c)  $2^{32}$
- (d)  $2^{31}$

**Q.23** Which of the following remarks about bit-slice processor is correct?

- (a) It can be cascaded to get any desired word length processor.
- (b) Its speed of operation is independent of the word length configured.
- (c) It does not contain anything equivalent to a program counter in a normal microprocessor.
- (d) It contains only the data path of a normal CPU.

**Q.24** The output of the multiplexer circuit in Figure can be represented by



- (a)  $AB + BC' + C'A + BC$
- (b)  $A + B + C$
- (c)  $A + B$
- (d)  $A'B'C + A'BC' + ABC$

## LEVEL-2

**Q.25** In excess -127 format when exponent  $E' = 225$  and mantissa  $M \neq 0$ , in such situation if we perform  $0/0$  or  $\sqrt{-1}$  then result is \_\_\_\_\_

- (a) Normalized
- (b) Denormalized
- (c) NaN
- (d) All

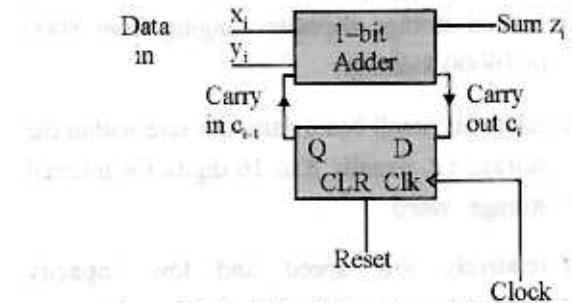
**Q.26** Which of the following enables a single processor to control a number of external devices such as keyboards, displays, magnetic and optical disks?

- (i) the control unit is faster than other devices connected to a computer system.
- (ii) the arithmetic unit is faster than other devices connected to a computer system.
- (iii) The logic unit is faster than other devices connected to a computer system.
- (a) (i), (ii)
- (b) (ii), (iii)
- (c) (i), (ii), (iii)
- (d) (i), (iii)

**Q.27** In the  $n \times n$  multiplication process using carry save addition, if bit pair recording of the multiplier is done then levels of carry save addition reduces from \_\_\_\_\_

- (a)  $1.7 \log_2 n - 1.7$  to  $1.7 \log_2 n - 3.4$
- (b)  $1.7 \log_2 n$  to  $-3.4 \log_2 n$
- (c)  $-1.7 \log_2 n$  to  $-3.4 \log_2 n$
- (d)  $1.7 \log_2 n$  to 0

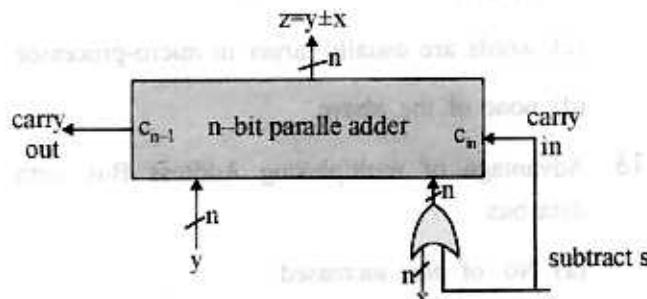
**Q.28** Consider the following figure



then which of the following statement is correct?

- (i) It works correctly on unsigned numbers
- (ii) It works correctly on positive numbers
- (a) (i)
- (b) (ii)
- (c) (i) and (ii)
- (d) All

**Q.29** The following circuit is best suited for \_\_\_\_\_



- (a) An n-bit 2's complement adder subtracter
- (b) An n-bit 1's complement adder subtracter
- (c) An n-bit parallel adder subtracter
- (d) All

**Q.30** In IEEE 754 format, the number of values in single extended format is \_\_\_\_\_ and double extended format is \_\_\_\_\_

- (a)  $1.98 \times 2^{31}, 1.99 \times 2^{63}$
- (b)  $1.98 \times 2^{31}$ , unspecified
- (c) unspecified,  $1.99 \times 2^{63}$
- (d) unspecified, unspecified

**Q.31** The speed imbalance between memory access and CPU operation can be reduced by

- (a) cache memory
- (b) memory interleaving
- (c) reducing the size of memory
- (d) None of the above

**Q.32**  $(10110011100011110000)_2$  in base 32 is

- (a) 22 14 7 16
- (b) 11 9 23 31
- (c) 11 9 7 16
- (d) 11 14 23 16

**Q.33** Choose the correct statements.

- (a) By scanning a bit pattern, one can say whether, it represents data or not.
- (b) Whether a given piece of information is a data or not depends on the particular application.
- (c) Positive numbers can't be represented in 2's complement form.
- (d) Positive numbers can't be represented in 1's complement form.

**Q.34** If  $(11A1B)_8 = (12c9)_{16}$  (c stands for decimal 12), then the values of A and B are

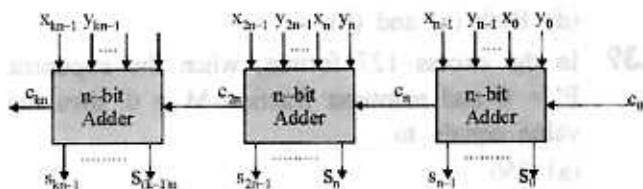
- (a) 5, 1
- (b) 7, 5
- (c) 5, 7
- (d) None of the above

**Q.35** A decimal number has 25 digits. The number of bits needed for its equivalent binary representation is, approximately,

- (a) 50
- (b) 60
- (c) 70
- (d) 75

## LEVEL-3

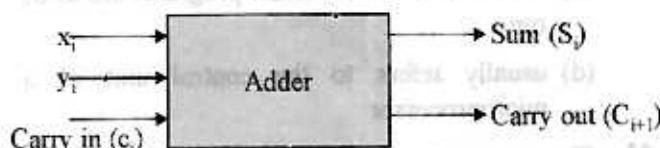
**Q.36** Consider the following circuitry used for addition purpose



Suppose the circuit to detect overflow is to be added in this, then such circuit is implemented by the logic expression \_\_\_\_\_

- (a) overflow =  $x_n y_n \bar{s}_n + \bar{x}_n \bar{y}_n s_n$
- (b) overflow =  $x_{n-1} y_{n-1} \bar{s}_n + \bar{x}_{n-1} \bar{y}_{n-1} s_n$
- (c) overflow =  $x_{n-1} y_{n-1} \bar{s}_{n-1} + \bar{x}_{n-1} \bar{y}_{n-1} s_{n-1}$
- (d) None of these

**Q.37** Consider the following adder circuit



we get the output as  $S_i = x_i \oplus y_i \oplus C_i$

and  $C_{i+1} = x_i y_i + x_i C_i + y_i C_i$  and let  $G_i$  and  $P_i$  are generate and propagate functions from it. then which of the following statement is incorrect?

- (a) If the generate function for stage i is equal to 1, then  $C_{i+1} = 1$  and it is independent of the input carry  $C_i$ .
- (b) The propagate function means that an input carry will produce an output carry when either  $x_i$  is 1 or  $y_i$  is 1.
- (c) All  $G_i$  and  $P_i$  functions can be formed independently and in parallel in one logic gate delay after the x and y vectors are applied to the inputs of an n-bit adder.
- (d) None of the above

**Q.38** In 2's complement representation, if two numbers with the same sign are added, then \_\_\_\_\_

- (a) overflow occurs if and only if the result has the opposite sign.
- (b) overflow occurs if and only if the result has same sign.
- (c) overflow does not occur.
- (d) Both (a) and (b)

**Q.39** In the excess-127 format, when the exponent  $E' = 0$  and mantissa fraction  $M = 0$ , then the value equals to \_\_\_\_\_

- (a) 250
- (b) 750
- (c) 0
- (d)  $\infty$

## GATE QUESTIONS

**Q.40** Microprogrammed control unit [GATE 1987]

- (a) is faster than a hard-wired control unit.
- (b) facilitates easy implementation of new instruction
- (c) is useful when very small programs are to be run
- (d) usually refers to the control unit of a microprocessor

**Q.41** The exponent of a floating-point number is represented in excess-N code so that

[GATE 1987]

- (a) the dynamic range is large
- (b) the precision is high
- (c) the smallest number is represented by all zeros
- (d) overflow is avoided

**Q.42** Bit-sliceprocessors [GATE 1992]

- (a) can be cascaded to get any desired word length processor
- (b) speed of operation is independent of the word length configured
- (c) don't contain anything equivalent of program counter in a 'normal' microprocessor
- (d) contain only the data path of a 'normal' CPU

**Q.43** In the following Pascal program segment, what is the value of X after the execution of the program segment?

X := -10; Y := 20;

If X > Y then if X < 0 then X := abs(X) else  
X := 2\*X;

[GATE 1995]

[1-Mark]

- (a) 10
- (b) -20
- (c) -10
- (d) None

**Q.44** The number of 1s in the binary representation of  $(2^8 \cdot 4096 + 15 \cdot 256 + 5 \cdot 16 + 3)$  [GATE 1995]

[2-Marks]

- (a) 8
- (b) 9
- (c) 10
- (d) 12

**Q.45** Booth's algorithm for integer multiplication gives worst performance when the multiplier pattern is

[GATE 1996]

- (a) 101010...1010
- (b) 100000...0001
- (c) 111111...1111
- (d) 011111...1110

**Q.46** Consider the following floating point number representation

[GATE 1996]

|          |    |          |   |
|----------|----|----------|---|
| 31       | 24 | 23       | 0 |
| Exponent |    | Mantissa |   |

The exponent is in 2's complement representation and mantissa is in the sign magnitude representation. The range of the magnitude of the normalized numbers in this representation is

- (a) 0 to 1
- (b) 0.5 to 1
- (c)  $2^{-23}$  to 0.5
- (d) 0.5 to  $(1 - 2^{-23})$

**Q.47** Given  $\sqrt{(224)_r} = (13)_r$

The value of the radix  $r$  is: [GATE 1997]  
[2-Marks]

- (a) 10
- (b) 8
- (c) 5
- (d) 6

**Q.48** The octal representation of an integer is  $(342)_8$ . If this were to be treated as an eight bit integer in an 8085 based computer, its decimal equivalent is [GATE 1998]

[1-Mark]

- (a) 226
- (b) -98
- (c) 76
- (d) -30

**Q.49** Booth's coding in 8 bits for the decimal number -57 is [GATE 1999]

[1-Mark]

- (a) 0 - 100 + 1000
- (b) 0 - 100 + 100 - 1
- (c) 0 - 1 + 100 - 10 + 1
- (d) 00 - 10 + 100 - 1

**Q.50** Zero has two representation in [GATE 1999]  
[1-Mark]

- (a) Sign magnitude
- (b) 1's complement
- (c) 2's complement
- (d) none of the above

**Q.51** The number 43 in 2's complement representation is [GATE 2000]

[1-Mark]

- (a) 01010101
- (b) 11010101
- (c) 00101011
- (d) 10101011

**Q.52** The decimal value 0.25 [GATE 2002]  
[1-Mark]

- (a) is equivalent to the binary value 0.1
- (b) is equivalent to the binary value 0.01
- (c) is equivalent to the binary value 0.00111...
- (d) cannot be represented precisely in binary

**Q.53** The 2's complement representation of decimal value -15 is [GATE 2002]

[1-Mark]

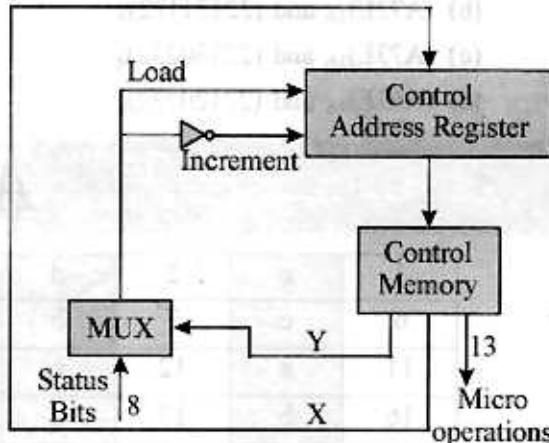
- (a) 1111
- (b) 11111
- (c) 111111
- (d) 10001

**Q.54** Sign extension is a step in [GATE 2002]

[1-Mark]

- (a) floating point multiplication
- (b) signed 16 bit integer addition
- (c) arithmetic left shift
- (d) converting a signed integer from one size to another

**Q.55** The microinstructions stored in the control memory of a processor have a width of 26 bits. Each microinstruction is divided into three fields: a micro-operation field of 13 bits, a next address field (X), and a MUX select field (Y). There are 8 status bits in the inputs of the MUX.



How many bits are there in the X and Y fields, and what is the size of the control memory in number of words? [GATE 2004]

[2-Marks]

- (a) 10, 3, 1024
- (b) 8, 5, 256
- (c) 5, 8, 2048
- (d) 10, 3, 512

**Q.56** Consider a multiplexer with X and Y as data inputs and Z as control input. Z = 0 selects input X, and Z = 1 selects input Y. What are the connections required to realize the 2-variable Boolean function  $f = T + R$ , without using any additional hardware? [GATE 2004]

[2-Marks]

- (a) T to X, R to Y, T to Z
- (b) R to X, I to Y, T to Z
- (c) T to X, R to Y, 0 to Z
- (d) R to X, 0 to Y, T to Z

**Q.57** Using a 4-bit 2's complement arithmetic, which of the following additions will result in an overflow?

- (i)  $1100 + 1100$
- (ii)  $0011 + 0111$

(iii)  $1111 + 0111$ 

[IT-GATE 2004]

- (a) (i) only
- (b) (ii) only
- (c) (iii) only
- (d) (i) and (iii) only

**Q.58** The number  $(123456)_8$  is equivalent to

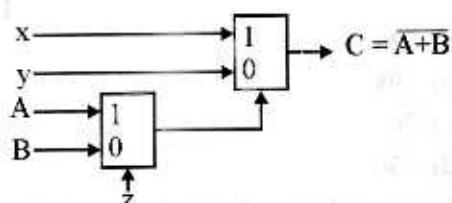
[IT-GATE 2004]

- (a)  $(A72E)_{16}$  and  $(22130232)_4$
- (b)  $(A72E)_{16}$  and  $(22131122)_4$
- (c)  $(A73E)_{16}$  and  $(22130232)_4$
- (d)  $(A62E)_{16}$  and  $(22120232)_4$

**Q.59**  $(34.4)_8 \times (23.4)_8$  evaluates to [GATE 2005] [2-Marks]

- (a)  $(1053.6)_8$
- (b)  $(1053.2)_8$
- (c)  $(1024.2)_8$
- (d) None of these

**Q.60** The circuit shown below implements a 2-input NOR gate using two 2-1 MUX (control signal 1 selects the upper input). What are the values of signals x,y and z?



[IT-GATE 2005]

[2-Marks]

- (a) 1, 0, B
- (b) 1, 0, A
- (c) 0, 1, B
- (d) 0, 1, A

## ANSWER KEY

|    |      |    |   |    |      |    |   |    |   |
|----|------|----|---|----|------|----|---|----|---|
| 1  | c    | 2  | d | 3  | d    | 4  | a | 5  | d |
| 6  | c    | 7  | b | 8  | c    | 9  | b | 10 | b |
| 11 | a    | 12 | c | 13 | a    | 14 | d | 15 | b |
| 16 | b    | 17 | d | 18 | a, b | 19 | a | 20 | c |
| 21 | d    | 22 | d | 23 | a    | 24 | b | 25 | c |
| 26 | c    | 27 | a | 28 | c    | 29 | a | 30 | d |
| 31 | a, b | 32 | a | 33 | b    | 34 | d | 35 | d |
| 36 | c    | 37 | d | 38 | a    | 39 | c | 40 | a |
| 41 | b, c | 42 | a | 43 | a    | 44 | c | 45 | a |
| 46 | d    | 47 | c | 48 | a    | 49 | b | 50 | a |
| 51 | b    | 52 | b | 53 | d    | 54 | d | 55 | a |
| 56 | b    | 57 | b | 58 | a    | 59 | a | 60 | a |

## SOLUTIONS

**S.1 (c)**

For representation of 0 in sign and magnitude, 1's complement and 2's complement format (i) and (ii) are correct.

**S.2 (d)**

(i), (ii), (iii) are correct about coprocessors.

**S.3 (d)**

In general in the booth scheme,  $-1$  times the shifted multiplicand is selected when moving from 0 to 1, and  $+1$  times the shifted multiplicand is selected when moving 1 to 0.

**S.4 (a)**

With the help of circuit we can perform

$$\text{bool } f = (a \&\& b) \mid\mid (!a \&\&!b)$$

**S.5 (d)**

As in most processors, virtual memory is implemented using cache memory for faster execution and searching of data item.

**S.6 (c)**

Definition of multicycle or multiple precision processing.

**S.7 (b)**

Definition and function of super computer.

**S.9 (b)**

The speed is gained by skipping over 1s depends on the data. On average, the speed of doing multiplication with the booth algorithm is the same as with normal algorithms.

**S.10 (b)**

In case of 2's complement representation expansion of bit length is possible by adding additional bit position to the left and fill in with the value of the original sign bit.

**S.11 (a)**

Definition of guard bits.

**S.12 (c)**

The booth algorithm generates a  $2n$ -bits product and treats positive and negative 2's complement  $n$  bit operands uniformly.

**S.16 (b)**

When the address bus is multiplexed with the data bus, number of pins on the microprocessor chip are reduced as the same microprocessor pins can be used for address bus as well as for data bus at different.

**S.18 (a, b)**

It is commutative because  $A \oplus B = B \oplus A$

It is associative because

$$(A \oplus B) \oplus C = A \oplus (B \oplus C)$$

It is not distributive over AND because

$$A \oplus (B \text{ AND } C) = (A \oplus B) \text{ AND } (A \oplus C)$$

is not true. For e.g.,  $1 \oplus (0 \text{ AND } 1) = 1$

$$\text{But } (1 \oplus 0) \text{ AND } (1 \oplus 1) = 1 \text{ AND } 0 = 0$$

**S.19 (a)**

A 2-input multiplexer can select a single line out of the two-input lines. To select a single line out of the  $2^{10}$ , i.e., 1024 input lines, we have to use 1023 two-input multiplexers. In order to select 512 lines out of the 1024, we need 512 two-input multiplexers. After this, to select 256 we need 256 two-input multiplexers. Continuing this way, to ultimately get a single line, we need a total of  $512 + 256 + 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 1023$  two-input multiplexers.

**S.21 (d)**

The largest ' $n$ ' bit binary number is  $2^n - 1$ . If its equivalent decimal number has ' $d$ ' digits then it has to be less than  $10^d$ . So,  $2^n - 1 < 10^d$ , i.e.  $2^n < 10^d$  (approximately), so  $d > n \log_{10} 2$ .

**S.25 (c)**

In excess-127 format, when exponent  $E' = 225$  and mantissa  $M \neq 0$ , then in such situation if we perform  $0/0$  or  $\sqrt{-1}$  the result is NaN i.e. not a number.

**S.26 (c)**

The control, the arithmetic and logic units are many times faster than other devices connected to a computer system. This enables a single processor to control a number of external devices such as keyboards, displays, magnetic and optical disks.

**S.27 (a)**

In the multiplication process using carry save addition, if bit pair recording of the multiplier is done then the number of summands is reduced to  $n/2$ . This reduces the number of carry save addition levels required from  $1.7 \log_2 n - 1.7$  to  $1.7 \log_2 n - 3.4$ .

**S.28 (c)**

The given circuit is for a serial binary adder and it works correctly on unsigned and positive numbers.

**S.30 (d)**

In IEEE 754 format, the number of values in single extended format is unspecified and in double extended format is also unspecified.

**S.32 (a)**

To convert to base 8, we group in 3's, because  $2^3 = 8$ .

To convert to base 16, we group in 4's, because  $2^4 = 16$ .

To convert to base 32, we group in 5's because  $2^5 = 32$ .

Grouping in 5's, from the right, we can get the answer.

**S.33 (b)**

The contents of a word may represent an instruction or data. Just by looking at the contents, it is not possible to attach any meaning to it. A word pointed to by the program counter, is an instruction. Otherwise it need not be. Also, the word data has context sensitive meaning. One can write a program in Pascal that needs radius as the input data. The program, as a whole, is input data for the compiler during the compilation process.

Positive no.'s can be represented in 1's complement as well as 2's complement.

**S.34 (d)**

Converting to base 2, the equation reads

$$001\ 001\ A\ 001\ B = 0001\ 0010\ 1100\ 1001$$

Here A, B stand for a group of binary digits. So, grouping the right hand side in 3's, from the right and matching corresponding groups in both the sides, we get B = 001 and A = 011. So, A = 3 and B = 1.

**S.35 (d)**

The number M will be such that  $10^{24} \leq M < 10^{25}$ . A decimal number Y, needs approximately  $\log x$  (log is to the base 2) number of bits. So binary representation of M needs more than  $\log 10^{24}$  bits, but less than  $\log 10^{25}$  bits.  $\log 10^{24}$  is greater than  $\log 8^{24}$  ( $= \log 2^{72} = 72 \log 2 = 72$ ). So, more than 72 bits are needed. The nearest answer is 75.

**S.36 (c)**

Overflow can only occur when the signs of two operands are same. A circuit to detect overflow and added to the n-bit adder by implementing the logic expression is as follows

$$= X_{n-1}Y_{n-1}\bar{S}_{n-1} + \bar{X}_{n-1}\bar{Y}_{n-1}S_{n-1}$$

**S.37 (d)**

From given figure we have,

$$S_i = x_i \oplus y_i \oplus C_i$$

$$C_{i+1} = x_i y_i + x_i C_i + y_i C_i$$

then factoring this we get,

$$C_{i+1} = x_i y_i + (x_i + y_i) C_i$$

$$\therefore C_{i+1} = G_i + P_i C_i$$

where  $G_i = x_i y_i$  and  $P_i = x_i + y_i$

the expressions  $G_i$  and  $P_i$  are called the generate and propagate functions for stage i and therefore (a) (b) and (c) are true statements about this.

**S.38 (a)**

In 2's complement representation, if two numbers with the same sign are added then overflow occurs if any only if the result has opposite sign.

For example:

$$0101 = 5$$

$$0100 = 4$$

$$-----$$

$$1001 = \text{overflow}$$

$$(+5) + (+4)$$

**S.39 (c)**

In the excess-127 format when the exponent E' = 0 and mantissa fraction M = 0, then the value equals to exact 0, whereas when E' = 255 and M = 0, the value  $\infty$  is represented.

**S.41 (b, c)**

**SEEMMMMMMM**

Where S = sign - usually represented by a value

between 0 and 9, 0–4 positive and 5–9 negative. The EE is the exponent and is usually represented as excess-n format i.e. if we are using excess-500, then we must minus 500 from the exponent to get its stored form.

**MMMMMM is the mantissa.**

e.g. in excess-50 the following value would be stored as follows:

$0.123455 \times 10^{-3}$

would be stored as 05312345.

#### S.42 (a)

Bit-slice processor can be cascaded to get any desired word length processor.

#### S.43 (a)

$$X = -10; \quad Y = 20;$$

if  $X < 0$  then  $X = \text{abs}(-10)$

$$\therefore X = 10$$

#### S.44 (c)

Consider the given formula:

$$\begin{aligned} & (3*4096 + 15*256 + 5*16 + 3) \\ &= (2+1)*2^{12} + (8+4+2+1)*2^8 + (4+1)*2^4 + 2^1 + 2^0 \\ &= (2^1 + 2^0)2^{12} + (2^3 + 2^2 + 2^1 + 2^0)2^8 + (2^2 + 2^0) \\ &\quad * 2^4 + 2^1 + 2^0 \\ &= 2^{13} + 2^{12} + 2^{11} + 2^{10} + 2^9 + 2^8 + 2^6 + 2^4 + 2^1 + 2^0 \\ &\quad \downarrow \quad \downarrow \\ &= (1 \ 1 \ 1 \ 1 \ 1 \ 01 \ 01 \ 001 \ 1)_2 \end{aligned}$$

has 10 number of 1's.

#### S.45 (a)

By applying Booth's algorithm for integer multiplication we can say that, multiplier pattern is **01111...1110** which gives worst performance.

#### S.46 (d)

|          |       |          |
|----------|-------|----------|
| 31       | 24 23 | 0        |
| Exponent |       | Mantissa |

Given that, the exponent is in 2's complement and mantissa is in the sign magnitude form. The range of the magnitude of the normalized numbers in this representation is **0.5 to  $(1-2^{-23})$** .

#### S.47 (c)

$$\text{Given } \sqrt{(224)_r} = (13)_r$$

$$(224)^{1/2} = (13)_r$$

$$\sqrt{(2r^2 + 2r^1 + 4r^0)} = (r+3)$$

$$(2r^2 + 2r^1 + 4) = (r+3)^2$$

$$(2r^2 + 2r + 4) = (r+3)^2$$

$$2r^2 + 2r + 4 = r^2 + 6r + 9$$

$$r^2 - 4r - 5 = 0$$

$$(r-5)(r+1) = 0$$

$$r \neq -1$$

$$\therefore r = 5$$

#### S.48 (a)

| D <sub>7</sub> | D <sub>6</sub> | D <sub>5</sub> | D <sub>4</sub> | D <sub>3</sub> | D <sub>2</sub> | D <sub>1</sub> | D <sub>0</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 1              | 1              | 1              | 0              | 0              | 0              | 1              | 0              |

Here D<sub>7</sub> is sign bit.

#### S.49 (b)

0 – 100 + 100 – 1 evaluates to 57 in Booth's coding as follows.

$$\begin{aligned} & (0 * 2^7) - (1 * 2^6) + (0 * 2^5) + (0 * 2^4) \\ &+ (1 * 2^3) + (0 * 2^2) + (0 * 2^1) - (1 * 2^0) \\ &= -64 + 8 - 1 = -57. \end{aligned}$$

#### S.50 (a)

Zero has two representation in **sign magnitude**.

#### S.51 (b)

The number 43 in 2's complement representation is **11010101**.

#### S.52 (b)

The decimal value 0.25 is equivalent to the binary value **0.01**.

#### S.53 (d)

2's complement = 1's complement + 1 (-15) = **10001**.

#### S.54 (d)

Sign extension is a step in converting a signed integer from one size to another.

#### S.55 (a)

Total size of micro instruction = 26 bits

Size of micro-operation = 13 bits

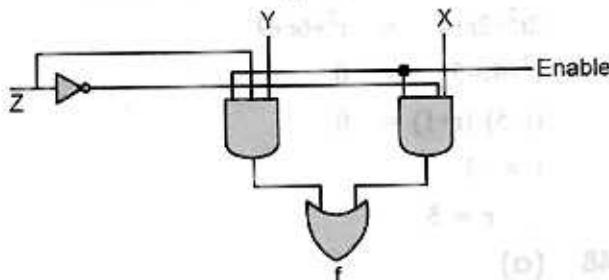
**3.1.12**

Mux input = 8 bits

So select line field size  $Y = 8 = 2^3 = 3$  bitsNext address field size  $X = 13 - 3 = 10$  bitsSize of control memory =  $2^{10} = 1024$ .**S.56 (b)**

$$f(a, b, c) = a'c + ac' + b'c$$

Consider the multiplexer as follows

If  $Z = 0$  then  $f = X$ if  $Z = 1$  then  $f = Y$ 

The output of multiplex is realized by f

$$f = ZY + Z'X$$

if  $X = R, Y = 1, Z = T$  then  $Z' = T'$ 

$$f = T.R + T'R$$

$$f = T + T'R$$

$$f = T + R$$

$$[x + x'y = x + y]$$

**S.57 (b)**

$$(i) \frac{1100}{10000}$$

$$(ii) \frac{0011}{1010}$$

$$(iii) \frac{1111}{0110}$$

In case of (i), no sign change takes place so no overflow.

In case of (ii), sign changes from (+ve) to (-ve) hence overflows.

In case of (iii), one member is positive and other is negative, hence there is no chance of overflow.

**S.58 (a)**

Binary equivalent of

$$(123456)_8 = (001\ 010\ 011\ 100\ 101\ 110)_2$$

converting it into hexadecimal group, we have

$$= (0000\ 1010\ 0111\ 0010\ 1110)_2$$

$$= (\text{A72E})_{16}$$

Similarly for converting it into base 4 group to 2 bit.

$$= (22130232)_4$$

**S.59 (a)**

$$(34.4)_8 \times (23.4)_8$$

multiply numbers on the base of 8

Number      Carry

$$8 \rightarrow 0$$

$$9 \rightarrow 1$$

$$10 \rightarrow 2$$

$$11 \rightarrow 3$$

$$12 \rightarrow 4$$

Similarly for other digits.

So

$$\begin{array}{r} 34.4 \\ \times 23.4 \\ \hline 1620 \\ 1254 \times \\ 710 \times \times \\ \hline 1053.60 \end{array}$$

$$\text{So } (34.4)_8 \times (23.4)_8 = (1053.60)_8$$

**S.60 (a)**if  $x = 1, y = 0, z = B$ 

from the given circuit, solving equation we get

$$Z = A\bar{B} + B$$

$$C = 1(\overline{A\bar{B}+B}) + 0(A\bar{B}+B)$$

$$= \overline{A\bar{B}+B} = (\overline{A} + B)\bar{B}$$

$$= \overline{A}\bar{B}$$

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





## CPU ORGANIZATION

### LEVEL-1

**3.2**

**Q.1** Which of the following statement is incorrect about immediate addressing mode ?

- (i) The operand data is directly specified in the operand field.
  - (ii) The instruction is a multiword instruction, where the operand immediately follow the opcode.
  - (iii) The opcode and the operand are fetched from memory using program counter.
- (a) (i), (iii)  
 (b) (i), (ii), (iii)  
 (c) (ii)  
 (d) None of these

**Q.2** In \_\_\_, the instruction contains an address that points to the memory location where the effective direct address to be used for operand is stored,

- (a) Extended Addressing mode
- (b) Indirect Addressing mode
- (c) Direct Address memory
- (d) Base Addressing mode

to different instruction word than calculated A  
 instruction word will be stored in memory  
 so less memory access required instead each (i)  
 memory location

and no memory access required A  
 location less time required to

execute the program A  
 less time required to

so less time required to execute the program A  
 less time required to

or machine instructions required A  
 memory location

with respect to you will no need to do A (i)  
 permit longer-distance to access

**Q.3** Which of the following statement is incorrect about program counter (PC)?

- (i) PC does not affect instruction sequencing
- (ii) During branch operation, program counter points to the memory from which instruction is to be fetched and executed.
- (a) (i)  
 (b) (ii)  
 (c) both (i) and (ii)  
 (d) Neither (i) nor (ii)

**Q.4** Which of the following statement is correct about the program counter?

- (i) Program counter points to the instruction to be executed.
- (ii) The program counter contents are used as address and instruction is read from memory.
- (a) (i)  
 (b) (ii)  
 (c) (i) and (ii)  
 (d) Neither (i) nor (ii)

**3.2.2****COMPUTER ORGANIZATION AND ARCHITECTURE**

- Q.5** A computer must have instructions capable of performing which of the following operations?
- (i) Data transfer between the memory and the processor register
  - (ii) Arithmetic and logic operations on data
  - (iii) Program sequencing and control
  - (iv) I/O transfers
- (a) (i), (ii)
  - (b) None of these
  - (c) (i), (ii), (iii), (iv)
  - (d) (ii), (iv)
- Q.6** Which of the following is valid design method for hardwired controllers?
- (i) The standard algorithmic approach to sequential circuit
  - (ii) A method based on the use of clocked delay elements for control-signal timing.
  - (iii) A related method that uses counters, for timing purposes.
- (a) (i), (ii)
  - (b) (i), (ii), (iii)
  - (c) (iii), (i)
  - (d) (ii), (iii)
- Q.7** Which of the following statement is incorrect?
- (i) A horizontal micro instruction format allows no encoding of control information, whereas a vertical format does.
  - (ii) A vertical micro instruction can specify only one micro operation, while a horizontal micro instruction can specify more than one micro operations.
- (a) (i)
  - (b) (ii)
  - (c) (i) and (ii)
  - (d) Neither (i) nor (ii)
- Q.8** Consider an  $(n+k)$  bit instruction with a  $k$ -bit opcode and a single  $n$ -bit address. Then, this instruction allows \_\_\_\_\_ operations and \_\_\_\_\_ addressable memory cells.
- (a)  $2^k, 2^{n+k}$
  - (b)  $2^{n+k}, 2^n$
  - (c)  $2^k, 2^n$
  - (d)  $2^{n-k}, 2^n$
- Q.9** The following diagram shows, which addressing mode?
- The diagram illustrates an instruction word structure. It consists of three fields: 'Opcode' (shaded grey), 'Lower address' (white), and 'Higher address' (white). An arrow originates from the 'Higher address' field and points to a 'Memory' block. Inside the 'Memory' block, there is a wavy line representing data and the word 'Operand' below it.
- (a) Immediate Addressing mode
  - (b) Indirect Addressing mode
  - (c) Extended Addressing mode
  - (d) Register Addressing mode
- Q.10** Addressing modes are
- (a) Explicitly specified
  - (b) Implied by the instruction
  - (c) Both (a) and (b)
  - (d) Neither (a) nor (b)
- Q.11** Control circuit designed by \_\_\_\_\_ tend to have a random structure
- (a) a state table method
  - (b) delay element method
  - (c) sequence counter method
  - (d) None of these
- Q.12** Which of following processor registers are used for fetch and execute operations?
- (i) Program counter
  - (ii) Instruction register
  - (iii) Address register
- (a) (i), (iii)
  - (b) (ii), (iii)
  - (c) (i), (ii)
  - (d) None of these
- Q.13** The immediate addressing modes can be used for \_\_\_\_\_
- (i) Loading internal registers with initial value
  - (ii) Perform arithmetic or logical operation on immediate data
- (a) (i)
  - (b) (ii)
  - (c) both (i) and (ii)
  - (d) Neither (i) nor (ii)

- Q.14** Microinstruction length is determined by \_\_\_\_\_  
 (i) The maximum number of simultaneous micro operations that must be specified  
 (ii) The way in which the control information is represented or encoded  
 (iii) The way in which the next micro instruction address is specified  
 (a) (i), (ii)  
 (b) (ii), (iii)  
 (c) (i), (iii)  
 (d) All
- Q.15** The control unit of computer  
 (a) performs ALU operations on the data  
 (b) controls the operation of the output devices  
 (b) is a device for manually operating the computer  
 (d) directs the other units of computer
- Q.16** Which of the following is the internal memory of the system (computer)?  
 (a) CPU register  
 (b) Cache  
 (c) Main memory  
 (d) All of these
- Q.17** The register which contains the instruction to be executed is called  
 (a) instruction register  
 (b) memory address register (MAR)  
 (c) index register  
 (d) memory data register (MDR)
- Q.18** The three buses associated with the three bus system are I/O bus, memory bus and  
 (a) address bus  
 (b) unibus  
 (c) direct memory access bus (DMA)  
 (d) data bus
- Q.19** The bus which is used to transfer data from main memory to peripheral devices is  
 (a) data bus  
 (b) input bus  
 (c) DMA bus  
 (d) output bus
- Q.20** Which of the following is used as storage locations both in the ALU and the control section of a computer?  
 (a) Register  
 (b) Decoder  
 (c) Adder  
 (d) Accumulator
- Q.21** Which of the following addressing modes permit relocation without any change what so ever is the code?  
 (a) Base register addressing  
 (b) PC relative addressing  
 (c) Indexed addressing  
 (d) Indirect addressing
- Q.22** The most relevant addressing mode to write position independent code is  
 (a) relative mode  
 (b) indirect mode  
 (c) indexed mode  
 (d) direct mode
- Q.23** A micro-processor is \_\_\_\_\_  
 (a) solid state device  
 (b) capable of performing arithmetic operations  
 (c) capable of performing logical operations  
 (d) all of the above
- Q.24** What is control unit's function in the CPU?  
 (a) to decode program instruction  
 (b) to perform logic operation  
 (c) to store program instructions  
 (d) none of the above
- Q.25** A computer that is capable of parallel processing is \_\_\_\_\_ computer.  
 (a) Micro  
 (b) Main frame  
 (c) Host  
 (d) Parallel
- Q.26** Addressing mode used in instruction L × 1 B 0345 H  
 (a) Indirect  
 (b) Direct  
 (c) Extended  
 (d) Immediate

- Q.27** The group of operations that a microprocessor can perform is called its  
 (a) Memory  
 (b) Opcode  
 (c) Instruction set  
 (d) None of the above
- Q.28** Types of instructions found in a microprocessor's instruction set.  
 (a) Data transfer  
 (b) Arithmetic  
 (c) Store  
 (d) None of the above
- Q.29** The section of the CPU that interprets the opcode placed in the instruction register and directs the control and timing section how to execute the instruction is called the  
 (a) instruction decoder  
 (b) instruction coder  
 (c) accumulator  
 (d) none of the above
- Q.30** Name functional units contained on most microprocessor chips  
 (a) Memory  
 (b) ALU  
 (c) Program counter  
 (d) None of the above
- Q.31** Horizontal microprogramming  
 (a) does not require use of signal decoders  
 (b) results in larger sized microinstructions than vertical microprogramming  
 (c) uses one bit for each control signal  
 (d) all of the above
- Q.32** The minimum time delay between the initiations of two independent memory operations is called  
 (a) access time  
 (b) cycle time  
 (c) transfer rate  
 (d) latency time
- Q.33** Which of the following comments about the Program Counter (PC) are true?  
 (a) It is a register  
 (b) It is a cell in ROM  
 (c) During execution of the current instruction, its content changes.  
 (d) None of the above
- Q.34** Which of the following are registers?  
 (a) Accumulator  
 (b) Stack pointer  
 (c) Program counter  
 (d) Buffer
- Q.35** Choose the correct statements.  
 (a) Bus is a group of information carrying wires.  
 (b) Bus is needed to achieve reasonable speed of operation.  
 (c) Bus can carry data or address.  
 (d) A bus can be shared by more than one device.  
 (e) All
- Q.36** The sequence of events that happen during a typical fetch operation is  
 (a) PC → MAR → Memory → MDR → IR  
 (b) PC → Memory → MDR → IR  
 (c) PC → Memory → IR  
 (d) PC → MAR → Memory → IR
- Q.37** A byte addressable computer has a memory capacity of  $2(m + 10)$  bytes and can perform  $2^n$  operations. If the computer is word addressable with the word size being 8 bytes then the answer will be  
 (a)  $3m$  bits  
 (b)  $3m + n$  bits  
 (c)  $m + n$  bits  
 (d) None of the above
- Q.38** A computer uses ternary system instead of the traditional binary system. An ' $n$ ' bit string in the binary system will occupy  
 (a)  $3 + n$  ternary digits  
 (b)  $2n/3$  ternary digits  
 (c)  $n(\log_2 3)$  ternary digits  
 (d)  $n(\log_3 2)$  ternary digits

**Q.39** The addressing mode used in an instruction of the form ADD X Y, is

- (a) absolute
- (b) Immediate
- (c) Indirect
- (d) Index

**Q.40** The addressing mode used in the instruction PUSH B is

- (a) direct
- (b) register
- (c) register indirect
- (d) immediate

**Q.41** Assume a system having two CPU, each executing different programs. At the same time one of the CPU's tries to increment a variable x, while the other CPU tries to decrement the same x. The final value of x can never be

- (a) its original value
- (b) its original value + 1
- (c) its original value - 1
- (d) None of the above

**Q.42** An assembler that runs on one machine but produces machine code for another machine is called

- (a) simulator
- (b) emulator
- (c) cross-assembler
- (d) boot-strap loader

**Q.43** The most relevant addressing mode to write position independent code is

- (a) direct mode
- (b) indirect mode
- (c) relative mode
- (d) indexed mode

**Q.44** A computer has 32-bit instructions and 12-bit addresses. If there are 250 two-address instructions, how many one-address instructions can be formulated?

- (a)  $6 \times 2^{32}$
- (b)  $6 \times 2^{12}$
- (c)  $32 \times 2^6$
- (d)  $32 \times 2^{12}$

## LEVEL-2

**Q.45** While designing a control unit using hardwired control, a module-k sequence counter can behave like a cascade of \_\_\_\_\_ delay elements.

- (a) k
- (b)  $k-1$
- (c)  $k+1$
- (d)  $k^3$

**Q.46** Which of the following statement is correct about the advantage of register indirect addressing?

- (i) This mode is used to save program space
- (ii) This mode improve speed of program execution in some situations
- (a) (i)
- (b) (ii)
- (c) (i) and (ii)
- (d) Neither (i) nor (ii)

**Q.47** Which of the following statement is correct ?

- (i) In micro programmed control, control memories are usually ROMs.
- (ii) A Processor with a writable control memory is said dynamically micro programmable.
- (a) (i)
- (b) (ii)
- (c) (i) and (ii)
- (d) Neither (i) nor (ii)

**Q.48** Which of the following is valid micro instruction types?

- (a) Branch micro instruction
- (b) Operate micro instruction
- (c) Unbranch micro instruction
- (d) Both (a) and (b)

**Q.49** Match the following

| Group I |           | Group II |                           |
|---------|-----------|----------|---------------------------|
| 1.      | MOV X, R1 | (i)      | Three-address instruction |
| 2.      | STORE X   | (ii)     | Zero-address instruction  |
| 3.      | POP X     | (iii)    | One-address instruction   |
|         |           | (iv)     | Two-address instruction   |

- (a) (1)-(iv), (2)-(iii), (3)-(ii)
- (b) (1)-(iii), (2)-(ii), (3)-(i)
- (c) (1)-(ii), (2)-(iii), (3)-(iv)
- (d) None of these

**Q.50** Which of the following is not valid class of Interrupts?

- (i) Program
  - (ii) Timer
  - (iii) I/O
  - (iv) Hardware failure
- (a) (i),(iii)
  - (b) (i), (ii), (iv)
  - (c) (ii), (iii)
  - (d) None of these

**Q.51** A memory system of size 16 K bytes is required to be designed using memory chips which have 12 address lines and 4 data lines each. The number of such chips required to design the memory system is

- (a) 2
- (b) 4
- (c) 8
- (d) 16

**Q.52** How long will it take for a single instruction to execute in the pipelined implementation?

- (a) 300 ns
- (b) 360 ns
- (c) 400 ns
- (d) 380 ns

### Common Data For Questions 53 to 56:

An address space is specified by 16-bits and corresponding memory space by 12-bits.

**Q.53** No. of words in address space

- (a) 64 K
- (b) 48 K
- (c) 32 K
- (d) 16 K

**Q.54** No. of words in memory space

- (a) 8 K
- (b) 6 K
- (c) 4 K
- (d) 2 K

**Q.55** No. of pages in the system if a page has 256 words

- (a) 256 pages
- (b) 200 pages
- (c) 160 pages
- (d) 126 pages

**Q.56** No. of blocks in the system if a page has 256 words

- (a) 16 blocks
- (b) 12 blocks
- (c) 8 blocks
- (d) 4 blocks

**Q.57** Let  $*$  be defined as  $a * b = a' + b$ . Let  $m = a * b$ . The value of  $m * a$  is

- (a)  $a' + b$
- (b)  $a$
- (c) 0
- (d) 1

### Common Data For Questions 58 to 60:

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. The effective address will be if the addressing mode is:

**Q.58** Immediate:

- (a) 300
- (b) 702
- (c) 301
- (d) 200

**Q.59** Relative:

- (a) 702
- (b) 400
- (c) 600
- (d) 300

**Q.60** Index with RI as the index register:

- (a) 300
- (b) 700
- (c) 600
- (d) 301

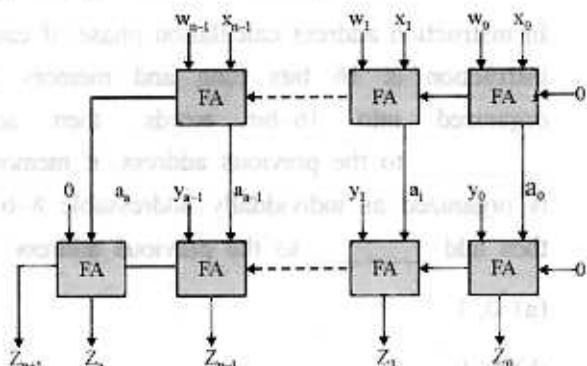
## LEVEL-3

**Q.61** Match the following:

| Group I |                       | Group II |                 |
|---------|-----------------------|----------|-----------------|
| (A)     | 0-address instruction | 1.       | T = Top (T-1)   |
| (B)     | 1-address instruction | 2.       | $Y = Y + X$     |
| (C)     | 2-address instruction | 3.       | $Y = A - B$     |
| (D)     | 3-address instruction | 4.       | $ACC = ACC - X$ |

- (a) A-1, B-2, C-3, D-4
- (b) A-3, B-2, C-4, D-1
- (c) A-2, B-3, C-1, D-4
- (d) A-1, B-4, C-2, D-3

**Q.62** Consider the following circuit:



The above circuit represents

- (a)  $W+X+Y$  by using single ripple carry adder
- (b)  $W+X+Y$  by using two ripple-carry adders
- (c)  $W+X+Y$  using  $n$  ripple carry adders
- (d)  $W+X+Y$  by using  $n$  carry save adders

**Q.63** The computer can execute 1,000,000 instructions per second. A program running on this computer

with 2200 instructions performs on average a one sector read and one sector write for every 200 instructions that it executes. The disk drive handling the I/O transfers requires 0.00010 seconds each to perform the read and write operations. Assuming no overlap of these operations, the percent of CPU time spent in the wait state is

- (a) 72%
- (b) 30%
- (c) 60%
- (d) 91%

**Q.64** The following are some sequences of operations in instruction cycle, which one is correct sequence?

(a) PC  $\rightarrow$  Address register

Data from memory  $\rightarrow$  Data register

Data register  $\rightarrow$  IR

PC + 1  $\rightarrow$  PC

(b) Address register  $\rightarrow$  PC

Data register  $\rightarrow$  Data from memory

Data register  $\rightarrow$  IR

PC + 1  $\rightarrow$  PC

(c) Data from memory  $\rightarrow$  Data register

PC  $\rightarrow$  Address register

Data register  $\rightarrow$  IR

PC + 1  $\rightarrow$  PC

(d) None of these.

**Linked Questions 65 & 66:**

**Q.65** The clock period of the pipelined processor is

$$(a) \text{Cycle Time}_{\text{pipelined}} =$$

$$\frac{\text{CycleTime}_{\text{unpipelined}}}{\text{Number of Pipeline Stage}} + \text{Pipeline latch latency}$$

$$(b) \text{CycleTime}_{\text{pipelined}} =$$

$$\frac{\text{CycleTime}_{\text{unpipelined}}}{\text{Number of Pipeline Stage}} - \text{Pipeline latch latency}$$

$$(c) \text{CycleTime}_{\text{pipelined}} =$$

$$\frac{\text{CycleTime}_{\text{unpipelined}}}{\text{pipeline latch latency} - \text{Number of Pipeline Stages}}$$

$$(d) \text{None of the above}$$

**Q.66** Using above result, calculate following an unpipelined processor has a cycle time of 25 ns, what is the cycle time of a pipelined version of the processor with 5 evenly divided pipeline stage, if each pipeline latch has latency of 1 ns?

- (a) 8ns
- (b) 6ns
- (c) 2ns
- (d) 4ns

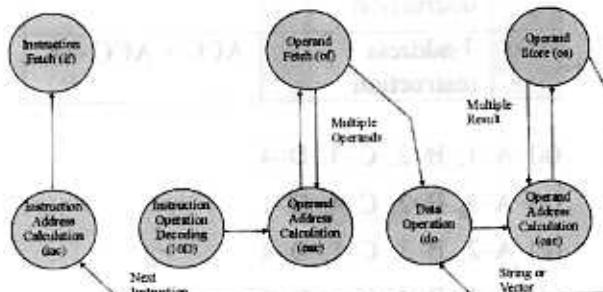
**Q.67** Assume that the times required for the five functional units, which operates in each of the five cycles, are as follows: 10ns, 9 ns, 9 ns, 7ns, 10 ns. Assume that pipelining is used and calculate the speed of pipelining.

- (a) 4.1
- (b) 4.09
- (c) 5.1
- (d) 4.07

**Q.68** Consider the unpipelined machine with 10 ns clock cycles. It uses four cycles for ALU operations and branches, whereas five cycles for memory operations. Assume that the relative frequencies of the operations are 40%, 20%, 40% respectively. Suppose that due to clock skew and setup, pipelining the machine adds 1 ns overhead to the clock. How much speed up in the instruction execution rate will we gain from a pipeline?

- (a) 7 times
- (b) 10 times
- (c) 4 times
- (d) 4.5 times

**Q.69** Consider the following instruction cycle state diagram



In instruction address calculation phase, if each instruction is 16 bits long and memory is organized into 16-bit words, then add \_\_\_\_\_ to the previous address, if memory is organized as individually addressable 8-bit, then add \_\_\_\_\_ to the previous address.

- (a) 0, 1
- (b) 1, 0
- (c) 1, 2
- (d) 2, 1

## GATE QUESTIONS

**Q.70** The most relevant addressing mode to write position-independent codes is [GATE 1987]

- (a) Direct mode
- (b) Indirect mode
- (c) Relative mode
- (d) Indexed mode

**Q.71** On receiving an interrupt from a Input-Output device the CPU [GATE 1987]

- (a) halts for a predetermined time
- (b) hands over control of address bus and data bus to the interrupting device
- (c) branches off to the interrupt service routine immediately
- (d) branches off to the interrupt service routine after completion of the current instruction

**Q.72** The capacity of a memory unit is defined by the number of words multiplied by the number of bits/word. How many separate address and data lines are needed for a memory of  $4K \times 16$ ? [GATE 1995]

[2-Marks]

- (a) 10 address, 16 data lines
- (b) 11 address, 8 data lines
- (c) 12 address, 16 data lines
- (d) 12 address, 12 data lines

**Q.73** A microprogram control unit is required to generate a total of 25 control signals. Assume that during any microinstruction, at most two control signals are active. Minimum number of bits required in the control word to generate the required control signals will be [GATE 1996]

- (a) 2
- (b) 2.5
- (c) 10
- (d) 12

**Q.74** Relative mode of addressing is most relevant to writing [GATE 1996]

- (a) co-routines
- (b) position-independent code
- (c) shareable code
- (d) interrupt handlers

**Q.75** A certain processor supports only the immediate and the direct addressing modes. Which of the following programming language features cannot be implemented on this processor?

[GATE 1996]

- (a) Pointers
- (b) Arrays
- (c) Records
- (d) Recursive procedures with local variable

**Q.76** The correct matching for the following pairs is [GATE 1997]

[1-Mark]

|     | <b>Group I</b>          |    | <b>Group II</b> |
|-----|-------------------------|----|-----------------|
| (a) | DMA I/O                 | 1. | High Speed RAM  |
| (b) | Cache                   | 2. | Disk            |
| (c) | Interrupt I/O           | 3. | Printer         |
| (d) | Condition Code Register | 4. | ALU             |

- (a) a - 4, b - 3, c - 1, d - 2
- (b) a - 2, b - 1, c - 3, d - 4
- (c) a - 4, b - 3, c - 2, d - 1
- (d) a - 2, b - 3, c - 4, d - 1

**Q.77** A micro instruction is to be designed to specify

- (i) none or one of the three micro-operations of one kind and
- (ii) none or upto six micro-operations of another kind

The minimum number of bits in the micro-instruction is [GATE 1997]

[2-Marks]

- (a) 9
- (b) 5
- (c) 8
- (d) none of the above

**Q.78** Which of the following devices should get higher priority in assigning interrupt? [GATE 1998]

[1-Mark]

- (a) Hard disk
- (b) Printer
- (c) Keyboard
- (d) Floppy disk

**Q.79** Arrange the following configurations for CPU in decreasing order of operating speeds:

Hard wired control, vertical micro-programming, horizontal micro-programming [GATE 1999]

[2-Marks]

- (a) Hard wired control, vertical micro-programming, horizontal micro-programming.
- (b) Hard wired control, horizontal micro-programming, vertical micro-programming
- (c) Horizontal micro-programming, vertical micro-programming, hard wired control.
- (d) Vertical micro-programming, horizontal micro-programming, hard wired control

**Q.80** The most appropriate matching for the following pairs

| Group I |                           | Group II |           |
|---------|---------------------------|----------|-----------|
| (X)     | Indirect addressing       | I        | Loops     |
| (Y)     | Immediate addressing      | II       | Pointers  |
| (Z)     | Auto decrement addressing | III      | Constants |

[GATE 2000]

- (a) X - 3, Y - 2, Z - 1
- (b) X - 1, Y - 3, Z - 2
- (c) X - 2, Y - 3, Z - 1
- (d) X - 3, Y - 1, Z - 2

**Q.81** Comparing the time  $T_1$  taken for a single instruction on a pipelined CPU with time  $T_2$  taken on a non-pipelined but identical CPU, we can say that [GATE 2000]

[1-Mark]

- (a)  $T_1 \leq T_2$
- (b)  $T_1 \geq T_2$
- (c)  $T_1 < T_2$
- (d)  $T_1$  is  $T_2$  plus the time taken for one instruction fetch cycle

**Q.82** A processor needs software interrupt to

[GATE 2001]

[1-Mark]

- (a) test the interrupt system of the processor
- (b) implement co-routines
- (c) obtain system services which need execution of privileged instructions
- (d) return from subroutine

**Q.83** Which is the most appropriate match for the items in the first column with the items in the second column? [GATE 2001]

[2-Marks]

| Group I |                          | Group II |                            |
|---------|--------------------------|----------|----------------------------|
| X       | Indirect Addressing      | I        | Array implementation       |
| Y       | Indexed Addressing       | II       | Writing relocatable code   |
| Z       | Base Register Addressing | III      | Passing array as parameter |

- (a) (X, III), (Y, I), (Z, II)
- (b) (X, II), (Y, III), (Z, II)
- (c) (X, III), (Y, II), (Z, I)
- (d) (X, I), (Y, III), (Z, II)

**Q.84** The performance of a pipelined processor suffers if [GATE 2002]

[2-Marks]

- (a) the pipeline stages have different delays
- (b) consecutive instruction are dependent on each other
- (c) the pipeline stages share hardware resources
- (d) all of the above

**Q.85** For a pipelined CPU with a single ALU, consider the following situations

1. The  $j+1^{\text{st}}$  instruction uses the result of the  $j^{\text{th}}$  instruction as an operand
2. The execution of a conditional jump instruction
3. The  $j^{\text{th}}$  and  $j+1^{\text{st}}$  instructions require the ALU at the same time

Which of the above can cause a hazard?

[GATE 2003]

[1-Mark]

- (a) 1 and 2 only
- (b) 2 and 3 only
- (c) 3 only
- (d) All the three

**Q.86** Consider an array multiplier for multiplying two  $n$  bit numbers. If each gate in the circuit has a unit delay, the total delay of the multiplier is

[GATE 2003]

[1-Mark]

- (a)  $\Theta(1)$
- (b)  $\Theta(\log n)$
- (c)  $\Theta(n^2)$
- (d)  $\Theta(n)$

#### Common Data For Questions 87 & 88:

Consider the following assembly language program for a hypothetical processor. A, B and C are 8 bit registers. The meanings of various instructions are shown as comments.

MOV B,#0 ;  $B \leftarrow 0$

MOV C,#8 ;  $C \leftarrow 8$

Z : CMP C,#0; compare C with 0

JZ X ; jump to X if zero flag is set

SUB C,#1;  $C \leftarrow C - 1$

RRC A,#1; right rotate A through carry by one bit. Thus:

; If the initial values of A and the carry flag are  $a_7 \dots a_0$  and

;  $c_0$  respectively, their values after the execution of this

; instruction will be  $c_0a_7 \dots a_1$  and  $a_0$  respectively.

JC Y ; jump to Y if carry flag is set

JMP Z ; jump to Z

Y : ADD B,#1;  $B \leftarrow B + 1$

JMP Z ; jump to Z

X :

**Q.87** If the initial value of register A is  $A_0$ , the value of register B after the program execution will be

[GATE 2003]

[2-Marks]

- (a) the number of 1 bits in  $A_0$
- (b) the number of 0 bits in  $A_0$
- (c)  $A_0$
- (d) 8

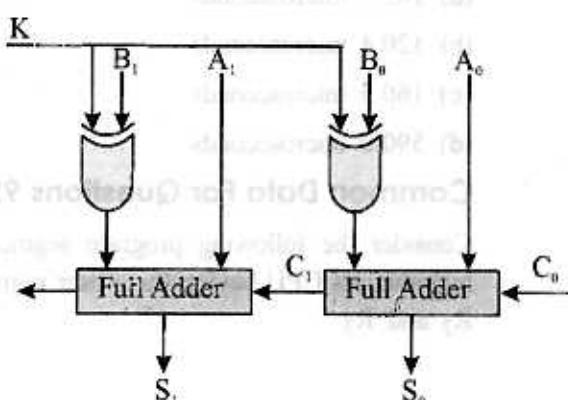
**Q.88** Which of the following instruction when inserted at location X will ensure that the value of register A after program execution is the same as its initial value?

[GATE 2003]

[2-Marks]

- (a) NOP ; no operation
- (b) LRCA,#1 ; left rotate A through carry flag by one bit
- (c) ADD A,#1
- (d) RRC A,#1

**Q.89** Consider the ALU shown below



If the operands are in 2's complement representation, which of the following operation can be performed by suitably setting the control lines K and  $C_0$  only (+ and - denote addition and subtraction respectively)?

[GATE 2003]

[2-Marks]

- (a)  $A + B$ , and  $A - B$ , and  $A + 1$

- (b)  $A + B$ , and  $A - B$ , but not  $A + 1$

- (c)  $A + B$ , and  $A + 1$ , but not  $A - B$

- (d)  $A + B$ , but not  $A - B$ , or  $A + 1$

**Q.90** Which of the following addressing modes are suitable for program relocation at run time?

1. Absolute addressing
2. Based addressing
3. Relative addressing
4. Indirect addressing

[GATE 2004]

[1-Mark]

- (a) 1 and 4
- (b) 1 and 2
- (c) 1, 2 and 4
- (d) 2 and 3

**Q.91** A 4-stage pipeline has the stage delays as 150, 120, 160 and 140 nanoseconds respectively. Registers that are used between the stages have a delay of 5 nanoseconds each. Assuming constant clocking rate, the total time taken to process 1000 data items on this pipeline will be

[GATE 2004]

[2-Marks]

- (a) 165.5 microseconds
- (b) 120.4 microseconds
- (c) 160.5 microseconds
- (d) 590.0 microseconds

#### Common Data For Questions 92 & 93:

Consider the following program segment for a hypothetical CPU having three user registers R<sub>1</sub>, R<sub>2</sub> and R<sub>3</sub>.

| Instruction     | Operation            | Instruction Size(in words) |
|-----------------|----------------------|----------------------------|
| MOV R1,<br>5000 | ; R1 ← Memory [5000] | 2                          |
| MOV<br>R2,(R1)  | ; R2 ← Memory [(R1)] | 1                          |
| ADD R2,<br>R3   | ; R2 ← R2 + R3       | 1                          |
| MOV 6000,<br>R2 | ; Memory [6000] ← R2 | 2                          |
| HALT            | ; Machine halts      | 1                          |

**Q.92** Consider that the memory is byte addressable with size 32 bits, and the program has been loaded starting from memory location 1000 (decimal). If an interrupt occurs while the CPU has been halted after executing the HALT instruction, the return address (in decimal) saved in the stack will be

[GATE 2004]

[2-Marks]

- (a) 1007
- (b) 1024
- (c) 1020
- (d) 1028

**Q.93** Let the clock cycles required for various operations be as follows:

| Operations                         | Req. Clock Cycles       |
|------------------------------------|-------------------------|
| Register to/form memory transfer   | 3 clock cycles          |
| ADD with both operands in register | 1 clock cycles          |
| Instructions fetch and decode      | 2 clock cycles per word |

The total number of clock cycles required to execute the program is

[GATE 2004]

[2-Marks]

- (a) 29
- (b) 23
- (c) 24
- (d) 20

**Q.94** Consider a system with 2 level caches. Access times of Level 1 cache, Level 2 cache and main memory are 1 ns, 10 ns, and 500 ns, respectively. The hit rates of Level 1 and Level 2 caches are 0.8 and 0.9, respectively. What is the average access time of the system ignoring the search time within the cache?

[IT-GATE 2004]

[1 Mark]

- (a) 13.0 ns
- (b) 12.8 ns
- (c) 12.6 ns
- (d) 12.4 ns

**Q.95** If we use internal data forwarding to speed up the performance of a CPU (R1, R2 and R3 are registers and M[100] is a memory reference), then the sequence of operations

$$R1 \rightarrow M[100]$$

$$M[100] \rightarrow R2$$

$$M[100] \rightarrow R3$$

Can be replaced by

[IT-GATE 2004]

[2-Marks]

(a)  $R1 \oplus R3$

$$R2 \rightarrow M[100]$$

(b)  $M[100] \rightarrow R2$

$$R1 \rightarrow R2$$

$$R1 \rightarrow R3$$

(c)  $R1 \rightarrow M[100]$

$$R2 \rightarrow R3$$

(d)  $R1 \rightarrow R2$

$$R1 \rightarrow R3$$

(e)  $R1 \rightarrow M[100]$

**Q.96** Consider a fully associative cache with 8 cache blocks (numbered 0–7) and the following sequence of memory block requests:

4, 3, 25, 8, 19, 6, 25, 8, 16, 35, 45, 22, 8, 3, 16, 25, 7

If LRU replacement policy is used, which cache block will have memory block 7?

[IT-GATE 2004]

[2-Marks]

(a) 4

(b) 5

(c) 6

(d) 7

**Q.97** In an enhancement of a design of a CPU, the speed of a floating point unit has been increased by 20% and the speed of a fixed point unit has been increased by 10%. What is the overall speedup achieved if the ratio of the number of floating point operations to the number of fixed point operations is 2:3 and the floating point operation used to take twice the time taken by the fixed point operation in the original design?

[IT-GATE 2004]

[2-Marks]

(a) 1.155

(b) 1.185

(c) 1.255

(d) 1.285

**Q.98** A hardwired CPU uses 10 control signals S1 to S10 in various time steps T1 to T5 to implement 4 instructions I1 to I4 as shown below.

|    | T1       | T2        | T3       | T4    | T5    |
|----|----------|-----------|----------|-------|-------|
| I1 | S1,S3,S5 | S2,S4,S6  | S1,S7    | S10   | S3,S8 |
| I2 | S1,S3,S5 | S8,S9,S10 | S5,S6,S7 | S6    | S10   |
| I3 | S1,S3,S5 | S7,S8,S10 | S2,S6,S9 | S10   | S1,S3 |
| I4 | S1,S3,S5 | S2,S6,S7  | S5,S10   | S6,S9 | S10   |

Which of the following pairs of expressions represent the circuit for generating control signals S5 and S10 respectively [(Ij + Ik) Tn indicates that the control signal should be generated in time step Tn if the instruction being executed is Ij or Ik].

[IT-GATE 2005]

[2-Marks]

(a)  $S5 = T1 + I2 \cdot T3 \& S10 = (I1 + I3) \cdot T4 + (I2 + I4) \cdot T5$

(b)  $S5 = T1 + (I2 + I4) \cdot T3 \& S10 = (I1 + I3) \cdot T4 + (I2 + I4) \cdot T5$

(c)  $S5 = T1 + (I2 + I4) \cdot T3 \& S10 = (I2 + I3 + I4) \cdot T2 + (I1 + I3) \cdot T4 + (I2 + I4) \cdot T5$

(d)  $S5 = T1 + (I2 + I4) \cdot T3 \& S10 = (I2 + I3) \cdot T2 + I4 \cdot T3 + (I1 + I3) \cdot T4 + (I2 + I4) \cdot T5$

**Q.99** Match List-I and List II and select the correct answer using the codes given below the lists:

|     | List I         |    | List II             |
|-----|----------------|----|---------------------|
| (A) | $A[I] = B[J];$ | 1. | Indirect addressing |
| (B) | While(*A++);   | 2. | Indexed addressing  |
| (C) | int temp = *X; | 3. | Autoincrement       |

[GATE 2005]

[2-Marks]

**Codes:**

|     | A | B | C |
|-----|---|---|---|
| (a) | 3 | 2 | 1 |
| (b) | 2 | 3 | 1 |
| (c) | 1 | 3 | 2 |
| (d) | 1 | 2 | 3 |

**Q.100** Consider a three word machine instruction

ADD A[R0],@B

The first operand (destination) "A[R0]" uses indexed addressing mode with R0 as the index register. The second operand (source) "@B" uses indirect addressing mode. A and B are memory addresses residing at the second and the third words, respectively. The first word of the instruction specifies the opcode, the index register designation and the source and destination addressing modes. During execution of ADD instruction, the two operands are added and stored in the destination (first operand).

The number of memory cycles needed during the execution cycle of the instruction is

[GATE 2005]

[2-Marks]

- (a) 3
- (b) 4
- (c) 5
- (d) 6

**Q.101** A CPU has 24-bit instructions. A program starts at address 300 (in decimal). Which one of the following is a legal program counter (all values in decimal)?

[GATE 2005]

[1-Mark]

- (a) 600
- (b) 400
- (c) 500
- (d) 700

**Q.102** Consider a direct mapped cache of size 32KB with block size 32 bytes. The CPU generates 32 bit addresses. The number of bits needed for cache indexing and the number of tag bits are respectively

[GATE 2005]

[2-Marks]

- (a) 10, 22
- (b) 15, 17
- (c) 10, 17
- (d) 5, 17

**Q.103** A 5 stage pipelined CPU has the following sequence of stages

IF – Instruction fetch from instruction memory,

RD – Instruction decode and register read,

EX – Execute: ALU operation for data and address computation,

MA – Data memory access – for write access the register read at RD stage it used,

WB – Register write back.

Consider the following sequence of instructions:

$I_1 : L R0, loc1; R0 \leftarrow M[loc1]$

$I_2 : A R0, R0 1; R0 \leftarrow R0 + R0$

$I_3 : A R2, R0 1; R2 \leftarrow R2 - R0$

Let each stage take one clock cycle.

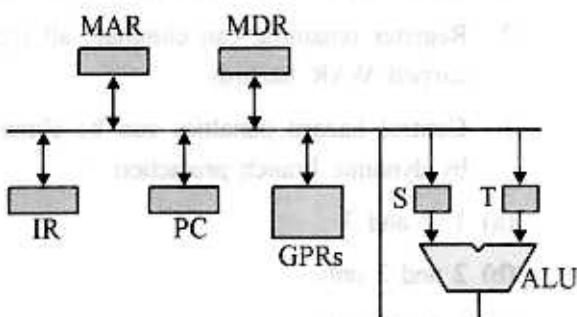
What is the number of clock cycles taken to complete the above sequence of instructions starting from the fetch of  $I_1$ ? [GATE 2005]

[2-Marks]

- (a) 8
- (b) 15
- (c) 12
- (d) 10

**Common Data For Questions 104 & 105:**

Consider the following data path of a CPU



The ALU, the bus and all the registers in the data path are of identical size. All operations including incrementation of the PC and the GPRs are to be carried out in the ALU. Two clock cycle are needed for memory read operation - the first one for loading address in the MAR and the next one for loading data from the memory but into the MDR.

**Q.104** The instruction "add R0,R1" has the register transfer interpretation  $R0 \leftarrow R0 + R1$ . The minimum number of clock cycles needed for execution cycle of this instruction is

[GATE 2005]

- (a) 5
- (b) 4
- (c) 3
- (d) 2

[2-Marks]

**Q.105** The instruction "call Rn, sub" is a two word instruction. Assuming that PC is incremented during the fetch cycle of the first word of the instruction, its register transfer interpretation is

$$Rn \leftarrow PC + 1;$$

$$PC \leftarrow M[PC];$$

The minimum number of CPU clock cycles needed during the execution cycle of this instruction is

[GATE 2005]

- (a) 3
- (b) 2
- (c) 4
- (d) 5

[2-Marks]

**Q.106** Consider a new instruction named branch-on-bit-set (mnemonic bbs). The instruction "bbs reg, pos, label" jumps to label if bit in position pos of register operand reg is one. A register is 32 bits wide and the bits are numbered 0 to 31, bit in position 0 being the least significant. Consider the following emulation of this instruction on a processor that does not have bbs implemented.  
 $\text{temp} \leftarrow \text{reg} \& \text{mask}$

Branch to label if temp is non-zero

The variable temp is a temporary register. For correct emulation, the variable mask must be generated by

[GATE 2006]

[2-Marks]

- (a)  $\text{mask} \leftarrow 0 \times \text{fffff}ff \gg \text{pos}$
- (b)  $\text{mask} \leftarrow \text{pos}$
- (c)  $\text{mask} \leftarrow 0 \times f$
- (d)  $\text{mask} \leftarrow 0 \times 1 \ll \text{pos}$

**Q.107** A CPU has a cache with block size 64 bytes. The main memory has k banks, each bank being c bytes wide. Consecutive c-byte chunks are mapped on consecutive banks with warp-around. All the k banks can be accessed in parallel, but two accesses to the same bank must be serialized. A cache block access may involve multiple iterations of parallel bank accesses depending on the amount of data obtained by accessing all the k banks in parallel. Each iteration requires decoding the bank numbers to be accessed in parallel and this takes  $k/2$  ns. The latency of one bank access is 80 ns. If  $c = 2$  and  $k = 24$ , then latency of retrieving a cache block starting at address zero from main memory is

[GATE 2006]

[2-Marks]

- (a) 184 ns
- (b) 92 ns
- (c) 104 ns
- (d) 172 ns

**Q.108** A CPU has five-stages pipeline and runs at 1GHz frequency. Instruction fetch happens in the first stage of the pipeline. A conditional branch instruction computes the target address and evaluates the condition in the third stage of the pipeline. The processor stops fetching new instructions following a conditional branch until the branch outcome is known. A program executes  $10^9$  instructions out of which 20% are conditional branches. If each instruction takes one cycle to complete on average, then total execution time of the program is [GATE 2006]

[2-Marks]

- (a) 1.6 seconds
- (b) 1.4 seconds
- (c) 1.2 seconds
- (d) 1.0 second

**Q.109** Consider a pipelined processor with the following four stages

IF : Instruction Fetch

ID : Instruction Decode and Operand Fetch

EX : Execute

WB : Write Back

The IF, ID and WB stages take one clock cycle each to complete the operation. The number of clock cycles for the EX stage depends on the instruction. The ADD and SUB instructions need 1 clock cycle and the MUL instruction need 3 clock cycles in the EX stage. Operand forwarding is used in the pipelined processor. What is the number of clock cycles taken to complete the following sequence of instructions?

[GATE 2007]

[2-Marks]

|            |     |     |    |                         |
|------------|-----|-----|----|-------------------------|
| <b>ADD</b> | R2, | R1, | R0 | $R2 \leftarrow R1 + R0$ |
| <b>MUL</b> | R4, | R3, | R2 | $R4 \leftarrow R3 * R2$ |
| <b>SUB</b> | R6, | R5, | R4 | $R6 \leftarrow R5 - R4$ |

- (a) 14
- (b) 10
- (c) 8
- (d) 7

**Q.110** Which of the following are NOT true in a pipelined processor? [GATE 2008]

[2-Marks]

- 1. Bypassing can handle all Raw hazards.
  - 2. Register renaming can eliminate all register carried WAR hazards.
  - 3. Control hazard penalties can be eliminated by dynamic branch prediction.
- (a) 1, 2 and 3
  - (b) 2 and 3 only
  - (c) 1 and 3 only
  - (d) 1 and 2 only

**Q.111** The use of multiple register windows with overlap causes a reduction in the number of memory accesses for

[GATE 2008]

[2-Marks]

- 1. function locals and parameters
  - 2. register saves and restores
  - 3. instruction fetches
- (a) 1, 2 and 3
  - (b) 3 only
  - (c) 2 only
  - (d) 1 only

**Q.112** Which of the following is/are true of the auto increment addressing mode? [GATE 2008]

[2-Marks]

- 1. It is useful in creating self relocating code
  - 2. If it is included in an Instruction Set Architecture, then an additional ALU is required for effective address calculation
  - 3. The amount of increment depends on the size of the data item accessed
- (a) 2 and 3 only
  - (b) 3 only
  - (c) 2 only
  - (d) 1 only

**Q.113** Which of the following must be true for the RFE (Return from Exception) instruction on a general purpose processor.

[GATE 2008]

1. It must be a trap instruction [2 Marks]
  2. It must be a privileged instruction
  3. An exception can not be allowed to occur during execution of an RFE instruction.
- (a) 1, 2 and 3 only  
 (b) 1 and 2 only  
 (c) 2 only  
 (d) 1 only

**Q.114** Consider a 4 stage pipeline processor. The number of cycles needed by the four instructions  $I_1, I_2, I_3, I_4$  in stages  $S_1, S_2, S_3, S_4$  is shown below

|       | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|-------|-------|-------|-------|-------|
| $I_1$ | 2     | 1     | 1     | 1     |
| $I_2$ | 1     | 3     | 2     | 2     |
| $I_3$ | 2     | 1     | 1     | 3     |
| $I_4$ | 1     | 2     | 2     | 2     |

What is the number of cycles needed to execute the following loop? [GATE 2009]

[2-Marks]

for ( $i = 1$  to 2) { $I_1, I_2, I_3, I_4$ }

- (a) 16  
 (b) 23  
 (c) 28  
 (d) 30

## ANSWER KEY

|     |   |     |   |     |     |     |       |     |       |
|-----|---|-----|---|-----|-----|-----|-------|-----|-------|
| 1   | d | 2   | b | 3   | a   | 4   | c     | 5   | c     |
| 6   | b | 7   | d | 8   | c   | 9   | c     | 10  | c     |
| 11  | a | 12  | c | 13  | c   | 14  | d     | 15  | d     |
| 16  | d | 17  | b | 18  | c   | 19  | c     | 20  | a     |
| 21  | a | 22  | a | 23  | d   | 24  | a     | 25  | d     |
| 26  | b | 27  | c | 28  | a,b | 29  | a     | 30  | b,c   |
| 31  | d | 32  | b | 33  | a,c | 34  | a,b,c | 35  | e     |
| 36  | a | 37  | d | 38  | d   | 39  | a     | 40  | c     |
| 41  | d | 42  | c | 43  | c   | 44  | b     | 45  | b     |
| 46  | c | 47  | c | 48  | d   | 49  | a     | 50  | d     |
| 51  | c | 52  | b | 53  | a   | 54  | c     | 55  | a     |
| 56  | a | 57  | b | 58  | c   | 59  | a     | 60  | c     |
| 61  | d | 62  | b | 63  | d   | 64  | a     | 65  | a     |
| 66  | b | 67  | b | 68  | c   | 69  | c     | 70  | c     |
| 71  | b | 72  | c | 73  | c   | 74  | b     | 75  | a,b,d |
| 76  | b | 77  | c | 78  | c   | 79  | b     | 80  | c     |
| 81  | b | 82  | c | 83  | a   | 84  | d     | 85  | c     |
| 86  | b | 87  | c | 88  | a   | 89  | d     | 90  | d     |
| 91  | d | 92  | b | 93  | d   | 94  | c     | 95  | d     |
| 96  | d | 97  | a | 98  | d   | 99  | b     | 100 | d     |
| 101 | c | 102 | c | 103 | d   | 104 | c     | 105 | a     |
| 106 | d | 107 | a | 108 | b   | 109 | c     | 110 | a     |
| 111 | b | 112 | b | 113 | a   | 114 | d     |     |       |

## SOLUTIONS

**S.1 (d)**

(i), (ii), (iii) are correct statements about immediate addressing mode.

**S.3 (a)**

The program counter contents are incremented each time to point to the next instruction. In this way the **program counter** maintains instruction sequencing required.

**S.4 (c)**

(i) and (ii) both are correct about program counter.

**S.5 (c)**

A computer must have instructions capable of performing all (i), (ii) (iii) (iv) operations.

**S.6 (b)**

(i), (ii), (iii) are valid design method for controllers (i)  $\Rightarrow$  **state table method**, it begins with the construction of state table for the control unit.

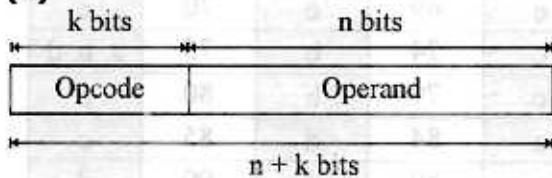
(ii)  $\Rightarrow$  **delay element method**

(iii)  $\Rightarrow$  **sequence counter method**

**S.7 (d)**

(i) and (ii) are correct about horizontal and vertical micro instructions (i) and (ii) are difference between horizontal and vertical micro instructions.

**S.8 (c)**



**Figure : Instruction Format**

Now maximum no. of operation supported by this instruction format:  $2^k$

Maximum no. of addressable memory cells are:  $2^n$

**S.9 (c)**

The diagram shows, **Extended Addressing mode**. In this, the effective memory address is directly specified and it is used by some of the processor and address specified is 16-bit address.

**S.10 (c)**

Addressing modes are either explicitly specified or implied by the instruction.

**S.11 (a)**

Disadvantage of **state table method**.

**S.12 (c)**

The processor use **program counter** and **instruction register** for fetch and execute operations respectively.

**S.13 (c)**

The immediate addressing modes can be used for type (i) and (ii).

**S.14 (d)**

Microinstruction length is determined by all three (i) (ii), (iii)

**S.27 (c)**

The group of operations that a microprocessor can perform is called its **instruction set**.

**S.28 (a, b)**

One grouping of instruction set operations lists the following categories : **arithmetic**, logical, **data transfer**, branch, subroutine call, return and miscellaneous instructions.

**S.29 (a)**

The **instruction decoder** of CPU interprets the op code in the instruction register and directs the control and timing section how to execute the instruction.

**S.30 (b, c)**

Most MPU chips contain atleast an **ALU**, several registers, a **program counter**, instruction-decoding circuitry, a timing and control section, bus buffers and latches, internal busses and control lines and several control inputs and outputs.

**S.32 (b)**

The minimum time delay required between the initiations of two successive memory operations (for example, the time between two successive READ operations) is called memory cycle time. It is usually longer than the access time.

**S.33 (a, c)**

During execution of the current instruction the content is incremented so that it points to the next instruction and also it is a register.

**S.37 (d)**

To specify a particular operation, out of the  $2^n$  possible operations, one needs n-bits. Since  $2^{(m+10)}$  bytes are there, and if it is word addressable, then the number of words is  $2^{(m+10)}$  divided by  $2^3$ , i.e.,  $2^m + 7$  words. So, one needs  $3(m+7) + n = 3m + n + 21$  bits.

**S.41 (d)**

If one CPU completes its operation before the other, the result will be original value. If one CPU has already fetched the value of x, and is on the process of updating it and at this point of time the other CPU fetches the value of x (which will be same as the value fetched by the first CPU), the first CPU after manipulating it will store back the result in x. After this the second CPU will store its manipulated value, over-writing what is stored by the first CPU. So, the final value may be its original value + 1 if the first CPU decremented x, its original value -1, otherwise.

**S.44 (b)**

$2^8 = 256$  combinations.

$$8 \quad 12 \quad 12 = 32 \text{ bits}$$

|        |         |         |
|--------|---------|---------|
| Opcode | Address | Address |
|--------|---------|---------|

(Two address instructions)

$256 - 250 = 6$  combinations can be used for one address.

|                    |                            |
|--------------------|----------------------------|
| Opcode             | Address                    |
| $6 \times 12^{12}$ | (One address instructions) |

Maximum number of one address instruction  
 $= 6 \times 2^{12}$   
 $= 24.576$

**S.45 (b)**

A modulo-k sequence counter can easily be made to behave like a cascade of k-1 delay elements.

**S.46 (c)**

(i) and (ii) are advantages of register indirect addressing.

(ii)  $\Rightarrow$  This mode improves the speed of program execution in situations where data elements are to be accessed from memory.

**S.47 (c)**

(i)  $\Rightarrow$  Control memories are usually ROMs so their contents can not be altered on time.

(ii)  $\Rightarrow$  A processor with a writable control memory is said to be dynamically micro-programmable because the control memory contents can be altered under program control.

**S.48 (d)**

Valid microinstruction types are:

(a) **Branch micro instruction**  $\Rightarrow$  specify no control information

(b) **Operate micro instruction**  $\Rightarrow$  activate control lines but have no branching capacity

**S.49 (a)**

(1)  $\Rightarrow$  Two address instructions

(2)  $\Rightarrow$  One address instructions

(3)  $\Rightarrow$  Zero address instructions

**S.50 (d)**

(i), (ii), (iii), (iv) are valid classes of interrupts.

(i) **program**  $\Rightarrow$  generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero etc.

(ii) **Timer**  $\Rightarrow$  generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.

(iii) **I/O**  $\Rightarrow$  generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.

(iv) **Hardware failure**  $\Rightarrow$  generated by failure such as power failure or memory parity error.

## S.51 (c)

We have to design 16 Kbyte memory or we can say 16K X8 bit memory ( $\because 1\text{byte} = 8\text{ bit}$ ) word length of chip is  $= 12 + 4 = 16 = 2^4$   
 $\therefore$  bits required per chip = 4 bits.

Hence total size of memory chip is  $= 2^{12} \times 4$  bits.

( $2^{14} \rightarrow$  due to 12 address lines in memory chip)  
 $2^{12} \times 4 = 2^{12} \times 2^2 = 2^{14} = 16\text{K bits}$

Means size of memory chip is 16K bits.

Now we have to find how many chips of 16K bits can be constructed using  $16\text{K} \times 8\text{bit}$  memory.

$$\therefore \text{Number of chips} = \frac{16\text{K} \times 8\text{bit}}{16\text{Kbit}} = 8 \text{ chips.}$$

## S.52 (b)

Pipelined version will take  $3 \times 100 + 3 \times 20 = 360\text{ ns}$  per instruction.

## S.53 (a)

$$N = 2^{16} = 64\text{ K}$$

## S.54 (c)

$$M = 2^{12} = 4\text{ K}$$

## S.55 (a)

There are 4-pages or blocks for each 1 K word  
 $n = 4 \times 64 = 256$  pages

## S.56 (a)

As we know memory contains blocks. We have memory space of 4K page size = 256 words.  
 Means 1K contain 4 block of memory.

$$\therefore 4\text{K contain} = 4 \times 4 \text{ block} \\ = 16 \text{ block.}$$

## S.57 (b)

$$m * a = (a * b) * a = (a' + b) * a = (a' + b)' * a = ab' \\ + a = a(b' + 1) = a$$

## S.58 (c)

Memory

|          |             |
|----------|-------------|
| PC → 300 | Opcode mode |
| 301      | 400         |
| R1 = 200 | 302         |

Operand = A

EA = 301

## S.59 (a)

| Memory   |             |
|----------|-------------|
| PC → 300 | Opcode mode |
| 301      | 400         |
| R1 = 200 | 302         |

$$\text{EA} = \text{PC} + \text{offset}$$

$$\text{EA} = 302 + 400 = 702$$

## S.60 (c)

| Memory   |             |
|----------|-------------|
| PC → 300 | Opcode mode |
| 301      | 400         |
| R1 = 200 | 302         |

$$\text{EA} = \text{R1} + \text{offset}$$

$$\text{EA} = 200 + 400 = 600$$

## S.61 (d)

## 3-address instruction

→ two operand locations and a result location are explicitly contained in the instruction word  
 e.g.  $Y = A - B$

## 2-address instruction

One of the address is used to specify both an operand and the result location.

$$\text{e.g. } Y = Y + X$$

## 1-address instruction

Two addresses are implied in the instruction and accumulator based operations

$$\text{e.g. } ACC = ACC - X$$

## 0-address instructions

These are applicable to a special memory organization called a stack. It interacts with a stack using push and pop operations. All addresses are implied as in register based operations.

$$\text{e.g. } T = \text{Top (T-1)}$$

## S.62 (b)

Using the circuit shown in figure

$$\begin{array}{r}
 w \quad 10101 \\
 +x \Rightarrow +11011 \\
 \hline
 a \quad 110000
 \end{array}$$

$$\begin{array}{r}
 +y \quad +010100 \\
 \hline
 z \quad 1000100
 \end{array}$$

⇒ W + X + Y using two ripple carry address.

## S.63 (d)

The wait time can be calculated in the following manner :

$$\text{Time to read one sector} = 0.00010$$

$$\text{Time to write one sector} = 0.00010$$

$$\text{Time to execute 200 instructions} = 0.00010 + 0.00010 = 0.00020$$

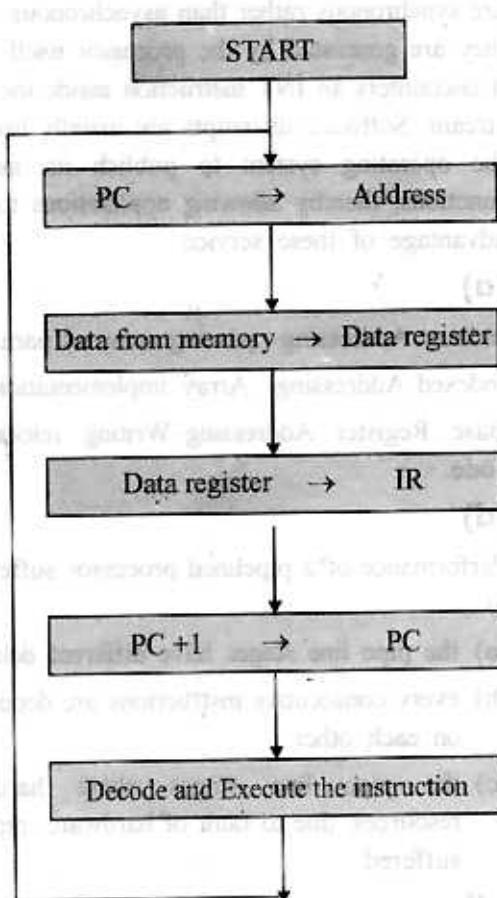
$$\text{Total program cycle time} = \frac{2200 \times 0.00020}{200} = 0.0022$$

$$\text{Utilization time} = \frac{0.00020}{0.0022} \times 100 = 9.09 \approx 9\%$$

$$\text{Wait time} = 100 - 9 = 91\%$$

## S.64 (a)

Sequence of operations in instruction cycle is as follows.



## S.66 (b)

$$\text{Cycle Time pipelined} = \frac{25}{5} + 1 = 6 \text{ ns}$$

## S.67 (b)

Since the unpipelined machine executes all instruction in a single clock cycle, its average time per instruction is simply the clock cycle time. The clock is equal to the sum of the times for each step in execution.

Average instruction execution time =  $10 + 9 + 9 + 7 + 10 = 45$  ns the clock cycle time on the pipelined machine must be the largest time for any stage in the pipeline (10 ns plus the overhead of 1 ns, for a total of 11 ns Since, the Cycles Per Instruction (CPI) is 1, this yields an average instruction execution time of 11 ns

## Speed from pipelining

$$= \frac{\text{Average instruction time unpipelined}}{\text{Average instruction time pipelined}}$$

$$= \frac{45 \text{ ns}}{11 \text{ ns}} = 4.09 \text{ times}$$

## S.68 (c)

Average instruction execution time

$$= \text{clock cycle} * \text{average CPI}$$

$$= 10 \text{ ns} * [(40\% + 20\%) * 40\% * 5]$$

$$= 10 \text{ ns} * 4.4$$

$$= 44 \text{ ns}$$

In the pipelined implementation, the clock must run at the speed of the slowest stage plus overhead which will be  $10 + 1$  or 11 ns, this is the average instruction execution time. Thus, the speed up from pipelining is

(Speed up) pipelining

$$= \frac{\text{Average instruction time unpipelined}}{\text{Average instruction time pipelined}}$$

$$= \frac{44 \text{ ns}}{11 \text{ ns}} = 4 \text{ times}$$

## S.69 (c)

In instruction address calculation phase,

- \* for 16 bits instruction → add 1 to the previous address
- \* for 8 bits instruction → add 2 to the previous address.

**S.70 (c)**

In relative addressing EA is calculated as follows

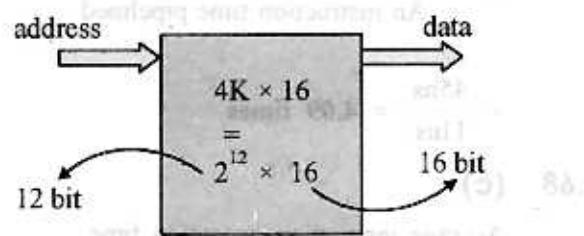
$$EA = \text{Program Counter (PC)} + \text{Offset}$$

Here program counter is used, by this it can be loaded anywhere in the memory without the need to adjust any addresses, this feature makes this addressing position independent.

**S.72 (c)**

It is given that the memory is  $4K \times 16$ . That is, it has 4 words and 16 bits per word.

$4K \text{ words} = 4 \times 2^{10} \text{ words} = 2^{12} \text{ words}$ . Hence it requires 12 bit address line and 16 bit data line.

**S.74 (b)**

Relative mode of addressing is most relevant to writing position independent code.

**S.75 (a, b, d)**

We cannot create pointers, arrays and recursive procedure with local variable.

**S.76 (b)**

By Definition.

**S.77 (c)**

Micro instruction. Each word (i) control memory contains within it a micro instruction. The micro instruction specifies one or more micro operations for the system. A sequence of micro instruction constitutes micro program.

the minimum number of bits = 27

n = number of micro operation

if n = 3 (for minimum), so

$$\text{minimum number of bits} = 2^3 = 8.$$

**S.78 (c)**

Keyboard should get higher priority in assigning interrupt.

**S.79 (b)**

Hardwired control is the fastest. Since horizontal microprogramming involves lesser number of instructions, it is faster than vertical microprogramming.

**S.80 (c)**

Indirect addressing – pointers

Immediate addressing – constants

Autodecrement address – Loops

**S.81 (b)**

Comparing the time  $T_1$  taken for a single instruction on a pipelined CPU with time  $T_2$  taken on a non pipelined but on identical CPU,  $T_1 \geq T_2$ .

**S.82 (c)**

Software interrupts play a major role in the normal operation of a processor. These interrupts are synchronous rather than asynchronous, since they are generated by the processor itself when it encounters an INT instruction inside the code stream. Software interrupts are usually used by the operating system to publish its internal functions, thereby allowing applications to take advantage of these service.

**S.83 (a)**

Indirect Addressing– passing array as parameter

Indexed Addressing– Array implementation

Base Register Addressing– Writing relocatable code.

**S.84 (d)**

Performance of a pipelined processor suffer due to

- (a) the pipe line stages have different delays
- (b) every consecutive instructions are dependent on each other.
- (c) the pipe line stages share hardware resources, due to fault of hardware, pipeline suffered.

**S.85 (d)**

- (i) The  $j+1^{\text{st}}$  instruction uses the result of the  $j^{\text{th}}$  instruction as an operand then read-after-write (RAW) hazard occurs. It is a part of data dependency.

- (ii) The execution of a conditional jump instruction causes a flushing so **conditional dependency occurs**.
- (iii) The  $j^{\text{th}}$  and  $j+1^{\text{st}}$  instructions require the ALU at the same time causes write-after-read (WAR) hazard.

**S.86 (d)**

An array ( $n \times n$ ) multiplier contains  $2n-1$  gate cells unit if each unit (cells) contains a  $\Theta(1)$  then delay then total delay is  $(2n-1) \Theta(1)$  which is  $\Theta(n)$ .

**S.87 (b)**

$$A \leftarrow A_0$$

The value of B after execution of the program is **number of 0 bits in  $A_0$** .

**S.88 (d)**

Insert the instruction

$$\text{RRC } A, \#1$$

**S.89 (a)**

If  $C_0 = 1$  and  $K = 1$

$$S_0 = B'_0 \oplus A_0 + C_0 = B'_0 \oplus A_0$$

then  $S_0 = A_0 B'_0 + B_0 A'_0$

and  $C_1 = A_0 + A'_0 B'_0$

and  $S_1 = A_1 \oplus B'_1 \oplus C_1$

if  $C_0 = 1$  and  $K = 0$

then  $S_0 = A_0 B_0 + A'_0 B'_0$

and  $C_1 = A_0 + A'_0 B_0$

So it computes  **$A + B$ ,  $A - B$  and  $A + 1$**

**S.90 (d)**

Based addressing mode & relative addressing mode are suitable for program relocation at run time due to following reason. In relative addressing scheme in which the operand field contains a relative address also called an offset or displacement D. If  $R_1, R_2, \dots, R_k$  are CPU register then A is computed as follows.

$$A : R + D$$

By changing contents of R, the processor can change the absolute address referred to by a block of instructions B. This address modification permits the processor to move (relocate) the entire block B from one region of main memory to another without invalidating the addresses in B. When used in this way R is referred to as a base register and its content as base addresses.

**S.91 (a)**

K stage pipeline can process n tasks in  $T_k$  time

$$T_k = [K + (n-1)]\tau$$

When  $\tau = \tau_m + d$  where  $\tau_m$  is the maximum stage delay. So, max (150, 120, 160, 140) = 160

$$\tau = 160 + 5$$

$$\tau = 165 \text{ ns}$$

$$\tau = 165 \times 10^{-3} \mu\text{sec}$$

$$K = 4, n = 1000$$

$$T = [4 + (1000-1)]\tau$$

$$= 1003 \times 165 \times 10^{-3}$$

$$= 165.5 \mu\text{sec}$$

**S.92 (b)**

| Instruction  | Instruction Size | Location (Decimal) |
|--------------|------------------|--------------------|
| MOV R1, 5000 | 2                | 1000 to 1007       |
| MOV R2, (R1) | 1                | 1008 to 1011       |
| ADD R2, R3   | 1                | 1012 to 1015       |
| MOV 6000, R2 | 2                | 1016 to 1023       |
| Halt         | 1                | 1024 to 1027       |

If an interrupt occurs the CPU has been halted after executing the HALT instruction the return address 1024 saved in the stack.

**S.93 (c)**

| Operation                              | Instruction Size | Required Clock cycles             |
|----------------------------------------|------------------|-----------------------------------|
| $R_1 \rightarrow \text{Memory}[5000]$  | 2                | $6 + 2 = 8$                       |
| $R_2 \rightarrow \text{Memory}[(R_1)]$ | 1                | $3 + 2 = 5$                       |
| $R_2 \leftarrow R_2 + R_3$             | 1                | $1 = 1$                           |
| Memory [6000] $\leftarrow R_2$         | 2                | $6 + 2 = 8$                       |
| Machine Halt                           | 1                | $\frac{1+1=2}{\text{Total} = 24}$ |

**S.94 (c)**

| Level       | Access time |
|-------------|-------------|
| 1           | 1 ns        |
| 2           | 10 ns       |
| Main memory | 500 ns      |

$$\text{So average access time} = (0.8 \times 1 + 0.2 \times 0.9 \times 10 + 0.2 \times 0.1 \times 500) \text{ ns}$$

$$= 12.6 \text{ ns}$$

0.2 and 0.1 is miss rate of level 2 and level 1 respectively.

**S.95 (d)**

$R_1$ ,  $R_2$  and  $R_3$  are registers.

Given sequence of instructions are

$$R_1 \rightarrow M[100]$$

$$M[100] \rightarrow R_2$$

$$M[100] \rightarrow R_3$$

Thus the content of  $R_1$  is loaded to  $M[100]$  as well as  $R_2$  and  $R_3$ . So equivalent instruction can be written as

$$R_1 \rightarrow R_2$$

$$R_1 \rightarrow R_3$$

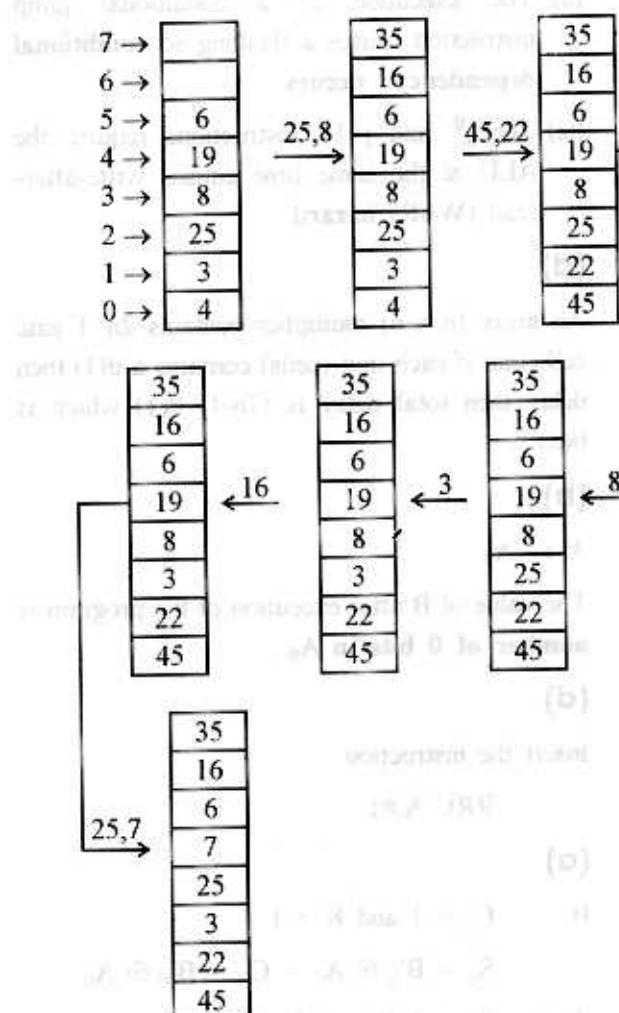
$$R_1 \rightarrow M[100]$$

in any sequence.

**S.96 (d)**

Given sequence is-

$$4, 3, 25, 8, 19, 6, 25, 8, 16, 35, 45, 22, 8, 3, 16, 25, 7$$



∴ 7 will be in memory block.

**S.97 (a)**

Let total operations are 100 and total time taken is 100 second.

$$\text{floating plant operations are } \frac{2}{5} \times 100 = 40$$

$$\text{Fixed point operation are } \frac{3}{5} \times 100 = 60$$

Let time taken by floating point operation =  $t_1$

Let time taken by floating point operation =  $t_2$

$$\text{So, } t_1 + t_2 = 100 \quad \dots(1)$$

time taken by 40 floating point operation =  $t_1$

$$\text{time taken by 1 floating point operation} = \frac{t_1}{40}$$

Similarly one fixed point operations taken time

$$= \frac{t_2}{60}$$

$$\begin{aligned} \therefore \frac{t_1}{40} &= 2 \times \frac{t_2}{60} \\ \Rightarrow 3t_1 &= 4t_2 \end{aligned} \quad \dots(2)$$

from equation (1) and (2)  $t_1 = \frac{400}{7}$   $t_2 = \frac{300}{7}$

after speed up  $t'_1 = \frac{t_1}{1.2} = \frac{400}{1.2 \times 7} = \frac{4000}{87}$

$$t'_2 = \frac{t_2}{1.1} = \frac{300}{1.1 \times 7} = \frac{3000}{77}$$

$$t'_1 t'_2 = \frac{4000}{84} + \frac{3000}{77}$$

$$\therefore t \propto \frac{1}{5}$$

$$\begin{aligned} \text{So } S_2 &= \frac{t}{t'} \times S_1 = \frac{100}{\frac{4000}{84} + \frac{3000}{77}} \times S_1 \\ &= 1.155 S_1 \end{aligned}$$

### S.98 (d)

Notice here that S5 have to be activated on T1 for all instructions and for I2 and I4 on T3.

S10 have to be activated.

for I2 and I4 on T5

for I1 and I3 on T4

for I4 on T3

for I2 and I3 on T2

### S.99 (b)

The correct matching is

| List I |                 | List II |                     |
|--------|-----------------|---------|---------------------|
| (A)    | $A[I] = B[J]$ ; | 1.      | Indexed addressing  |
| (B)    | While(*A++);    | 2.      | Auto increment      |
| (C)    | int temp = *X;  | 3.      | Indirect addressing |

### S.100 (d)

ADD A[R0],@B

A[R0] uses indexed addressing mode as follow with R0 as the index register

$$\text{Regs [R3]} \leftarrow \text{Regs [R3]} + \text{Mem [Regs[B] + Regs[R0]]} \quad \dots(1)$$

And @ B uses indirect addressing as follows

ADD A[R0],@B

$$\text{Regs [R0]} \leftarrow \text{Regs [R0]} + \text{Mem [Mem[B]]} \quad \dots(2)$$

So the first instruction needed 3 memory cycles and the second needed 2 and one memory cycle is needed for writing the result into the memory. So total 6 memory cycles are needed to execute ADD A[R0],@B.

### S.101 (a)

Size of instruction is 24 bits = 3 bytes.

Starting address of the program is 300. The size of instruction is 3 byte long. So the address is always the multiple of 3 byte. Therefore, the next address is 600 and it is also the next instruction of the program.

### S.97 (c)

Size of cache = 32KB

$$= 32 \times 2^{10} \text{ byte}$$

$$= 2^5 \times 2^{10} \text{ byte}$$

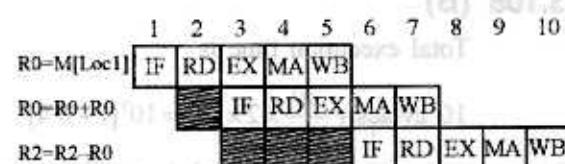
$$= 2^{15} \text{ byte} = 15 \text{ bits}$$

Size of tag =  $32 - 15 = 17$  bits

Cache indexing size = 10 bits

### S.103 (d)

Clock Cycles



So it requires 10 CPU clock cycles.

### S.104 (c)

RTL  $R0 \leftarrow R0 + R1$

The sequence of microinstruction take place in following cycles.

I cycle:  $R1_{out}, S_{in}$

II cycle:  $R2_{out}, T_{in}$

III cycle:  $S_{out}, T_{out}, ALU_{add}, R_{in}$

$\therefore$  3 cycles will be required to execute.

**S.105 (a)**

$$\text{RTL} \quad R_n = PC + 1$$

$$P_c = M[PC]$$

The sequence of microinstruction take place in following cycles.

I cycle;  $PC_{out}$ ,  $S_{in}$ ,  $MAR_{in}$  (MAR can be loaded with  $PC_{out}$ )

II cycle;  $S_{out}$ , ALU increment,  $R_n$  in

III cycle;  $MDR_{out}$ ,  $PC_{in}$  ( $MDR_{out}$  can be performed once  $MAR_{in}$  has been performed)

$\therefore$  3 cycles will be required to execute.

**S.106 (d)**

bbs reg, pos, label

temp  $\leftarrow$  reg & mark

Branch to label if temp is non zero. If we shift one position left by pos of  $0 \times 1$  then it evaluate the bbs instruction and mask  $\leftarrow 0 \times 1 \ll pos$ .

**S.107 (a)**

Cache block size = 64 bytes

Main memory has k banks or  $k = 24$

Each bank is 2 byte long because  $c = 2$

Total time for one parallel access

$$T = K/2 + \text{Latency}$$

$$= 24/2 + 80 = 12 + 80$$

$$= 92 \text{ ns}$$

Total latency time =  $cT$

$$= 2 \times 92 = 184 \text{ ns}$$

**S.108 (b)**

Total execution time is

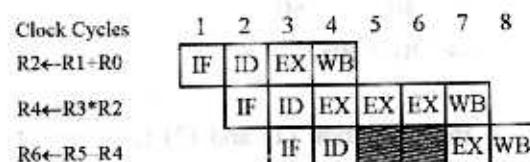
$$10^9 \text{ cycles} + \frac{20}{100} \times 2 \times 10^9 = 10^9 [1 + 0.4] \\ = 1.4 \times 10^9 \text{ cycles} \\ = 1.4 \times 1 \text{ sec} \\ = 1.4 \text{ sec}$$

**S.109 (c)**

Pipelined processor has four stages IF, ID, EX, WB

| Clock Cycles | Instruction |
|--------------|-------------|
| 1            | ADD         |
| 1            | SUB         |
| 3            | MUL         |

Consider the following diagram



So total required clocks cycle is 8.

**S.110 (a)**

All statement 1, 2 and 3 are false for a pipelined processor.

**S.111 (b)**

The use of multiple register windows with overlap causes a reduction in the number of memory access **instruction fetches** only.

**S.112 (b)**

In auto increment addressing mode the amount of increment depends on the size of the data item accessed. Consider the following example.

$$\text{Regs}[R1] \leftarrow \text{Regs}[R1] + \text{Mem}[\text{Regs}[R2]]$$

$$\text{Regs}[R2] \leftarrow \text{Reg}[R2] + d$$

Where d is the size of the data item.

**S.113 (a)**

A RFE (Return From Exception) instruction contains a trap instruction, privileged instruction and an exception can't be allowed to occur during the execution of an RFE instruction.

**S.114 (d)**

No. of cycles needed to execute then for loop when

$$i = 1$$

$$= 2 + 1 + 3 + 2 + 2 + 3 + 2 = 15$$

So total cycle needed to execute the loop

$$= 2 \times 15$$

$$= 30$$

Memory organization is concerned with the way memory is divided into blocks and how these blocks are addressed. It includes the concepts of segmentation, paging, and mapping.

## MEMORY ORGANIZATION

**3.3**

### LEVEL-1

- Q.1** Which of the following statement is correct?
- paging system is invented to get a large linear address space without having to buy more physical memory
  - segmentation is invented to allow programs and data to be broken up into logically independent address space and to aid sharing and protection.
  - Both (a) and (b)
  - neither (a) nor (b)
- Q.2** In \_\_\_\_\_ the total address space exceeds the size of physical memory
- paging
  - segmentation
- (i), (ii)
  - (i)
  - (ii)
  - None of these
- Q.3** In \_\_\_\_\_ the procedure and data is distinguished and separately protected.
- segmentation
  - mapping
  - paging
  - memory

- Q.4** A \_\_\_\_\_ is a software program stored in ROM that cannot be changed easily.

- Loader
- Editor
- Firmware
- None of these

- Q.5** Match the following:

|       | Group I               | Group II                                                             |
|-------|-----------------------|----------------------------------------------------------------------|
| (i)   | Contiguous allocation | (a) supports direct access without external fragmentation            |
| (ii)  | Linked allocation     | (b) Fastest and easiest for random access files                      |
| (iii) | Indexed allocations   | (c) No external fragmentation and file can grow without complication |

- (i)-(b) (ii)-(c) (iii)-(a)
- (i)-(c) (ii)-(b) (iii)-(a)
- (i)-(c) (ii)-(a) (iii)-(b)
- All

- Q.6** In a virtual memory system the address space specified by the address lines of the CPU must be \_\_\_\_\_ than the physical memory size and \_\_\_\_\_ than the secondary storage size
- smaller, smaller
  - larger, smaller
  - smaller, larger
  - larger, larger
- Q.7** Associative memories are also commonly known as \_\_\_\_\_
- Virtual memory
  - Look through
  - Content addressable memories
  - direct mapped cache memory
- Q.8** Which statement is false about flash memory?
- flash memory is intermediate between EPROM and EEPROM in both cost and functionality.
  - flash memory uses an electrical erasing technology
  - An entire flash memory can be erased in one or a few seconds, which is much slower than EPROM.
  - flash memory uses only one transistor per bit, and so achieves the high density of EEPROM.
- Q.9** The ALU of a computer normally contains a number of high speed storage elements called
- semi conductor memory
  - registers
  - hard disk
  - magnetic disks
- Q.10** Which of the following is fastest?
- CPU
  - Magnetic tapes and disks
  - Video terminals
  - Sensors mechanical controllers
- Q.11** Which of the following enables peripherals to pass a signal down the bus to the next devices on the bus during polling of the device?
- DMA
  - Interrupt vectoring
  - daisy chain
  - cycle stealing
- Q.12** In case of direct mapping of cache, the mapping is expressed as \_\_\_\_\_.
- cache line number = (main memory block number) modulo (number of lines in the cache)
  - cache line number = (number of lines in the cache) modulo (main memory block number)
  - number of lines in the cache = (cache line number) modulo (number of lines in the cache)
  - number of lines in the cache = (number of lines in the cache) modulo (cache line number).
- Q.13** Which of the following consists of an array of gates which can be used to patch an error in a ROM or alter the meaning of instruction?
- PSW
  - PLA
  - EAPROM
  - RAM
- Q.14** The larger the RAM of a computer, the faster is its speed, since it eliminates
- frequent disk I/Os
  - need for ROM
  - need for a data-wide path
  - none of the above
- Q.15** EPROM consist of
- MOSFETs
  - Bipolar transistors
  - Diodes
  - Easily erasable

- Q.16** Which of the following is not true of a magnetic disk?
- Users can easily update records by writing over the old data
  - It does not provide an automatic audit trail
  - It provides only sequential access to stored data
  - It is expensive relative to magnetic tape
- Q.17** The CPU of a computer transfers print output to a temporary disk memory at high speed and then gets back to processing another job without waiting for the output to go to the printer. In this way, the CPU does not remain idle due to its own high speed as compared to the low speed of the printer. What is the name of this memory?
- ROM
  - External memory
  - Buffer memory
  - None of the above
- Q.18** The most popular secondary storage today is
- Floppy disk
  - Mass storage
  - Magnetic tape
  - Semi-conductor
- Q.19** Which of the following memories has the shortest access time?
- RAM
  - Magnetic core memory
  - Magnetic memory
  - Cache memory
- Q.20** Specific applications of ROM and RAM in micro-computers
- RAM is used to store print programs
  - RAM is used as a scratch pad memory
  - ROM is used as a scratch pad memory
  - None of the above
- Q.21** Use of 'program inhibit mode' in EPROM 2764?
- program same data for different address
  - program with different data for same addresses
  - both (a) and (b)
  - neither (a) nor (b)
- Q.22** What is the maximum memory that can be accessed by 16 address lines?
- 64 K
  - 48 K
  - 4 K
  - 2 K
- Q.23** PROMs are used to store
- information to be accessed rarely
  - relatively permanent information
  - sequential information
  - bulk information
- Q.24** If the microcomputer accesses memory location 2001 H and if the storage device is placed in the write mode, a \_\_\_\_\_ of data from the data bus will be placed in the \_\_\_\_\_.
- byte, RAM
  - nibble, RAM
  - byte, ROM
  - nibble, ROM
- Q.25** The RAM in a memory map is said to reside on what page?
- 01 H
  - 10 H
  - 20 H
  - 21 H
- Q.26** Bipolar devices are desirable in the fabrication of which of the following components?
- Main memory
  - Cache memory
  - Micro program memory
  - All of the above
- Q.27** The idea of cache memory is based on the
- property of locality of reference
  - fact that only a small portion of a program is referenced relatively frequently
  - heuristic 90-10 rule
  - fact that reference generally tend to cluster

**Q.28** If memory access takes 20 ns with cache and 110 ns without it, then the hit-ratio, (cache uses a 10 ns memory) is,

- (a) 93%
- (b) 90%
- (c) 87%
- (d) 88%

**Q.29** If the cache needs an access time of 20 ns and the main memory 120 ns, then the average access time of a CPU is (assume hit-ratio is 80%)

- (a) 30 ns
- (b) 40 ns
- (c) 35 ns
- (d) 45 ns

**Q.30** FFFF will be the last memory location in a memory of size

- (a) 1 k
- (b) 16 k
- (c) 32 k
- (d) 64 k

**Q.31** A computer with a 32-bit wide data bus uses 4 K × 8 static RAM memory chips. The smallest memory this computer can have is

- (a) 32 KB
- (b) 16 KB
- (c) 8 KB
- (d) 24 KB

**Q.32** In a vectored interrupt the

- (a) branch address is assigned to a fixed location in memory
- (b) interrupting source supplies the branch information to the processor through an interrupt vector
- (c) branch address is obtained from a register in the processor
- (d) None of the above

**Q.33** Von Neumann architecture is

- (a) SISD
- (b) SIMD
- (c) MIMD
- (d) MISD

**Q.34** A magnetic disk system has the following parameters.

$$T_s = \text{average time to position the magnetic head over a track}$$

$$R = \text{rotation speed of disk in revolutions per second}$$

$$N_t = \text{number of bits per track}$$

$$N_s = \text{number of bits per sector}$$

Calculate the average time  $T_a$  that it will take to read one sector.

$$(a) T_a = T_s + \frac{1}{2R} + \frac{N_t}{N_s} \times \frac{1}{R}$$

$$(b) T_a = T_s + 2R + \frac{N_t}{N_s} \times \frac{1}{R}$$

$$(c) T_a = T_s + \frac{1}{2R} + \frac{N_s}{N_t} \times \frac{1}{R}$$

- (d) none of the above

## LEVEL-2

**Q.35** For a memory system Hit time is given as 2 ns with miss rate of 1.5% and miss penalty of 60 ns, then average memory access time =

- (a) 2.98 ns
- (b) 2.9 ns
- (c) 4.19 ns
- (d) 3.98 ns

**Q.36** Consider a display that has a display format of  $80 \times 50$  characters with a  $6 \times 15$  character cell. What is the video buffer RAM for the display to be used in monochrome (1bit per pixel) graphics mode?

- (a) 360000
- (b) 6000
- (c) 3600
- (d) 600000

**Q.37** For a memory system, the average access time is 4.5ns with hit time 3ns and miss rate of 2.5% then miss penalty \_\_\_\_\_

- (a) 80 ns
- (b) 100 ns
- (c) 60 ns
- (d) 55 ns

**Q.38** If the page size is n bytes the average amount of space wasted in the last page of a program by internal fragmentation will be \_\_\_\_\_

- (a)  $n/3$  bytes
- (b)  $n/2$  bytes
- (c) n bytes
- (d) none of these

**Q.39** Which of the following is/are not an advantage of virtual memory?

- (a) Faster access to memory on an average.
- (b) Processes can be given protected address spaces.
- (c) Linker can assign addresses independent of where the program will be loaded in physical memory.
- (d) Programs larger than the physical memory size can be run.

**Q.40** Consider a system in which, the random access of memory is used with following specifications:

- (i) cycle time = 20 ns
  - (ii)  $T_A$  = Average access time = 22 ns  
then calculate data transfer rate for RAM.
- (a)  $0.05 \times 10^8$
  - (b)  $0.05 \times 10^9$
  - (c)  $5 \times 10^{-8}$
  - (d)  $5 \times 10^{-9}$

**Q.41** A 100 K-byte memory is managed using variable partitions but no compaction. It currently has two partitions of sizes 200K bytes and 260K bytes respectively. The smallest allocation request in K bytes that could be denied is for

- (a) 541
- (b) 231
- (c) 181
- (d) 121

**Q.42** Listed below are some operating system abstractions (in the left column) and the hardware components (in the right column)?

| Group I |                       | Group II |           |
|---------|-----------------------|----------|-----------|
| (A)     | Thread                | 1.       | Interrupt |
| (B)     | Virtual address space | 2.       | Memory    |
| (C)     | File system           | 3.       | CPU       |
| (D)     | Signal                | 4.       | Disk      |

Codes:

- |     | A | B | C | D |
|-----|---|---|---|---|
| (a) | 4 | 1 | 2 | 3 |
| (b) | 3 | 2 | 4 | 1 |
| (c) | 1 | 2 | 3 | 4 |
| (d) | 2 | 4 | 3 | 1 |

**Q.43** Choose the correct statement(s) from the following.

- (a) PROM contains a programmable AND array and a fixed OR array
- (b) PROM contains a fixed AND array and a programmable OR array
- (c) PLA contains a fixed AND array and a programmable OR array
- (d) PLA contains a programmable AND array and a programmable OR array

**Q.44** If the microcomputer addresses memory location 0001H, the ROM will output \_\_\_\_\_ (8-bits).

- (a) 00000001
- (b) 00001111
- (c) 01010101
- (d) None of the above

**Common Data For Questions 45 to 47:**

A memory stores 8 K of 16 bits words.

**Q.45** No. of data input/output lines, it has

- (a) 16
- (b) 14
- (c) 12
- (d) 10

**Q.46** No. of address lines it has

- (a) 17
- (b) 15
- (c) 13
- (d) 11

**Q.47** No. of bytes (capacity)

- (a) 20 KB
- (b) 18 KB
- (c) 16 KB
- (d) 14 KB

### LEVEL-3

**Q.48** Consider the memory system, in which cache time = 100 ns and main memory access time 1200 ns. If we would like to have effective (average) memory access time to be or more than 20% higher than cache access time, then the hit ratio for the cache must at least be

- (a) 85%
- (b) 82%
- (c) 98%
- (d) 99%

**Q.49** For a machine assume that the cache miss penalty is 50 clock cycles, and all instructions normally take 2 clock cycles. Assume the miss rate is 2% and there is an average of 1.33 memory references per instruction. What is impact on performance when behavior of cache is included?

- (a) 68.5
- (b) 58.7
- (c) 78.2
- (d) None of these

**Q.50** The memory read operation takes 20 ns as cache access time, and 28 ns as main memory access time,  $h=6/7$  then what will be the mean access time ?

- (a) 16 ns
- (b) 48 ns
- (c) 28 ns
- (d) 24 ns

**Q.51** What is the impact of two different cache organizations on the performance of a CPU? Assume that the CPI with a perfect cache is 2.0 and the clock cycle time 2 ns, that there are 1.3 memory references per instruction, and that the size of both cache is 64 KB and both have block size of 32 bytes. One cache is direct mapped and the other is two-way set associative. Assume CPU clock cycle time with set associative cache is stretched 1.10 times than the other. The cache miss penalty is 70 ns for either cache organization. Calculate the average memory access time and CPU performance.

Assume the hit time is one clock cycle. Also the miss rate of a direct-mapped 64 KB cache is 1.4% and the miss rate for a two-way set-associative cache is of the same size is 1.0%.

- (a) 1.01
- (b) 10.1
- (c) 1.09
- (d) 1.001

### GATE QUESTIONS

**Q.52** The refreshing rate of dynamic RAMs is in the range of [GATE 1987]

- (a) 3 microseconds
- (b) 2 milliseconds
- (c) 50 milliseconds
- (d) 500 milliseconds

- Q.53** The total size of address space in a virtual memory system is limited by [GATE 1991]
- the length of MAR
  - the available secondary storage
  - the available main memory
  - All of the above
  - None of the above
- Q.54** A computer system has a 4K word cache organized in block-set-associative manner with 4 blocks per set, 64 words per block. The number of bits in the SET and WORD fields of the main memory address format is [GATE 1995]
- 15, 40
  - 6, 4
  - 7, 2
  - 4, 6
- Q.55** A ROM is used to store the table for multiplication of two 8-bit unsigned integers. The size of ROM required is [GATE 1996]
- $256 \times 16$
  - $64K \times 8$
  - $4K \times 6$
  - $64K \times 16$
- Q.56** The main memory of a computer has  $2 cm$  blocks while the cache has  $2 c$  blocks. If the cache uses the set associative mapping scheme with 2 blocks per set, then block  $k$  of the main memory maps to the set [GATE 1999]
- $(k \bmod m)$  of the cache
  - $(k \bmod c)$  of the cache
  - $(k \bmod 2c)$  of the cache
  - $(k \bmod 2cm)$  of the cache
- Q.57** More than one word are put in one cache block to [GATE 2001]
- exploit the temporal locality of reference in a program
  - exploit the spatial locality of reference in program
  - reduce the miss penalty
  - none of the above
- Q.58** Which of the following is not a form of memory? [GATE 2002]
- instruction cache
  - instruction register
  - instruction opcode
  - translation lookaside buffer
- Q.59** A hard disk with a transfer rate of 10 M bytes/second is constantly transferring data to memory using DMA. The processor runs at 600 MHz, and takes 300 and 900 clock cycles to initiate and complete DMA transfer respectively. If the size of the transfer is 20 Kbytes, what is the percentage of processor time consumed for the transfer operation? [GATE 2004]
- 0.1%
  - 5.0%
  - 1.0%
  - 0.5%
- Q.60** Consider a small two-way set-associative cache memory, consisting of four blocks. For choosing the block to be replaced, use the least recently used (LRU) scheme. The number of cache misses for the following sequence of block addresses is 8, 12, 0, 12, 8 [GATE 2004]
- 2
  - 3
  - 5
  - 4
- Q.61** What is the minimum size of ROM required to store the complete truth table of an 8-bit  $\times$  8-bit multiplier? [IT-GATE 2004]
- $32 K \times 16$  bits
  - $64 K \times 16$  bits
  - $16 K \times 32$  bits
  - $64 K \times 32$  bits

**Q.62** In a virtual memory system, size of virtual address is 32bit, size of physical address is 30 bit, page size is 4 Kbyte and size of each page table entry is 32-bit. The main memory is byte addressable. Which one of the following is the maximum number of bits that can be used for storing protection and other information in each page table entry? [IT-GATE 2004]

[2-Marks]

- (a) 2
- (b) 10
- (c) 12
- (d) 14

**Q.63** A dynamic RAM has a memory cycle time of 64 nsec. It has to be refreshed 100 times per msec and each refresh takes 100 nsec. What percentage of the memory cycle time is used for refreshing? [IT-GATE 2005]

[1 Mark]

- (a) 10
- (b) 6.4
- (c) 1
- (d) 0.64

**Q.64** In a computer system, four files of size 11050 bytes, 4990 bytes, 5170 bytes and 12640 bytes need to be stored. For storing these files on disk, we can use either 100 byte disk blocks or 200 byte disk blocks (but can't mix block sizes). For each block used to store a file, 4 bytes of bookkeeping information also needs to be stored on the disk. Thus, the total space used to store a file is the sum of the space taken to store the file and the space taken to store the bookkeeping information for the blocks allocated for storing the file. A disk block can store either bookkeeping information for a file or data from a file, but not both.

What is the total space required for storing the files using 100 byte disk blocks and 200 byte disk blocks respectively? [IT-GATE 2005]

[2-Marks]

- (a) 35400 and 35800 bytes
- (b) 35800 and 35400 bytes
- (c) 35600 and 35400 bytes
- (d) 35400 and 35600 bytes

**Q.65** Increasing the RAM of a computer typically improves performance because [GATE 2005] [1-Mark]

- (a) Virtual memory increases
- (b) Larger RAMs are faster
- (c) Fewer segmentation faults occur
- (d) Fewer page faults occur

**Q.66** Consider a disk drive with the following specifications 16 surfaces, 512 tracks/surface, 512 sectors/track, 1KB/sector, rotation speed 3000 rpm. The disk is operated in cycle stealing mode whereby whenever one byte word is ready it is sent to memory; similarly, for writing, the disk interface reads a 4 byte word from the memory in each DMA cycle. Memory cycle time is 40 nsec. The maximum percentage of time that the CPU gets blocked during DMA operation is [GATE 2005]

- (a) 25
- (b) 10
- (c) 40
- (d) 50

#### Common Data For Questions 67 & 68:

Consider two cache organizations: The first one is 32 KB 2-way set associative with 32-byte block size. The second one is of the same size but direct mapped. The size of an address is 32 bits in both cases. A 2-to-1 multiplexer has latency of 0.6 ns while a k-bit comparator has a latency of  $k/10$  ns. The hit latency of the set associative organization is  $h_1$  while that of the direct mapped is  $h_2$ .

**Q.67** The value of  $h_1$  is [GATE 2006]

- (a) 2.4 ns
- (b) 2.3 ns
- (c) 1.8 ns
- (d) 1.7 ns

**Q.68** The value of  $h_2$  is [GATE 2006]

- (a) 2.4 ns
- (b) 2.3 ns
- (c) 1.8 ns
- (d) 1.7 ns

**Q.69** Consider a disk pack with 16 surfaces, 128 tracks per surface and 256 sectors per track. 512 bytes of data are stored in a bit serial manner in a sector. The capacity of the disk pack and the number of bits required to specify a particular sector in the disk are respectively

[GATE 2007]

[1-Mark]

- (a) 256 Mbytes, 28 bits
- (b) 512 Mbytes, 20 bits
- (c) 256 Mbytes, 19 bits
- (d) 64 Gbytes, 28 bits

### Common Data For Questions 70 to 72:

Consider the following program segment. Here R1, R2 and R3 are the general purpose registers.

| Instruction        | Operation               | Instruction size(no. of words) |
|--------------------|-------------------------|--------------------------------|
| MOV R1,(3000)      | R1 $\leftarrow$ M[3000] | 2                              |
| LOOP: MOV R2, (R3) | R2 $\leftarrow$ M[R3]   | 1                              |
| ADD R2, R1         | R2 $\leftarrow$ R1+R2   | 1                              |
| MOV (R3), R2       | M[R3] $\leftarrow$ R2   | 1                              |
| INC R3             | R3 $\leftarrow$ R3+1    | 1                              |
| DEC R1             | R1 $\leftarrow$ R1-1    | 1                              |
| BNZ LOOP           | Branch on not zero      | 2                              |
| HALT               | Stop                    | 1                              |

Assume that the content of memory location 3000 is 10 and the content of the register R3 is 2000. The content of each of the memory locations from 2000 to 2010 is 100. The program is loaded from the memory location 1000. All the numbers are in decimal.

**Q.70** Assume that the memory is word addressable. After the execution of this program, the content of memory location 2010 is [GATE 2007]

[2-Marks]

- (a) 110
- (b) 101
- (c) 102
- (d) 100

**Q.71** Assume that the memory is byte addressable and the word size is 32 bits. If an interrupt occurs during the execution of the instruction "INC R3", what return address will be pushed on to the stack?

[GATE 2007]

[2-Marks]

- (a) 1040
- (b) 1020
- (c) 1024
- (d) 1005

**Q.72** Assume that the memory is word addressable. The number of memory references for accessing the data in executing the program completely is

[GATE 2007]

[2-Marks]

- (a) 21
- (b) 11
- (c) 20
- (d) 10

**Q.73** Consider a 4-way set associative cache consisting of 128 lines with a line size of 64 words. The CPU generates a 20-bit address of a word in main memory. The number of bits in the TAG, LINE and WORD fields are respectively

[GATE 2007]

[1-Mark]

- (a) 7, 7, 6
- (b) 9, 6, 5
- (c) 7, 5, 8
- (d) 9, 5, 6

### Common Data For Questions 74 & 75:

Consider a machine with a byte addressable main memory of  $2^{16}$  bytes. Assume that a direct mapped data cache consisting of 32 lines of 64 bytes each is used in the system. A  $50 \times 50$  two-dimensional array of bytes is stored in the main memory starting from memory location 1100H. Assume that the data cache is initially empty. The complete array is accessed twice. Assume that the contents of the data cache do not change in between the two accesses.

**Q.74** How many data cache misses will occur in total?

[GATE 2007]

- (a) 56 [2-Marks]
- (b) 48
- (c) 59
- (d) 50

**Q.75** Which of the following lines of the data cache will be replaced by new blocks in accessing the array

[GATE 2007]

[2-Marks]

- (a) line 0 to line 7
- (b) line 4 to line 11
- (c) line 0 to line 8
- (d) line 4 to line 12

**Q.76** In an instruction execution pipeline, the earliest that the data TLB (Translation Lookaside Buffer) can be accessed is

[GATE 2008]

[2-Marks]

- (a) after data cache lookup has completed
- (b) after effective address calculation has completed
- (c) during effective address calculation
- (d) before effective address calculation has started

**Q.77** For a magnetic disk with concentric circular tracks, the seek latency is not linearly proportional to the seek distance due to

[GATE 2008]

[2-Marks]

- (a) use of unfair arm scheduling policies
- (b) higher capacity of tracks on the periphery of the platter
- (c) arm starting and stopping inertia
- (d) non-uniform distribution of request

### Common Data For Questions 78 to 80:

Consider a machine a 2-way set associative data cache of size 64 kbytes and block size 16 bytes. The cache is managed using 32 bit virtual addresses and the page size is 4 Kbytes. A program to be run on this machine begins as follows:

```
double ARR[1024][1024]
int i, j;
/* Initialize array ARR to 0.0 */
for (i = 0; i<1024; i++)
    for (j = 0; k<1024; j++)
        ARR [i][j] = 0.0;
```

The size of double is 8 bytes. Array ARR is located in memory starting at the beginning of virtual page 0xFF000 and stored in row major order. The cache is initially empty and no prefetching is done. The only data memory references made by the program are those to array ARR.

**Q.78** The total size of the tags in the cache directory is

[GATE 2008]

[2-Marks]

- (a) 68 kbits
- (b) 64 kbits
- (c) 34 kbits
- (d) 32 kbits

**Q.79** Which of the following array elements has the same cache index as ARR[0][0]?

[GATE 2008]

[2-Marks]

- (a) ARR[5][0]
- (b) ARR[0][5]
- (c) ARR[4][0]
- (d) ARR[0][4]

**Q.80** The cache hit ratio for this initialization loop is

[GATE 2008]

[2-Marks]

- (a) 75%
- (b) 50%
- (c) 25%
- (d) 0%

**Q.81** For inclusion to hold between two cache levels L1 and L2 in a multilevel cache hierarchy, which of the following are necessary?

[GATE 2008]

[2-Marks]

1. L1 must be a write-through cache
  2. L2 must be a write-through cache
  3. The associativity of L2 must be greater than that of L1
  4. The L2 cache must be at least as large as the L1 cache
- (a) 1, 2, 3 and 4  
 (b) 1, 2 and 4 only  
 (c) 1 and 4 only  
 (d) 4 only

**Q.82** How many  $32K \times 1$  RAM chips are needed to provide a memory capacity of 256 K-bytes?

[GATE 2009]

[1-Mark]

- (a) 8  
 (b) 32  
 (c) 64  
 (d) 128

#### Common Data For Questions 83 & 84:

A hard disk has 63 sectors per track, 10 platters each with 2 recording surfaces and 1000 cylinders. The address of a sector is given as a triple  $\langle c, h, s \rangle$ , where c is the cylinder number, h is the surface number and s is the sector number. Thus, the 0<sup>th</sup> sector is addressed as  $\langle 0, 0, 0 \rangle$ , the 1<sup>st</sup> sector as  $\langle 0, 0, 1 \rangle$ , and so on.

**Q.83** The address  $\langle 400, 16, 29 \rangle$  corresponds to sector number:

[GATE 2009]

[2-Marks]

- (a) 505035  
 (b) 505036  
 (c) 505037  
 (d) 505038

**Q.84** The address of 1039<sup>th</sup> sector is:

[GATE 2009]

[2-Marks]

- (a)  $\langle 0, 15, 31 \rangle$   
 (b)  $\langle 0, 16, 30 \rangle$   
 (c)  $\langle 0, 16, 31 \rangle$   
 (d)  $\langle 0, 17, 31 \rangle$

**Q.85** Consider a 4-way set associative cache (initially empty) with total 16 cache blocks. The main memory consists of 256 blocks and the request for memory blocks is in the following order:

0, 255, 1, 4, 3, 8, 133, 159, 216, 129, 63, 8, 48, 32, 73, 92, 155

Which one of the following memory block will NOT be in cache if LRU replacement policy is used?

[GATE 2009]

[2-Marks]

- (a) 3  
 (b) 8  
 (c) 129  
 (d) 216

- (e) 63  
 (f) 133  
 (g) 159  
 (h) 129

- (i) 8  
 (j) 48  
 (k) 32  
 (l) 73

# ANSWER KEY

|    |      |    |     |    |   |    |   |    |   |
|----|------|----|-----|----|---|----|---|----|---|
| 1  | c    | 2  | a   | 3  | a | 4  | c | 5  | a |
| 6  | b    | 7  | c   | 8  | c | 9  | b | 10 | a |
| 11 | c    | 12 | a   | 13 | b | 14 | a | 15 | a |
| 16 | c    | 17 | c   | 18 | b | 19 | d | 20 | b |
| 21 | b    | 22 | a   | 23 | a | 24 | a | 25 | c |
| 26 | b, c | 27 | all | 28 | b | 29 | b | 30 | d |
| 31 | b    | 32 | a   | 33 | a | 34 | c | 35 | b |
| 36 | a    | 37 | c   | 38 | b | 39 | c | 40 | b |
| 41 | a    | 42 | b   | 43 | b | 44 | a | 45 | a |
| 46 | c    | 47 | c   | 48 | c | 49 | a | 50 | d |
| 51 | a    | 52 | a   | 53 | a | 54 | d | 55 | d |
| 56 | b    | 57 | b   | 58 | c | 59 | a | 60 | d |
| 61 | b    | 62 | a   | 63 | c | 64 | c | 65 | d |
| 66 | a    | 67 | a   | 68 | b | 69 | c | 70 | d |
| 71 | c    | 72 | a   | 73 | a | 74 | a | 75 | b |
| 76 | c    | 77 | b   | 78 | c | 79 | c | 80 | d |
| 81 | d    | 82 | c   | 83 | c | 84 | c | 85 | d |

## SOLUTIONS

**S.1 (c)**

(a) and (b) are true statements about paging and segmentation. These are reason for which these schemes are invented.

**S.2 (a)**

In **paging and segmentation** the total address space can exceed the size of physical memory.

**S.3 (a)**

In **segmentation** the procedure and data is distinguished and separately protected.

**S.4 (c)**

**Firmware** are coded instructions that are stored permanently in read-only memory (ROM)

**S.5 (a)**

Definitions of contiguous, linked and indexed allocation.

**S.6 (b)**

In virtual memory concept we can execute a program having size larger than physical memory but the program size **cannot exceed** the secondary.

**S.7 (c)**

The fastest techniques for implementing tag comparison is **associative or content addressing**, which permits the input tag to be compared simultaneously to all tags in cache tag memory.

**Associative addressing:** In an associative memory any stored item can be accessed by using the content of the item in question, generally some specified sub field, as an address. Associative memories are also commonly known as **content addressable memories (CAM)**.

**S.8 (c)**

An entire flash memory can be erased in one or a few seconds which is much slower than EPROM.

**S.14 (a)**

Since if RAM is not enough, data has to be stored on disk and hence disk I/O is needed which is wasteful of time.

**S.20 (b)**

A ROM is used to store permanent programs such as the monitor program and other frequently used but fixed programs. A RAM is used as a **scratch-pad memory space** and for storing temporary programs, which user discards after use in current session.

**S.21 (b)**

"**Program Inhibit Mode**" can be used when multiple 2764's are programmed with **different data for the same addresses**. Those 2764's which are not being programmed can be forced into this mode by pulling their CE down.

**S.22 (a)**

A 16-line address bus can access a **64 K memory**. The addresses range from 0000 H through FFFFH, which equals  $(65,536)_{10}$  locations. In this case the memory words are 8-bits wide; however, other systems may have 4 or 16-bits per word  $2^{16} = 64K$ .

**S.24 (a)**

A storage location 2001 H is in **RAM** and each word is 8-bits wide, which is called a **byte**.

**S.25 (c)**

The RAM is said to reside on **page 20H** of memory. The two high-order hexadecimal digits are considered the page number in this system.

**S.27 (all)**

90 – 10 is a heuristic rule that says 90% of the execution time is spent on 10% of the code.

**S.28 (b)**

Memory access time = cache access time  $\times$  hit ratio + (1 – hit ratio)  $\times$  main memory access time  
Let m be the hit-ratio. Then,  $20 = 10 \times m + (1 - m) \times 110$ .

Solving we get  $m = 0.9$ . i.e., 90%.

**S.29 (b)**

Average access time will be  $= (0.8 \times 20) + (1 - 0.8) \times 120 = 40$  ns.

**S.30 (d)**

64 K is  $2^{16}$  bytes. i.e.  $16^4$  bytes i.e., 100000 bytes in hex code.

So last accessible address is  $10000-1 = FFFF$ .

**S.34 (c)**

Average time,

$T_a = T_s + \text{time for half revolution} + \text{time to read a sector}$

$$T_a = T_s + \frac{1}{2R} + \frac{N_s}{N_t} \times \frac{1}{R}$$

**S.35 (b)**

$$\begin{aligned} T_{avg} &= \text{Hit time} + \text{Miss rate} \times \text{Miss penalty} \\ &= 2 + 0.015 \times 60 \\ &= 2.9 \text{ ns} \end{aligned}$$

**S.36 (a)**

Number of bits required = number of characters  $\times$  cell size

$$= 80 \times 50 \times 6 \times 15$$

$$= 360000$$

**S.37 (c)**

$$\begin{aligned} T_{avg} &= \text{Hit time} + \text{Miss rate} \times \text{Miss penalty} \\ 4.5 &= 3 + 0.025 \times \text{Miss penalty} \end{aligned}$$

$$\therefore \text{Miss penalty} = 60 \text{ ns}$$

**S.38 (b)**

Page size = n bytes

$\therefore$  The average amount of space wasted in the last page of a program by internal fragmentation will be  $n/2$ . It means using a small page size to minimize wastage.

**S.39 (c)**

Virtual memory is in fact stored in the secondary storage, which increased memory access time.

Processes can be given larger protected address spaces in virtual memory than is possible by just segmenting real memory.

A linker always assigns address independent of where the programs is loaded in physical memory so, it is not a particular advantage of virtual memory.

**S.40 (b)**

For random access memory, data transfer rate =

$$\frac{1}{\text{cycle time}} = \frac{1}{20\text{ns}} = 0.05 \times 10^9$$

**S.41 (a)**

Request for 541 kbyte is denied because it is larger than both the available memory blocks

**S.42 (b)**

Threads refer to the execution of different processes simultaneously in the CPU. Virtual address space is implemented in memory. The file-system in the way in which data in a disk is organized. A signal is acted upon using the mechanism of interrupt.

**S.43 (b)**

In this case

#### OR ARRAY AND ARRAY

PROM Programmable Fixed

PLA Programmable Programmable

**S.44 (a)**

According to the memory map location 0001H accesses ROM, which outputs the permanently stored data of 00000001.

**S.45 (a)**

It has 16 input data and 16 output data lines.

**S.46 (c)**

$$(8 * 1024) = 2^3 * 2^{10} = 2^{13}$$

Hence it has 13 address lines.

**S.47 (c)**

$$8\text{ KB} * \frac{16}{8} = 16\text{ KB} = 16 * 1024\text{ bytes}$$

**S.48 (c)**

The hit ratio of at least 98% can be shown by following formula:

$$T_c = H T_c + (1-H) T_m$$

where  $T_c$  = Cache access time = 100 ns

$T_m$  = main memory access time = 1200 ns

$H$  = hit ratio expressed as 85...95 etc. and the other parameters as defined in the question.

$$\therefore 100H + 1200(1-H) \leq T_c + 0.20 T_c$$

$$= 100 + 20$$

$$= 120 \text{ ns}$$

(20% more than cache access time)

$$\therefore 100H + 1200 - 1200H \leq 120$$

i.e.  $H \geq 0.98 \Rightarrow 98\%$  or higher is the correct one.

**S.49 (a)**

$$\begin{aligned} \text{CPU time} &= \text{Instruction count} \times \\ &\left[ \text{CPI}_{\text{Execution}} + \frac{\text{memory clock cycle}}{\text{instructions}} \right] \times \text{clock} \\ &\text{cycle time} \end{aligned}$$

$\therefore$  Performance, including cache misses, is of (CPU) with cache time

$$= \text{Instruction count} \times [2 + (1.33 \times 2\% \times 50)] \times \text{clock} \\ \text{cycle time}$$

$$= \text{Instruction count} \times 3.33 \times \text{clock cycle time}$$

The clock cycle time and instruction count are the same, with or without a cache so CPU time increase with CPI from 2.0 for a prefetch cache to 3.33 with a cache that can miss. Hence,

including little memory hierarchy in the CPI calculations increases the CPU time by a factor of 1.67 without any memory hierarchy at the CPI would increase to **(2.0+50×1.33)** or **68.5**.

**S.50 (d)**

$$\text{Mean access time} = c + (1-h) m$$

where,

$c$  = cache access time

$h$  = hit ratio

$m$  = main memory access time

$$\begin{aligned}\therefore \text{Mean access time} &= 20 \text{ ns} + (1-6/7) 28 \text{ ns} \\ &= 20 \text{ ns} + 1/7 \times 28 \text{ ns} \\ &= 24 \text{ ns}\end{aligned}$$

**S.51 (a)**

$$\text{Average Memory access time, } T_{av} = \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}$$

$$T_{av} (\text{1 way}) = 2.0 + (0.014 \times 70) = 2.98 \text{ ns}$$

$$T_{av} (\text{2 way}) = 2.0 \times 1.10 + (0.010 \times 70) = 2.90 \text{ ns}$$

$$\text{CPU time} = IC \times \left( \text{CPI}_{\text{Ex}} + \frac{\text{Misses}}{\text{Instruction}} \times \text{Miss Penalty} \right) \times \text{clock cycle time}$$

$$\begin{aligned}&= IC \times \left[ \text{CPI}_{\text{Ex}} \times \text{clock cycle time} \right. \\ &\quad \left. + \left( \frac{\text{Memory Accesses}}{\text{Instruction}} \times \text{Miss rate} \right. \right. \\ &\quad \left. \left. \times \text{Miss penalty} \times \text{Clock cycle time} \right) \right]\end{aligned}$$

Substituting 70 ns for  $(\text{Miss Penalty} \times \text{Clock cycle time})$ , we get

$$\begin{aligned}\text{CPU time (1-way)} &= IC \times \{2.0 \times 2 + (1.3 \\ &\quad \times 0.014 \times 70)\} \\ &= 5.27 \times IC\end{aligned}$$

$$\begin{aligned}\text{CPU time (2-way)} &= IC \times \{2.0 \times 2 \times 1.10 \\ &\quad + (1.31 \times 0.010 \times 70)\} \\ &= 5.31 \times IC\end{aligned}$$

$$\therefore \text{Relative Performance} = \frac{\text{CPU time}_{\text{2 way}}}{\text{CPU time}_{\text{1 way}}}$$

$$= \frac{5.31 \times IC}{5.27 \times IC}$$

$$= \frac{5.31}{5.27} = 1.01$$

**S.52 (a)**

DRAM Stores data as a series of electron charges in individual cells. This data must be constantly recharged or ‘refreshed’ to keep the data from dissipating.

**S.54 (d)**

There are 64 words in a block

$$\text{Hence, } 4 \text{ K cache has } \frac{4 \times 1024}{64} = 64 \text{ blocks}$$

Since there are 4 blocks per set.

$$\text{so, } 64 \text{ blocks form } \frac{64}{4} = 16 \text{ set}$$

$$\text{Number of bits in the set} = 16 = 2^4 = 4 \text{ bits}$$

$$\text{Number of bits in word} = 64 = 2^6 = 6 \text{ bits}$$

**S.55 (d)**

We have to find size of ROM, for multiplexing 8-bit unsigned integer no.

$$\begin{aligned}\therefore \text{size of ROM} &= (2^8 \times 2^8) \times (8 + 8) \\ &= 2^{16} \times 16 \\ &= 64K \times 16\end{aligned}$$

**S.56 (b)**

Since the cache has  $2c$  blocks and there are 2 blocks per set, the cache thus has  $c$  sets. So, block  $k$  of the main memory would map to block  $(k \bmod c)$  of the cache.

**S.57 (b)**

More than one words are put in one cache block to exploit the spatial locality of reference in a program.

**S.58 (c)**

Instruction opcode is not a form of memory.

**S.59 (a)**

$$\begin{aligned}\text{Data transfer rate} &= 10 \text{ M bytes/sec} \\ &= 10 \times 2^{10} \text{ K byte/sec}\end{aligned}$$

$$\text{Size of transfer} = 20 \text{ Kbytes}$$

$$10 \times 2^{10} \times x\% = 20$$

$$10 \times 2^{10} \times x/100 = 20$$

$$x = \frac{20 \times 100}{10 \times 2^{10}}$$

$$x = \frac{200}{1024}$$

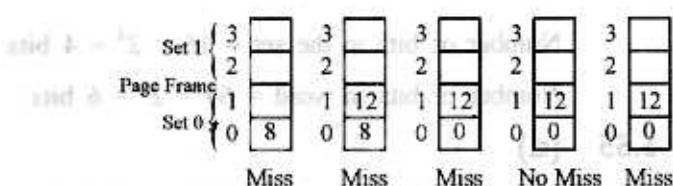
$$x = 0.19\%$$

$$\text{or } x = 0.10\%$$

**S.60 (d)**

Sequence 8, 12, 0, 12, 8

Apply the LRU as follows



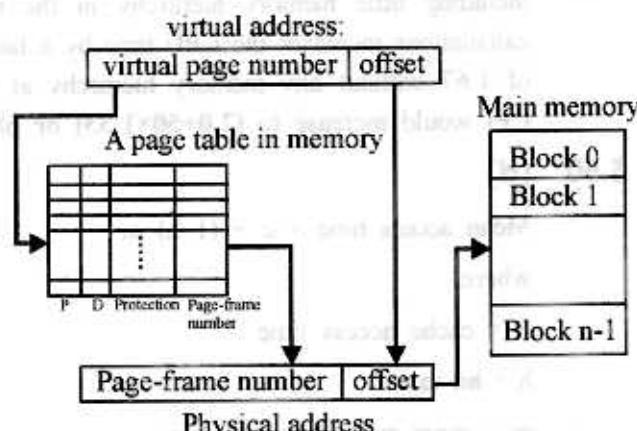
$$\text{Total number of miss} = 4$$

**S.61 (b)**

Minimum size of ROM required to store the complete truth table of 8 bit  $\times$  8 bit multiplex is  $64 \text{ K} \times 16 \text{ bit}$ , because it has 16 address lines and 16 data lines.

**S.62 (a)**

Mapping of virtual and physical address is given below.



Given that pages are of 4 KB in size, So

$$\log_2 4 \times 2^{10} \times 2^3 = 15$$

15 bits are required for offset field of both the virtual and physical address.

For virtual address: virtual page number =  $32 - 15 = 17$  bit required

For physical address: physical page number field =  $30 - 15 = 15$  bit required.

So maximum number of bits that can be used for storing protection and other information is

$$17 - 15 = 2 \text{ bits.}$$

**S.63 (c)**

Memory cycle time = 64 nsec.

In 1 ns refresh cycle arc = 100

$$\begin{aligned}\text{In } 64 \text{ ns refresh cycle arc} &= \frac{100}{10^{-3}} \times 64 \times 10^{-9} \\ &= 64 \times 10^{-4}\end{aligned}$$

1 refresh cycle takes = 100 ns

$$\begin{aligned}64 \times 10^{-4} \text{ refresh cycle takes} &= 64 \times 10^{-4} \times 100 \text{ ns} \\ &= 64 \times 10^{-2} \text{ ns}\end{aligned}$$

So percentage refresh cycle time to memory cycle time is

$$= \frac{64 \times 10^{-2} \text{ ns}}{64 \text{ ns}} \times 100 = 1\%$$

## S.64 (c)

| File size | 100 bytes block | Block needed for information | Byte for information |
|-----------|-----------------|------------------------------|----------------------|
| 11050     | 111             | 444                          | 5                    |
| 4990      | 50              | 200                          | 2                    |
| 5170      | 52              | 208                          | 3                    |
| 12640     | 127             | 508                          | 6                    |
| 33850     | 340             |                              | 16                   |

In case I, i.e. 100 byte block,

$$\begin{aligned} \text{total space required for storing 100 byte disk blocks} &= 340 \times 100 + 16 \times 100 \\ &= 35,600 \text{ bytes} \end{aligned}$$

| File size | 200 bytes blocks | Block needed for information | Byte for information |
|-----------|------------------|------------------------------|----------------------|
| 11050     | 56               | 224                          | 2                    |
| 4990      | 25               | 100                          | 1                    |
| 5170      | 26               | 104                          | 1                    |
| 12640     | 64               | 256                          | 2                    |
| 33850     | 171              |                              | 6                    |

In case II, i.e. 200 byte block,

$$\begin{aligned} \text{total space required for storing 200 byte disk blocks} &= 171 \times 200 + 6 \times 200 \\ &= 35,400 \text{ bytes.} \end{aligned}$$

## S.65 (d)

Increasing the RAM means increase the primary memory which reduce the swapping so fewer page faults occur.

## S.66 (a)

$$\text{Revolution per minute} = 3000$$

$$\text{Or } 3000/60 = 50 \text{ RPS}$$

$$\begin{aligned} \text{In one round it can read} &= 512 \text{ KB} = 2^{19} \text{ byte} \\ \text{for writing it reads} &= 4 \text{ bytes} = 2^2 \text{ byte} \end{aligned}$$

$$\begin{aligned} \text{No. of track read per second} &= \frac{2^{19}}{2^2} \times 50 \\ &= 2^{17} \times 50 \\ &= 6553600 \text{ C.P.U.} \end{aligned}$$

Each interruption take = 40 nsec.

So, 6553600 interrupt take = 0.262144 sec.

$$\text{So, Percentage} = \frac{0.262144}{1} \times 100$$

$$= 26.2144\% \text{ (approx)}$$

In case, we use 1KB = 1000B, then answer is 25%.

## S.67 (a)

Tag Set Block

|    |   |   |
|----|---|---|
| 18 | 9 | 5 |
|----|---|---|

$$\begin{aligned} \text{So } h_1 &= 18/10 + 0.6 \text{ ns} \\ &= 1.8 + 0.6 \text{ ns} \\ &= 2.4 \text{ ns} \end{aligned}$$

## S.68 (b)

Tag Set Block

|    |    |   |
|----|----|---|
| 17 | 10 | 5 |
|----|----|---|

$$\begin{aligned} \text{So } h_2 &= 17/10 + 0.6 \text{ ns} \\ &= 1.7 + 0.6 \text{ ns} \\ &= 2.3 \text{ ns} \end{aligned}$$

## S.69 (c)

$$\text{No. of surfaces} = 16$$

$$\text{No. of tracks/surface} = 128$$

$$\text{No. of sector/track} = 256$$

$$\text{Total size of disk} = 16 \times 128 \times 256 \times 512 \text{ bytes}$$

$$= 2^4 \times 2^7 \times 2^8 \times 2^9 \text{ bytes}$$

$$= 2^8 \times 2^{20} \text{ byte}$$

$$= 2^8 \text{ Mega byte}$$

$$= 256 \text{ MB}$$

## Total Number of Sector in the disk

$$= 16 \times 128 \times 256 \text{ byte}$$

$$= 2^4 \times 2^7 \times 2^8 \text{ byte}$$

$$= 2^{19} \text{ byte}$$

So 19 bits are needed.

## S.70 (d)

| Address of location assuming byte addressability | Address of location assuming word addressability | Contents of memory location | Contents of location, mnemonic | Ins size (no of words) |
|--------------------------------------------------|--------------------------------------------------|-----------------------------|--------------------------------|------------------------|
| 1000                                             | 1000                                             | MOV R1,<br>(3000)           | R1 $\leftarrow M[3000]$        | 2                      |
| 1008                                             | 1002                                             | MOV R2, (R3)                | R2 $\leftarrow M[R3]$          | 1                      |
| 1012                                             | 1003                                             | ADD R2, R1                  | R2 $\leftarrow R1 + R2$        | 1                      |
| 1016                                             | 1004                                             | MOV (R3), R2                | M[R3] $\leftarrow R2$          | 1                      |
| 1020                                             | 1005                                             | INC R3                      | R3 $\leftarrow R3 + 1$         | 1                      |
| 1024                                             | 1006                                             | DEC R1                      | R1 $\leftarrow R1 - 1$         | 1                      |
| 1028                                             | 1007                                             | BNZ LOOP                    | Branch on not zero             | 2                      |
| 1036                                             | 1009                                             | HALT                        | stop                           | 1                      |
| -----                                            | -----                                            |                             |                                |                        |
| -----                                            | -----                                            |                             |                                |                        |
| 2000 To 2010                                     | 100                                              |                             |                                |                        |
| -----                                            | -----                                            |                             |                                |                        |
| -----                                            | -----                                            |                             |                                |                        |
|                                                  | 3000                                             | 10                          |                                |                        |

The only place we store into the memory is from R2 to the contents of R3. R3 indexes the memory starting from 2000 and moving onto 2010. In the first pass through the loop we store 110 into location 2000. In the next pass R3 has the value 2001 and we store 110 into this location. In the tenth pass through the loop R3 has the value of 2009 and we store 110 into 2009. **2010 is not affected at all and so it remains 100 at the end of the loop.**

## S.71 (c)

Refer Table

From the memory map we see that the address through byte addressable memory, where the interrupt occurs is 1020, **so the address stored on the stack is 1024.**

## S.72 (a)

Given  $M[3000] = 10$

| Instruction              | Required Memory Reference | Total |
|--------------------------|---------------------------|-------|
| $R_1 \leftarrow M[3000]$ | 1                         |       |
| $R_2 \leftarrow M[R_3]$  | 10                        |       |
| $M[R_3] \leftarrow R_2$  | 10                        |       |
|                          | Total = 21                |       |

## S.73 (a)

If there are 128 line or  $2^7$  then it require 7 bits.

Size of line 64 words =  $2^6$

So 6 bits are needed to identify cache word uniquely

Tag bit =  $20 - 6 - 7 = 7$  bits

## S.76 (c)

In instruction execution pipeline, the earliest that the data TLB can be accessed **during effective address calculation** has started.

## S.77 (b)

In magnetic disk once head is in position, the desired cell may be in wrong part due to a high capacity of tracks. Some time is required for this cell to reach the read-write head so that data transfer can begin. The average time for this

movement to take place is seek latency which is not linearly proportional to the seek distance because a higher capacity of tracks on the periphery of the platter.

**S.78 (c)**

| Tag | Set | Block |
|-----|-----|-------|
| 17  | 11  | 4     |

$$\begin{aligned}\text{Total size of tags} &= 17 \times 2 \times 1024 \text{ bits} \\ &= 34 \text{ kbits}\end{aligned}$$

**S.79 (c)**

Because of page size is 4 KB. 1 row contain 1024 elements means  $2^{10}$  location. So ARR [0] [0] is same as ARR [4][0] which is same location in next page.

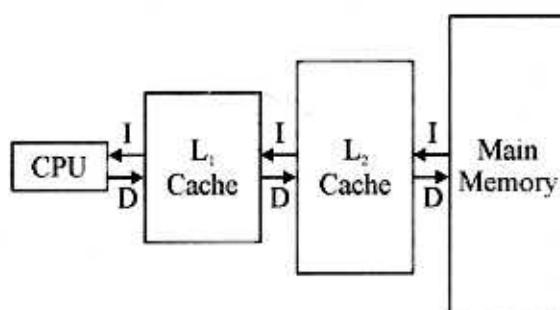
**S.80 (b)**

The Hit ratio for the given loop is:

$$= \frac{1024}{1024+1024} = \frac{1024}{2048} = \frac{1}{2} = 50\%$$

**S.81 (d)**

Consider the following diagram



So L<sub>2</sub> Cache must be at least as large as the L<sub>1</sub>.

**S.82 (c)**

$$\text{Basic RAM} = 32 \text{ K} \times 1$$

Design RAM = 256 K × 8 (Total memory capacity available)

$$\text{No. of RAM} = \frac{256 \text{ K} \times 8}{32 \text{ K} \times 1} = \frac{2^{18} \times 2^3}{2^{15} \times 2^0} = 64$$

It requires 8 parallel line and in each parallel line 8 serial RAM chips are required.

**S.83 (c)**

The address (400, 16, 29) corresponds to sector no.

$$400 \times 2 \times 10 \times 63 + 16 \times 63 + 29 = 505037$$

**S.84 (c)**

The address (0, 16, 31) corresponds to sector no.

$$16 \times 63 + 31 = 1039$$

**S.85 (d)**

There are total 4 sets in the cache and each set contains 4 blocks.

|       |     |     |
|-------|-----|-----|
|       | 8   | 48  |
| Set 0 | X   | 32  |
|       | 8   |     |
|       | 216 | 92  |
|       | 1   |     |
| Set 1 | 133 |     |
|       | 73  |     |
|       | 129 |     |
| Set 2 |     |     |
|       |     |     |
|       |     |     |
| Set 3 | 255 | 155 |
|       | 3   |     |
|       | 159 |     |
|       | 63  |     |

$$0 \bmod 4 = 0 \quad \text{set 0}$$

$$255 \bmod 4 = 3 \quad \text{set 3}$$

$$1 \bmod 4 = 1 \quad \text{set 1}$$

$$4 \bmod 4 = 0 \quad \text{set 0}$$

$$3 \bmod 4 = 3 \quad \text{set 3}$$

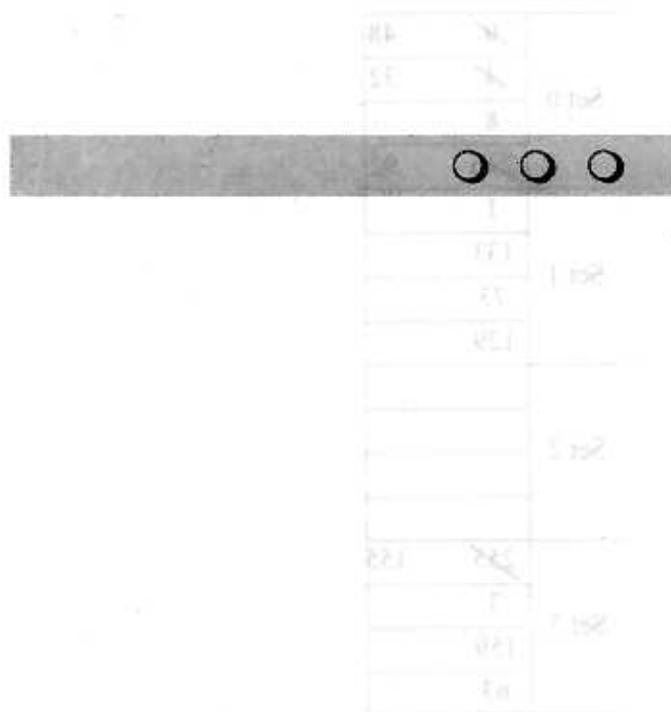
$$8 \bmod 4 = 0 \quad \text{set 0}$$

$$133 \bmod 4 = 1 \quad \text{set 1}$$

## 3.3.20

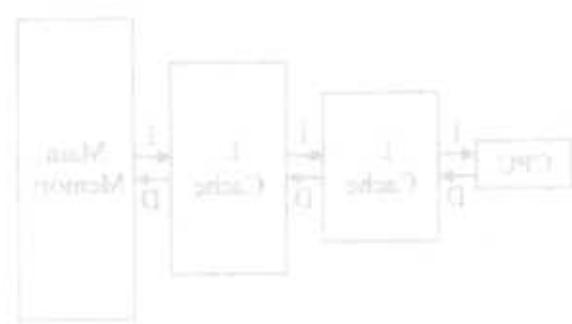
## COMPUTER ORGANIZATION AND ARCHITECTURE

159 mod 4 = 3    set 3  
 216 mod 4 = 0    set 0  
 129 mod 4 = 1    set 1  
 63 mod 4 = 3    set 3      (c) 88.2  
 8 mod 4 = 0    set 0 (already in cache)  
 48 mod 4 = 0    set 0 (48 will replace block 0  
 using LRU)      (d) 88.2  
 32 mod 4 = 0    set 0 (32 will replace block  
 4)      (e) 88.2  
 73 mod 4 = 1    set 1  
 92 mod 4 = 0    set 0 (92 will replace block  
 216)      (b) 88.2  
 155 mod 4 = 3    set 3 (155 will replace 255)  
 ∴ 216 will not be cache as it has been  
 replaced by 92.



- 0. 159    0 → 159 from 0
- 1. 216    1 → 216 from 0
- 2. 129    1 → 129 from 1
- 3. 63    0 → 63 from 1
- 4. 8    0 → 8 from 2
- 5. 48    1 → 48 from 2
- 6. 32    0 → 32 from 2
- 7. 73    1 → 73 from 3

a cache must have enough slots to accommodate identical data items in consecutive memory locations so that no object is evicted before a previous object is evicted.  
 (c) 88.2  
 (d) 88.2  
 (e) 88.2



- set 0 is assigned to read from address 0 to 3
- 1. 159 → 159 from 0
- 2. 216 → 216 from 0
- 3. 129 → 129 from 1
- 4. 63 → 63 from 1
- 5. 8 → 8 from 2
- 6. 48 → 48 from 2
- 7. 32 → 32 from 2
- 8. 73 → 73 from 3

# 3.4

## INPUT OUTPUT

### LEVEL-1

- Q.1** Which of the following statement is incorrect?
- The advantage of EPROM chip is that their contents can be erased and reprogrammed and it required dissipating the charges trapped in the transistors of memory cells.
  - The disadvantage of EEPROM is that different voltages are needed for erasing, writing and reading the stored data.
- (i)
  - (ii)
  - (i) and (ii)
  - Neither (i) nor (ii)
- Q.2** System calls are usually invoked by using
- an indirect jump
  - a software interrupt
  - polling
  - a privileged instruction.
- (ii), (iii)
  - (i), (ii)
  - (i), (ii), (iii), (iv)
  - (iii)
- Q.3** How many units in a single bus structure communicate at a time?
- one
  - two
  - three
  - four

- Q.4** A single bus structure is primarily found in
- main frames
  - super computer
  - high performance computers
  - mini and micro computers
- Q.5** Computer peripheral is
- a computer device which is not connected to CPU
  - a device which is connected to the CPU
  - a device for manually operating the computer
  - none of these
- Q.6** The ability of a medium sized computer system to increase data processing capability by addition of devices such as mass storage device, I/O devices etc. is called
- computer expendability
  - computer mobility
  - computer enhancement
  - computer upward capability
- Q.7** The bus connected between the CPU and main memory that permits transfer of information between main memory and the CPU is called
- DMS bus
  - memory bus
  - address bus
  - control bus

- Q.8** An I/O processor controls the flow of information between \_\_\_\_\_.
- cache memory and I/O devices
  - main memory and I/O devices
  - two I/O devices
  - cache and main memories
- Q.9** In an 8085 microprocessor system with memory mapped I/O.
- I/O devices have 16-bit addresses
  - I/O devices are accessed using IN and OUT instructions
  - there can be a maximum of 256 input devices and 256 output devices
  - arithmetic and logic operations can be directly performed with the I/O data.
- Q.10** The following are the disadvantages of \_\_\_\_\_.
- The I/O transfer rate is limited by the speed with which the processor can test and service a device.
  - The processor is tied up in managing an I/O transfer, a number of instruction must be executed for each I/O transfer.
- Programmed I/O
  - Interrupt driven I/O
  - Both (a) and (b)
  - Neither (a) nor (b)
- Q.11** In the programmed I/O method, the CPU stays in a program loop to \_\_\_\_\_.
- indicate that it is ready for data transfer
  - indicate that it completes data transfer
  - indicate that it is not ready for data transfer
  - none of these
- Q.12** In \_\_\_\_\_, the I/O device does not have direct access to memory.
- programmed I/O
  - interrupted initiated I/O
  - DMA
  - None of these
- Q.13** Pipelined instruction execution is the key element in the implementation of \_\_\_\_\_ architecture.
- CISC
  - RISC
  - SISD
  - None of these
- Q.14** The basic aim of multiprocessor system is to improve \_\_\_\_\_.
- Throughput
  - Reliability
  - Flexibility
  - Availability
- (i), (iii)
  - (ii), (iv)
  - (i), (ii), (iii), (iv)
  - (ii), (iii), (iv)
- Q.15** Which of the following statement is correct?
- Isolated I/O uses a different set of instruction to read/write to memory than to read/write to I/O devices.
  - Memory mapped I/O uses the same instruction set but different addresses for these operations.
- (i)
  - (ii)
  - (i) and (ii)
  - Neither (i) nor (ii)
- Q.16** Which of the following is incorrect?
- In the programmed I/O method, the CPU waits for the I/O devices.
  - In the interrupt driven I/O device informs the CPU of its ready status via an interrupt.
  - In DMA, the CPU sends its I/O to the DMA controller which manages the entire transaction.
- (i), (ii)
  - (ii), (iii)
  - (i), (ii), (iii)
  - None of these

- Q.17** An interrupt in which the external device supplies its address as well as the interrupt requests is called  
 (a) designated interrupt  
 (b) non maskable interrupt  
 (c) vectored interrupt  
 (d) maskable interrupt
- Q.18** Which of the following architecture is/are suitable for realizing SIMD?  
 (a) Vector processor  
 (b) Von Neumann  
 (c) Array processor  
 (d) None of these
- Q.19** Which of the following interrupt is both level and edge sensitive?  
 (a) TRAP  
 (b) RST 7.5  
 (c) RST 5.5  
 (d) INTER
- Q.20** Bus Arbitration is  
 (a) deciding the controller of the bus  
 (b) latching information in the bus  
 (c) clearing the bus  
 (d) none of the above
- Q.21** When INTR is encountered, the processor branches to the memory location, which is  
 (a) 0024H  
 (b) determined by the 'call address' instruction issued by the I/O device  
 (c) determined by the 'RST n' instruction issued by the I/O device  
 (d) All of the above
- Q.22** If SUB A, B means  $B - A$ , then SUB 4(R0), \*5(R1) means  $((X))$  means content of register or memory location X)  
 (a)  $((R1) + 5) - (4 * (R0))$   
 (b)  $((R1) + 5) - ((R0) + 4)$   
 (c)  $((R1) + 5) - (4 * (R0))$   
 (d)  $((R1) + 4) - (R0 + 4)$
- Q.23** How many characters per sec (7 bits + parity) can be transmitted over a 2400-bps line in asynchronous mode?  
 (a) 300  
 (b) 240  
 (c) 320  
 (d) None of the above
- Q.24** Word 20 contains 40  
 Word 30 contains 50  
 Word 40 contains 60  
 Word 50 contains 70
- Which of the following instructions loads 60 into the accumulator?  
 (a) load immediate 60  
 (b) load direct 30  
 (c) load indirect 20  
 (d) load indirect 30
- Q.25** Pseudo-instructions are  
 (a) assembler directives  
 (b) instructions in any program that have no corresponding machine code instruction  
 (c) instructions in any program whose presence or absence will not change the output for any input  
 (d) None of the above
- Q.26** Which of the following instructions will not be there in a memory-mapped I/O system?  
 (a) LDA  
 (b) IN  
 (c) ADD  
 (d) OUT
- Q.27** The first operation performed in INTEL 8085 after RESET is  
 (a) instruction fetch from location 0000H  
 (b) memory read from location 0000H  
 (c) instruction fetch from location 8000H  
 (d) stack initialization

**Q.28** Consider the following four instructions.

- (1) PUSH PSW
- (2) CALL ADDR
- (3) XTHL
- (4) RST n

The stack pointer will be affected by the instruction(s)

- (a) 1 only
- (b) 1 and 2 only
- (c) 1,2 and 4 only
- (d) 1,2 and 3 only

**Q.29** In order to complement the lower order nibble of the accumulator, one can use

- (a) ANI 0FH
- (b) XRI 0FH
- (c) ORI 0FH
- (d) CMA

**Q.30** Which one of the following interrupts is non-maskable?

- (a) TRAP
- (b) RST 7.5
- (c) INTR
- (d) RST 6.5

**Q.31** The contents of the A15-A8 (higher order address lines) while executing "IN addr" instruction are

- (a) same as the contents of A7-A0
- (b) irrelevant
- (c) all bits reset (i.e. 00H)
- (d) all bits set (i.e. FFH)

**Q.32** Which one of the following instructions may be used to clear the accumulator content (i.e. A = 00H) irrespective of its initial value?

- (a) CLR A
- (b) ORA A
- (c) SUB A
- (d) MOV A, 00H

**Q.33** The instruction used to shift right the accumulator contents by one bit through the carry flag bit is

- (a) RLC
- (b) RAL
- (c) RRC
- (d) RAR

**Q.34** A sequence of two instructions that multiplies the contents of the DE register pair by 2 and stores the result in the HL register pair (in 8085 assembly language) is

- (a) XCHG and DAD B
- (b) XTHL and DAD H
- (c) PCHL and DAD D
- (d) XCHG and DAD H

**Q.35** Determine the number of clock cycles that it takes to process 200 tasks in a six-segment pipeline.

- (a) 205
- (b) 200
- (c) 195
- (d) none of the above

## LEVEL-2

**Q.36** SIMD machines with word slice machines uses \_\_\_\_\_ as memory where as bit slices machines uses \_\_\_\_\_ as the memory module.

- (a) RAM, associative memory
- (b) Associative memory, RAM
- (c) RAM, RAM
- (d) Associative memory, associative memory

**Q.37** An asynchronous serial communication controller that uses a start-stop scheme for controlling the serial I/O of a system is programmed for a string of length 7 bit, one parity bit (odd parity) and one stop bit. The transmission rate is 1000 bits/second. Then how many strings can be transmitted per second?

- (a) 100
- (b) 50
- (c) 200
- (d) 150

**Q.38** Consider a CRT display that has next mode display format of  $75 \times 30$  characters with a 9  $\times$  12 characters cell. What is the video buffer RAM for the display to be used in monochrome (1-bit per pixel) graphics mode?

- (a) 365000
- (b) 545000
- (c) 412500
- (d) 243000

**Q.39** A CPU respond to an interrupt

- (a) CPU halts execution of program
- (b) Active signal on any interrupt lines can interrupt CPU
- (c) Active signal on any interrupt lines can't interrupt CPU
- (d) None of the above

**Q.40** The seek time of a disk is 30 ms. It rotates at the rate of 30 rotations per second. Each track has a capacity of 300 words. The access time is approximately

- (a) 47 ms
- (b) 50 ms
- (c) 60 ms
- (d) 62 ms

**Q.41** Assume that a slow memory device is interfaced with an 8085 microprocessor and a 1-wait state generating circuit is connected to READY input. Then, the execution time needed for the following program would be,

|      |      |
|------|------|
| LDA  | TEMP |
| ADD  | B    |
| LHLD | TEMP |

- (a) 43 T-states
- (b) 30 T-states
- (c) 40 T-states
- (d) 33 T-states

**Q.42** A DMA controller transfers 16-bit words to memory using cycle stealing. The words are assembled from a device that transmits characters at a rate of 2400 characters per second. The CPU is fetching and executing instructions at an average rate of 1 million instructions per second. By how much will the CPU be slowed down because of the DMA transfer?

- (a) 1.2%
- (b) 0.12%
- (c) 0.0127%
- (d) 12%

## GATE QUESTIONS

**Q.43** The data transfer rate of a double-density floppy disk system is about [GATE 1987]

- (a) 5 Kbits/sec
- (b) 50 Kbits/sec
- (c) 500 Kbits/sec
- (d) 5000 Kbits/sec

**Q.44** Start and stop bits do not contains an 'information' but are used in serial communication for [GATE 1992]

- (a) Error detection
- (b) Error correction
- (c) Synchronization
- (d) Slowing down the communication

**Q.45** For the daisy chain scheme of connecting I/O devices, which of the following statements is true? [GATE 1996]

- (a) It gives non-uniform priority to various devices.
- (b) It gives uniform priority to all devices
- (c) It is only useful for connecting slow devices to a processor device
- (d) It requires a separate interrupt pin on the processor for each device.

**Q.46** I/O redirection

[GATE 1997]

[1-Mark]

- (a) implies changing the name of a file
- (b) can be employed to use an existing file as input file for a program
- (c) implies connection of 2 programs through a pipe
- (d) none of the above

**Q.47** In serial communication employing 8 data bits, a parity bit and 2 stop bits, the minimum baud rate required to sustain a transfer rate of 300 characters per second is [GATE 1998]

[1-Mark]

- (a) 2400 baud
- (b) 19200 baud
- (c) 4800 baud
- (d) 1200 baud

**Q.48** In serial data transmission, every byte of data is padded with a '0' in the beginning and one or two '1's at the end of byte because

[GATE 2002]

[1-Mark]

- (a) Receiver is to be synchronized for byte reception
- (b) Receiver recovers lost '0's and '1's from these padded bits
- (c) Padded bits are useful in parity computation
- (d) None of the above

**Q.49** What is the bit rate of a video terminal unit with 80 characters/line, 8 bits/character and horizontal sweep time of 100  $\mu$ s (including 20  $\mu$ s of retrace time)?

[IT-GATE 2004]

[1 Mark]

- (a) 8 Mbps
- (b) 6.4 Mbps
- (c) 0.8 Mbps
- (d) 0.64 Mbps

**Q.50** We have two designs D1 and D2 for a synchronous pipeline processor. D1 has 5 pipelines stages with execution times of 3 nsec, 2 nsec, 4 nsec, 2 nsec and 3 nsec while the design D2 has 8 pipeline stages each with 2 nsec execution time. How much time can be saved using design D2 over design D1 for executing 100 instructions?

[IT-GATE 2005]

[2-Marks]

- (a) 214 nsec
- (b) 202 nsec
- (c) 86 nsec
- (d) -200 nsec

**Q.51** Consider a 2-way set associative cache memory with 4 sets and total 8 cache blocks (0-7) and a main memory with 128 blocks (0-127). What memory blocks will be present in the cache after the following sequence of memory block references, if LRU policy is used for cache block replacement? Assuming that initially the cache did not have any memory block from the current job?

0 5 3 9 7 0 16 55

[IT-GATE 2005]

[2-Marks]

- (a) 0 3 5 7 16 55
- (b) 0 3 5 7 9 16 55
- (c) 0 5 7 9 16 55
- (d) 3 5 7 9 16 55

**Q.52** Normally user programs are prevented from handling I/O directly by I/O instructions in them. For CPUs having explicit I/O instructions, such I/O protection is ensured by having the I/O instructions privileged. In a CPU with memory mapped I/O, there is no explicit I/O instruction. Which one of the following is true for a CPU with memory mapped I/O? [GATE 2005] [1-Mark]

- (a) I/O protection is ensured by a hardware trap
- (b) I/O protection is ensured during system configuration
- (c) I/O protection is not possible
- (d) I/O protection is ensured by operating system routine(s)

**Q.53** Which one of the following is true for a CPU having a single interrupt request line and a single interrupt grant line? [GATE 2005]

[1-Mark]

- (a) Neither vectored interrupt nor multiple interrupting devices are possible
- (b) Vectored interrupts and multiple interrupting devices are both possible
- (c) Vectored interrupt is possible but multiple interrupting devices are not possible
- (d) Vectored interrupts are not possible but multiple interrupting devices are possible

**Q.54** What is the swap space in the disk used for?

[GATE 2005]

[1-Mark]

- (a) Saving temporary html pages

- (b) Storing the super-block

- (c) Saving process data

- (d) Storing device drivers

**Q.55** A CPU generally handles an interrupt by executing an interrupt service routine

[GATE 2009]

[1-Mark]

- (a) as soon as an interrupt is raised

- (b) by checking the interrupt register at the end of fetch cycle

- (c) by checking the interrupt register after finishing the execution of the current instruction

- (d) by checking the interrupt register at fixed time intervals

## ANSWER KEY

|    |      |    |   |    |   |    |      |    |   |
|----|------|----|---|----|---|----|------|----|---|
| 1  | d    | 2  | a | 3  | b | 4  | d    | 5  | b |
| 6  | a    | 7  | b | 8  | b | 9  | a    | 10 | c |
| 11 | a    | 12 | a | 13 | b | 14 | c    | 15 | c |
| 16 | d    | 17 | c | 18 | b | 19 | a    | 20 | a |
| 21 | b, c | 22 | b | 23 | b | 24 | a, c | 25 | a |
| 26 | b, d | 27 | a | 28 | c | 29 | b    | 30 | a |
| 31 | a    | 32 | c | 33 | d | 34 | d    | 35 | a |
| 36 | a    | 37 | a | 38 | d | 39 | a, b | 40 | a |
| 41 | a    | 42 | b | 43 | a | 44 | c    | 45 | a |
| 46 | b    | 47 | a | 48 | a | 49 | b    | 50 | b |
| 51 | c    | 52 | d | 53 | d | 54 | c    | 55 | b |

## SOLUTIONS

**S.1 (d)**

(i) and (iii) are correct statements about EPROM and EEPROM.

**S.2 (a)**

System calls usually invoked by using (ii) and (iii).

**S.9 (a)**

Memory mapped I/O

1. I/O devices have 16 bit addresses
2. Device are accessed by registers
3. Max. devices =  $2^{16}$

**S.10 (c)**

(i) and (ii) are the disadvantages of programmed I/O and interrupt driven I/O.

**S.11 (a)**

In the programmed I/O method, the CPU stays in a program loop to indicate that it is ready for transfer.

**S.13 (b)**

An important feature that is common to most RISC designs is pipelined instruction execution. Pipelining is used in many high performance computers, but it is a key element in the implementation of the RISC architecture.

**S.14 (c)**

The basic aim of multiprocessor system to improve throughput, reliability, flexibility, and availability.

**S.15 (c)**

(i) and (ii) are correct about Isolated I/O and memory mapped I/O.

**S.16 (d)**

(i), (ii), (iii) are correct about programmed I/O, interrupt driven I/O and DMA.

**S.23 (b)**

7 databits + 1 Parity bit + 2 stop bit

So character / sec

$$= \frac{\text{Total capacity/sec}}{\text{Total bits}}$$

$$= \frac{2400}{10} \\ = 240$$

**S.24 (a, c)**

Immediate mode instruction contains the operand = 60

Indirect mode contains the address of EA that is 20, which contains 40, having operand = 60

**S.34 (d)**

XCHG exchange H & L with D & E

DAD H add register pair to H & L register.

**S.35 (a)**

k = 6 segment

n = 200 tasks

Number of clock cycle = (k + n - 1)

$$= 6 + 200 - 1$$

$$= 205 \text{ cycles}$$

**S.36 (a)**

SIMD machines are divided into

- i. word slice machine
- ii. Bit slice machines

Word slice machines uses RAM as a memory and bits slice machines uses associative memory as the memory module.

**S.37 (a)**

Complete bit stream that is transmitted for the '0110101' is 1011010101 as per requirement. (1 start bit + 7 bit string + 1 parity bit + 1 stop bit)

The number of strings that can be transmitted =  $1000/10 = 100$ .

**S.38 (d)**

Number of bits required = No of characters  $\times$  cell size

$$= 75 \times 30 \times 9 \times 12$$

$$= 243000 \text{ bytes}$$

= size of RAM.

**S.39 (a, b)**

An active signal on any interrupt lines can interrupt the CPU while it is executing a program. When an interrupt occurs the CPU halts the execution of the program temporarily and runs another program demanded by the interrupt on priority basis. Then it returns back to the initial program.

**S.40 (a)**

Access time is seek time plus latency time. Seek time is the time taken by the read-write head (i.e., RWH) to get into the right track. Latency time is the time taken by the RWH, to position itself in the right sector, so that actual transfer can take place. Here, a track has 300 words. So, on an average to position at the right word the RWH should traverse 150 words. Time taken for this will be  $150/(30 \times 300)$  sec. = 16.66 ms (approximately). So, the access time will be  $30 + 16.66 = 46.66$  ms  $\approx 47$  ms

**S.42 (b)**

CPU refers to memory on the average once (or more) every 1  $\mu$ sec ( $10^{-6}$  sec).

Characters arrive on every  $1/2400 = 416.6$   $\mu$ sec. Two characters of 8 bits each are packed into a 16-bit word every  $2 \times 416.6 = 833.3$   $\mu$ sec.

The CPU is slowed down by no more than  $(1/833.3) \times 100 = 0.12\%$ .

**S.43 (a)****Double Density FDD**

RX 01 & RX 0.2, are double Density drives available with 10k bits/sec and 5 kbytes/sec interleaved.

**S.44 (c)**

Start and stop bits do not contain an information but are used in serial communication for synchronization.

**S.45 (a)**

For the daisy chain scheme of connecting I/O devices, it gives non uniform priority to various devices.

**S.46 (b)**

I/O redirection can be employed to use an existing file as input file for a program.

**S.47 (a)**

8 data bits + 2 stop bits but, baud rate is due to only 8 data bits with transfer rate of 300 characters per second.

$\therefore$  We require baud rate of 2400.

**S.48 (a)**

In serial data transmission, every byte of data is padded with a '0' in the beginning and one or two '1's at the end of byte because receiver is to be synchronized for byte reception.

**S.49 (b)**

Given:

$$\text{No. of character per line} = 80$$

$$\text{No. of bits per character} = 8$$

$$\text{Horizontal Sweep Time (HST)} = 100 \mu\text{s}$$

$$\text{Retrace time} = 20 \mu\text{s}$$

$$\text{Bit rate} = \frac{\text{No. of bits per line}}{\text{HST}}$$

$$= \frac{8 \times 80 \times 10^6}{100}$$

$$= 6.4 \text{ Mbps}$$

**S.50 (b)**

Design D1: 5 stages with execution time 3ns, 2ns, 4ns, 2ns, 3ns.

There is bottleneck execution time of 4ns.

so total time required for 100 instructions is

$$T = (m + n - 1)t_p$$

$$T_1 = (5 + 100 - 1) \times 4 = 416 \text{ ns}$$

Design D2: 8 stages each with 2ns execution time.

$$T_2 = (8 + 100 - 1) \times 2 = 214 \text{ ns}$$

so Time saved by design D2 is  $416 - 214 = 202$  ns.

**S.51 (c)**

2-way set associative each memory with 4-sets then LRU work as

| Set 0 | Set 1 | Set 2 | Set 3 |  | Set 0 | Set 1 | Set 2 | Set 3 |
|-------|-------|-------|-------|--|-------|-------|-------|-------|
| 0     | 5     |       | 3     |  | 0     | 5     |       | 55    |
| 16    | 9     |       | 7     |  | 16    | 9     |       | 7     |

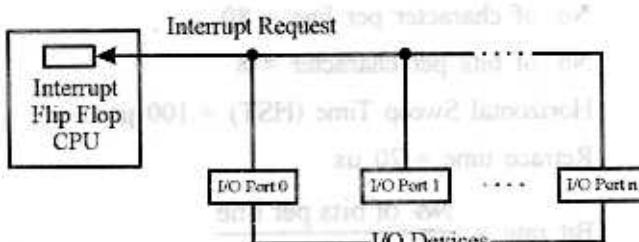
Notice that conflict occurs in placement of block 55. Because 3 comes first and never reuse in set 3. So this is only victim for replacement.

### S.52 (d)

Memory-mapped I/O requires that memory locations and I/O ports share the same set of addresses. So on address bit pattern that is assigned to memory can not be assigned to an I/O port and vice versa. **I/O protection in this approach is ensured by operating systems macros and routines.**

### S.53 (d)

Consider the following figure



A single line interrupt system contains a single interrupt request line and a interrupt grant line. In this system it may be possible that more than one output devices request interrupt at the same time for example I/O port 0 to I/O port n may request an interrupt at the same time but only one request will be granted according to priority. So, In single interrupt system vectored interrupts are not possible but multiple interrupting devices are possible.

### S.54 (c)

The swap space is basically used for saving process data. Consider CPU contains two process  $P_1$  and  $P_2$ , if  $P_1$  is running on the CPU then  $P_2$  is swapped out and all data (for process, content etc.) are saved in the disk. When CPU executes process  $P_2$  then all data items of  $P_1$  are saved on a disk.

(c) 12.2

Ans + three columns that follows the rule.

| Page | Time | Count | Op | Page | Time | Count | Op |
|------|------|-------|----|------|------|-------|----|
| 21   | ?    | 01    | ?  | 7    | ?    | 01    | ?  |
| 7    | ?    | 01    | ?  | 21   | ?    | 01    | ?  |

**UNIT-4**

# **PROGRAMMING**

# Unitär PROGRAMMING

1

# 4.1

## C-FUNDAMENTALS

### LEVEL-1

- Q.1** By default any real number in 'C' is treated as  
 (a) a float  
 (b) a long double  
 (c) depends upon the memory model that you are using  
 (d) a double
- Q.2** Trace the output:  
 main ()  
 {  
 print ("%c", abcdefgh[4]);  
 }  
 (a) abcdefgh  
 (b) Error  
 (c) e  
 (d) d
- Q.3** The declarations  

```
typedef float ht[100];
ht men, women;
defines _____
```

  
 (a) men and women as floating point variables  
 (b) ht, men and women as floating point variables  
 (c) men and women as 100 element floating point array  
 (d) none of these
- Q.4** If y is of integer type, then the expressions  
 $3 * (y - 8) / 9$  and  $(y - 8) / 9 * 3$   
 (a) may or may not yield the same value  
 (b) must yield the same value  
 (c) must yield different values  
 (d) none of the above
- Q.5** The difference between rand( ) and srand( ) is  
 (a) Type of rand( ) function is int and srand( ) is void  
 (b) There is no difference between rand( ) and srand( ) both do same function  
 (c) The srand( ) function is used to initialize the random number generator whereas rand( ) function returns a random positive integer.  
 (d) both (a) and (c)
- Q.6** To scan a and b given below which scan f( ) statement would you use?  
 float a;  
 double b;  
 (a) scanf("%f%lf", &a, &b);  
 (b) scanf("%f%Lf", &a, &b);  
 (c) scanf("%Lf%Lf", &a, &b);  
 (d) scanf("%f%f", &a, &b);

**Q.7** Maximum combined length of the command line arguments including the spaces between adjacent arguments is

- (a) It may vary from one operating system to another
- (b) 67 characters
- (c) 256 characters
- (d) 128 characters

**Q.8** The following program fragment

```
int k = - 7 ;
printf("%d", 0 < !k);
```

- (a) prints an unpredictable value
- (b) is illegal
- (c) prints a non-zero value
- (d) prints 0

**Q.9** Which of the following are not keywords in C?

- (a) printf
- (b) main
- (c) IF
- (d) none of the above.

**Q.10** If variable can take any integral values from 0 to n, where n is a constant integer, then the variable can be represented as a bits field whose width is the integral parts of-

- (a)  $\log_2(n)+1$
- (b)  $\log_2(n+1)+2$
- (c)  $\log_2(n-1)+1$
- (d) None of these

**Q.11** The following lines, if included in a program, will cause one of the following errors. Indicate the correct one.

```
{  
    double c;  
    scanf("%c",c);
```

- (a) Runtime error
- (b) Compilation error
- (c) Typedef error
- (d) No error

**Q.12** What will be the output of the following program?

```
#include <stdio.h>
```

```
int main()  
{
```

```
    char * s1;  
    char far * s2;
```

```
    char huge * s3;
```

```
    printf ("%d%d\n", sizeof(s1), sizeof(s2),
```

```
    sizeof(s3));
```

```
    return 0;
```

```
}
```

- (a) 442 in TC/C++ compiler

- (b) error in code

- (c) error in VC++ or gcc compiler

- (d) None of the above

**Q.13** Which of the following is the correct output for the program given below?

```
#include <stdio.h>
```

```
void fun (int);
```

```
int main ()
```

```
{
```

```
int a ;
```

```
a = 3;
```

```
fun (a);
```

```
printf ("\n");
```

```
return 0;
```

```
}
```

```
void fun (int n)
```

```
{
```

```
if (n > 0)
```

```
{
```

```
    fun (-n);
```

```
    printf ("%d", n);
```

```
    fun (-n);
```

```
}
```

```
}
```

- (a) 0 2 1 0

- (b) 1 1 2 0

- (c) 0 1 0 2

- (d) 0 1 2 0

**C-FUNDAMENTALS**

**Q.14** What do you mean by a translation unit?

- (a) a set of source files seen by the compiler and translated as a unit.
- (b) a set of linkers for compiler and translated as a unit.
- (c) is a database that stores so called "segments".
- (d) Is a language interface provider.

**Q.15** Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main()
{
    printf("%d%d\n", 32<<1, 32<<0);
    printf("%d%d\n", 32<<-1, 32<<-0);
    printf("%d%d\n", 32>>1, 32>>0);
    printf("%d%d\n", 32>>-1, 32>>-0);
    return 0;
}
```

- (a) garbage values
- (b) 64 32
- (c) all zeros
- (d) 8 0

32 0  
0 16

**Q.16** C is a

- (a) high level language
- (b) low level language
- (c) high level language with some low level features.
- (d) low level language with some high level features.

**Q.17** #include <stdio.h>

```
int main ()
{
    auto int i = 100;
    printf ("i = %d\n", i);
    {
        int i = 1;
        printf ("i = %d\n", i);
    }
    printf ("i = %d\n", i);
```

(a) i = 100

i = 2

i = 2

i = 2

(b) i = 100

i = 100

i = 101

(c) i = 100

i = 1

i = 2

i = 2

(d) i = 100

i = 100

i = 101

i = 101

i = 100

(e) i = 100

(f) i = 100

(g) i = 100

(h) i = 100

- Q.18** Even if a particular implementation doesn't limit the number of characters in an identifier, it is advisable to be concise because  
 (a) chances of typographic errors are less  
 (b) it may be processed by assembler, loaders, etc. which may have their own rules that may contradict the language rules  
 (c) by being concise, one can be mnemonic  
 (d) None of the above
- Q.19** Coercion  
 (a) takes place across an assignment operator.  
 (b) takes place if an operator has operands of different data types.  
 (c) means casting  
 (d) none of the above
- Q.20** Which of the following comments are true?  
 (a) C provides non input-output features.  
 (b) C provides no file access features.  
 (c) C borrowed most of its ideas from BCPL.  
 (d) C provides no features to manipulate composite objects.
- Q.21** Which of the following comments about wide characters is/are true?  
 (a) It is the binary representation of a character in the extended binary set.  
 (b) It is of integer type wchar\_t.  
 (c) End of file is represented by WEOF.  
 (d) None of the above.
- Q.22** int i = 5;  
 is a statement in a C program. Which of the following are true?  
 (a) during execution, value of i may change but not its address.  
 (b) during execution both the address and value may change  
 (c) repeated execution may result in different addresses for i  
 (d) i may not have an associated address
- Q.23** C preprocessor  
 (a) takes care of conditional compilation  
 (b) takes care of macros  
 (c) takes care of include files  
 (d) acts before compilation
- Q.24** The statement printf("%d", 10?0?5:11:12); prints  
 (a) 10  
 (b) 0  
 (c) 12  
 (d) 11
- Q.25** What are the files which are automatically opened when a C file is executed?  
 (a) Stdin  
 (b) Stdout  
 (c) Stderr  
 (d) All of the above
- Q.26** The fields in a structure of a C program are by default  
 (a) private  
 (b) public  
 (c) protected  
 (d) none of the above

## LEVEL-2

**Q.27** main ( )

```
{     int a =6, b =10, x;
x=a & b;
printf("%d", x);
}
```

find the value of x.

- (a) 2
- (b) 10
- (c) 5
- (d) 7

**Q.28** What will be the values of a, b and c after execution?

```
int a, b, c;
```

```
b = 10;
```

```
c = 15;
```

```
a = ++b + c++;
```

- (a) a = 27, b = 11, c = 16
- (b) a = 26, b = 11, c = 16
- (c) a = 25, b = 10, c = 15
- (d) a = 27, b = 10, c = 15

**Q.29** Consider a following declaration

```
main ( )
{
    int i, n;
    float p;
    scanf("%d%d%f", &i, &n, &p);
    print("%f", p*pow((1+i), n));
}
```

Select the correct statement

- (a) The above program is also use for  $(1 + i^1 + i^2 + i^3 + \dots + i^n)^n$
- (b) The above program is use for finding  $(a + (a + 1)^1 + (a + 2)^2 + \dots)$
- (c) The above program is use for finding compound interest.
- (d) The above program is use for finding the fibonacci series with n numbers and n power.

**Q.30** Consider a following declaration:

```
int i = 8, j = 5;
float x = 0.005, y = -0.01;
char c = 'c', d = 'd';
f = 2 * ((i/5) + (4 * (j - 3))%(i + j - 2));
```

Then, the data type of f is \_\_\_\_\_ and value is \_\_\_\_\_

- (a) float, 18
- (b) int, 13
- (c) int, 18
- (d) float, 13

**Q.31** What is the output of the following 'C' program?

```
main ( )
{
```

```
    int i = 2;
    printf("%d%d", ++i, ++i);
}
```

- (a) 4 4
- (b) 3 4
- (c) 4 3
- (d) Output may vary from compiler to compiler

**Q.32** In the following 'C' code in which order the functions would be called?

```
a = (f1 (23, 14) * f2 (12/4)) + f3 ( );
```

- (a) None of these
- (b) f3, f2, f1
- (c) f1, f2, f3
- (d) The order may vary from compiler to compiler

**Q.33** According to ANSI specifications which is the correct way of declaring main ( ) when it receives command line arguments?

(a) main ( )

```
{  
    int argc; char*argv[ ];  
}
```

(b) main (argc, argv)

```
int argc; char*argv[ ];
```

(c) main (int argc, char argv [ ]) (d)

(d) None of the above

**Q.34** What does the following 'C' program do?

main ( )

```
{  
    unsigned int num;  
    int i;  
    scanf("%u", &num);  
    for (i = 0; i < 16; i++)  
        printf("%d", (num<< i & 1 << 15)? 1 : 0);  
}
```

(a) It prints binary equivalent of num

(b) It prints all odd bits from num

(c) It prints all even bits from num

(d) None of these

**Q.35** What is the output of the following 'C' program?

main ( )

```
{  
    unsigned int m = 32,  
    printf("%x", ~m);  
}
```

(a) ddfdf

(b) ffdf

(c) ffffff 001 = a 00 = d 000 = g (d)

(d) 0000 001 = a 00 = d 000 = g (b)

**Q.36** What is the output of the following 'C' program?

```
main ()
{
    extern int fun (float);
    int a;
    a = fun (3.14);
    printf ("%d", a);
}
int fun (aa)
{
    float aa;
    {
        return ((int) aa);
    }
}
```

(a) Error  
 (b) 3.14  
 (c) 0  
 (d) 3

**Q.37** Write a program to swap two numbers:

```
#include<stdio.h>
int main ()
{
    int a,b, temp;
    clrscr ();
    a=100, b=a+200;
    printf ("before swapping a=%d and
b=%d\n",a,b);
    temp =a;
    a=b;
    b=temp;
    printf ("After swapping a=%d and b=%d",a,b);
}
```

- (a) a=200, b=100; a=100, b = 200  
 (b) a=100, b=300; a = 300, b = 100  
 (c) a=300, b=100; a = 100, b = 300  
 (d) a=200, b=200; a = 200, b = 200

**Q.38** Consider a following declaration

```
int i =7;
float f = 8.5;
```

then which of the following statement is correct?

- (i+f)%4
- ((int)(i+f))%2
- ((float)(f+i))%4
- ((int)f)%2

(a) Only (ii)  
 (b) (i),(iii)  
 (c) (ii),(iv)  
 (d) All of the above

**Q.39** Trace the output:

```
main ()
{
    float a =-4.9, b = 8.6; int c;
    c=floor (a)*a-(-b);
}
```

(a) 28.2  
 (b) 28  
 (c) 33.1  
 (d) 33

**Q.40** Trace the final value of s and i

```
void main ()
{
    int c =1, s=0 i=1;
    while (c<=5)
    {
        s=s+i,
        i=i+2;
        c+=1;
    }
    print ("%d %d", s,i);
    getch () ;
}
```

- (a) 36 13  
 (b) 25 11  
 (c) 64 17  
 (d) 81 21

**Q.41** The following program

```
main () {
    static int a [ ] = {7,8,9};
    printf ("%d", 2[a]+a[2]);
}
```

(a) results in bus error  
 (b) results in segmentation violation error  
 (c) will not compile successfully  
 (d) none of the above

**Q.42** Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main () {
    int x, y, z;
    x = y = z = 1;
    z = ++x || ++y && ++z;
    printf ("x = %d y = %d z = %d\n", x, y, z);
    return 0;
}
```

(a) x = 2 y = 1 z = 1  
 (b) x = 2 y = 2 z = 1  
 (c) x = 2 y = 2 z = 2  
 (d) x = 1 y = 2 z = 1

**Q.43** Choose the correct statements.

- (a) Constant expressions are evaluated at compile time
- (b) String constants can be concatenated at compile time
- (c) Size of array must be known at compile time
- (d) None of the above

**Q.44** In standard C, trigraph in the source program are translated

- (a) before the lexical analysis
- (b) after the syntax analysis
- (c) before the recognition of escape characters in strings
- (d) during the intermediate code generation phase

**Q.45** #include <stdio.h>

```
void pri (int, int);
void printit (float, int);
int main( )
{
    float a = 3.14;
    int i = 99;
    pri (i, a);
    printit (a, i);
    return 0;
}
void pri (int i, int a)
{
    printf ("i=%d a=%f\n", i, a);
    printf ("a=%f i=%d\n", a, i);
}
void printit (float a, int i)
{
    printf ("a=%f i=%d\n", a, i);
    printf ("i=%d a=%f\n", i, a);
}
(a) i = 99 a = 3.000000
     a = 3.000000 i = 99
     a = 3.140000 i = 99
     i = 99 a = 3.140000
(b) i = 99 a = 0.000000
     a = 0.000000 i = 124503
     a = 3.140000 i = 99
     i = 99 a = 3.140000
(c) i = 99 a = 0
     a = 0 i = 99
     a = 3.14 i = 99
     i = 99 a = 3.14
(d) None of the above
```

## LEVEL-3

```
#include <stdio.h>
int main( )
{
    int i, j, k;
    for (j=1; j<=4; j++)
    {
        if (j*j==16)
            goto secretplace;
    }
    for (i=1; i<=5; i++)
    {
        k=i*i;
        j=k+2;
        secretplace:
        printf("Murphy's second law\n");
        printf("Good computers are
always priced..\n");
        printf("just beyond your
budget\n");
    }
    return 0;
}
```

select the correct statement

(a) Murphy's second law

Good Computers are always priced

.....

Just beyond your budget.

(b) Murphy's second law

Good Computers are always priced

.....

just beyond your budget.

After this the code causes a runtime exception.

(c) only runtime exception

(d) None of the above

## GATE QUESTIONS

**Q.46** #include <stdio.h>

Let  $x$  be an integer which can take a value of 0 or 1. The statement  $\text{if } (x == 0) \text{ } x = 1; \text{ else } x = 0;$  is equivalent to which one of the following?

[IT-GATE 2004]

[1 Mark]

- (a)  $x = 1 + x;$
- (b)  $x = 1 - x;$
- (c)  $x = x - 1;$
- (d)  $x = 1 \% x;$

**Q.48** A common property of logic programming languages and functional languages is

[GATE 2005]

[1-Mark]

- (a) both are declarative
- (b) all of the above
- (c) both are procedural language
- (d) both are based on  $\lambda$ -calculus

**Q.49** What is the output printed by the following program?

```
# include <stdio.h>
int f(int n, int k) {
    if (n == 0) return 0;
    else if (n%2) return f(n/2, 2*k) + k;
    else return f(n/2, 2*k) - k;
}
int main () {
    printf("%d", f(20,1));
    return 0;
}
```

[IT-GATE 2005]

[2-Marks]

- (a) 5
- (b) 8
- (c) 9
- (d) 20

# ANSWER KEY

|    |         |    |      |    |         |    |         |    |      |
|----|---------|----|------|----|---------|----|---------|----|------|
| 1  | d       | 2  | b    | 3  | c       | 4  | a       | 5  | a, c |
| 6  | a       | 7  | a    | 8  | d       | 9  | a, b, c | 10 | a    |
| 11 | a       | 12 | a, c | 13 | d       | 14 | a       | 15 | b    |
| 16 | c       | 17 | c    | 18 | a, b    | 19 | a, b    | 20 | all  |
| 21 | a, b, c | 22 | b, c | 23 | all     | 24 | d       | 25 | d    |
| 26 | b       | 27 | a    | 28 | b       | 29 | c       | 30 | c    |
| 31 | d       | 32 | c    | 33 | c       | 34 | a       | 35 | b    |
| 36 | d       | 37 | b    | 38 | c       | 39 | d       | 40 | b    |
| 41 | d       | 42 | a    | 43 | a, b, c | 44 | a, c    | 45 | b    |
| 46 | b       | 47 | b    | 48 | a       | 49 | c       |    |      |

## SOLUTIONS

**S.3 (c)**

We are defining an array of 100 float value with **typedef**.

**S.4 (a)**

Since we are doing / and \* operations alternately, therefore we may or may not yield the same value.

**S.5 (a, c)**

| <b>rand( )</b>                       |                                                  | <b>srand( )</b> |  |
|--------------------------------------|--------------------------------------------------|-----------------|--|
| 1. Type : int                        | 1. Type : void                                   |                 |  |
| 2. returns a random positive integer | 2. use to initialize the random number generator |                 |  |

**S.8 (d)**

$k = -7$ . So, if 'k' is used as a Boolean variable, it will be treated as a true condition. So, !k will be false i.e., 0. So,  $0 < ? !k$  is actually  $0 < 0$ , which is false. So, 0 will be printed.

**S.9 (a, b, c)**

IF is not a keyword, because it is in upper

**S.10 (a)**

Variable with integral values from 0 to n.

Then its width is the integral part of  $\log_2(n)+1$ .

**S.11 (a)**

Since, we are declaring C as double, and we are printing C as character type.

**∴ Runtime error.**

**S.12 (a, c)**

In a 16-bit compiler like Turbo C/C++ the output will be 4 4 2. However, in a 32-bit compiler like VC++ or gcc or Visualstudio, the compiler will report an error since 32-bit compilers do not recognize near, far and huge pointers. In 32-bit compilers every pointer is 4 bytes wide.

**S.14 (a)**

A translation unit is a set of source files seen by the compiler and translated as a unit—generally one '.c' file, plus all header files mentioned in #include directives.

**S.17 (c)**

Output

i = 100

i = 1

i = 2

i = 2

i = 100

**Explanation:**

In the outermost block (a block is statements within a pair of braces) the variable i has been declared as an auto storage class variable, with an initial value 100. This value gets printed through the first `printf()`. Then the control reaches inside the next block, where again i is declared to have a value 1. This i is different, since it has been defined inside another block. The value of this i is then printed out through the second `printf()`. Inside the next block, this i is incremented to 2 and then printed out. Then the control reaches outside this block where another `printf()` is encountered. This `printf()` is within the same block in which the second i has been defined. Hence it prints out the value of this i, which is still 2. And now the control reaches outside the block in which the second i has been defined. Therefore the second i dies. The first i is however still active, since the control has not gone out of the block in which it has been defined. Hence the last `printf()` prints out the value of this i, which is 100.

**S.24 (d)**

STEP 1: 0 ? 5 ; 11, which is false, so, 11

STEP 2: 10 ? 11 ; 12 which is true, so, 11

**S.25 (d)**

All are required i.e., Standard input, Standard output, Standard error.

**S.27 (a)**

(6)<sub>10</sub> is represented as (0110)<sub>2</sub> in binary.

Similarly, (10)<sub>10</sub> is represented as (1010)<sub>2</sub>

$$\begin{array}{r} 0110 \\ \& 1010 \\ \hline 0010 \end{array} \Rightarrow 2$$

**S.28 (b)**

b = 10, c = 15

a = 11 + 15 = 26

∴ a = 26, b = 11, c = 16

**S.29 (c)**

The formula `p*(pow((i + 1), n))` is used for calculating compound interest.

**S.30 (c)**

$$\begin{aligned} f &= 2 * ((8/5) + (4*(2))\%(11)) \\ &= 2 * ((1 + 8)\%(11)) \\ &= 2 * (9) \\ &= 18 \Rightarrow \text{data type} \Rightarrow \text{int} \end{aligned}$$

**S.31 (d)**

The unary increment/decrement operator has the right to left associativity. Hence starting from right, the first values is 3 and 4 going from right to left.

**S.32 (c)**

f1, f2, f3

Hence function are evaluated according to the arithmetic expression precedence of arithmetic operator.

**S.33 (c)**

In ANSI standard declares the variables inside the parenthesis.

**S.34 (a)**

`(num << i & 1 << 15) ? 1 : 0`

This statement return either 0 or 1 depending on the condition inside the parenthesis. Hence printf becomes either `printf ("%d", 1);` or `printf ("%d", 0);`

**S.35 (b)**

$m = (32)_{10} = (0000\ 0000\ 0010\ 0000)_2$

Now,  $-m = (1111\ 1111\ 1101\ 1111)_2$

= `ffdf`

**S.36 (d)**

There is casting from float to int data type.

**S.38 (c)**

We cannot apply % operator on floating point value

(i)  $(i+f)\%4$  is invalid, because  $(i+f)$  is floating value

(iii)  $((float)(f+i))\%4 \Rightarrow$  floating value.

**S.39 (b)**

$$\begin{aligned} C &= \text{floor}(a)*a - (-b) \\ &= \text{floor}(-4.9)*(-4.9)+8.6 \\ &= -5*-4.9+8.6 \\ &= 33.1 \end{aligned}$$

But due to `int c`  $\Rightarrow 33$ .

**S.40 (c)**

In while loop,  $i <= 5$  is true

Then in the while loop condition is false

| $1 \leq i$ | $2 \leq i$ | $3 \leq i$ | $4 \leq i$ | $5 \leq i$  |
|------------|------------|------------|------------|-------------|
| $S=0+1=1$  | $S=1+3=4$  | $S=4+5=9$  | $S=9+7=16$ | $S=16+9=25$ |
| $i=1+2+3$  | $i=3+2=5$  | $i=5+2=7$  | $i=7+2=9$  | $i=9+2=11$  |
| $C=C+i=2$  | $C=2+1=3$  | $C=3+1=4$  | $C=4+1=5$  | $C=6$       |

**Q.41 (d)**

$a[2]$  will be converted to  $a(a+2)$ .

$*(a+2)$  can as well be written as  $*(2+a)$ .

$*(2+a)$  is nothing but  $2[a]$ . So,  $a[2]$  is same as  $2[a]$ , which is same as  $*(2+a)$ . So, it prints  $9 + 9 = 18$ . Some of the modern compilers don't accept  $2[a]$ .

**S.45 (b)**

Output

$i = 99$   $a = 0.000000$

$a = 0.000000$   $i = 124503$

$a = 3.140000$   $i = 99$

$i = 99$   $a = 3.140000$

**Explanation:**

When `pri()` is called the values passed to it are collected in the variable `i` and `a`. The first `printf()` in `pri()` prints out the value of `i` correctly, but value of `a` gets messed up because we are trying to print an integer value collected in `a` using the specification `%f`. The second `printf()`

) messes up both the values since the first specification itself in this `printf()` is wrong. Remember that once the output of one variable goes awry, the `printf()` messes up the output of the rest of the variables too.

When `printit()` is called `a` has been declared as `float` whereas `i` has been declared as an `int`. Therefore both the `printf()`'s output the values as expected.

**S.46 (b)**

Murphy's second law

Good computers are always priced...

just beyond your budget

After this the code causes a runtime exception

**Explanation:**

Look at the first `for` loop. The moment  $j*j$  equals 16, the `goto` statement is executed, which takes the control to `secretplace` inside the second `for` loop. This is perfectly acceptable, `goto` can virtually take the control anywhere— even deep inside a `for` loop. Having reached the `secretplace`, Murphy's second law is printed out and then the control reaches the closing brace of the `for` loop. As a result, control jumps to the beginning of loop i.e. to `i++`. Here it attempts to increment `i`. However, since `i` has not been initialized it would hold a garbage value because we entered the loop directly at `secretplace` as a result of which `i = 1` couldn't get executed. That is why a runtime exception is raised with a message ‘The variable ‘i’ is being used without being initialized.

**S.47 (b)**

Given, C code is:

```
if (x == 0)
    x = 1;
else
    x = 0;
```

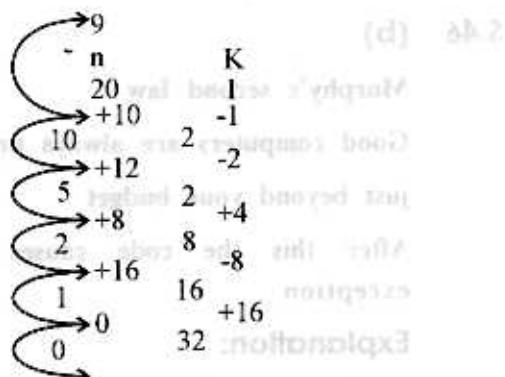
If `x` is zero then set `x = 1` else set `x` which is same as `x = 1 - x`.

**S.48 (a)**

A common property of logic programming language and functional languages is **both are declarative because we declare any statement before we will use it.**

**S.49 (c)**

The program execution sequence is



**So final result is 9.**



second solar system's a blood disease to be treated  
it is developed in different pool cell protein, we  
will have to increase the infection rate  
a new blood is developed without a cell a  
neutral base good to fight disease off organism  
blood that good

(d) 50.2

x also 7 more

$$(0 = x)^2$$

$$1 = x$$

$$x^2 = 1$$

$$x = \pm 1$$

as double x the only  $1 < x$  the next case is  $x = 1$   
 $x = 1 = x$  so same

(b) 86.2

more points on vectors of right angle all

solve

smaller or equal number of integer (0)

solve

value greater than 0 (0.0000000000000002)

(c) 96.2

(d)  $f(x) = 0$  for  $x = 3$

$f(3) = (3-1)(3-2) + 1000$

$= 2 \times 2 + 1000$

$= 1004$

$f(x) = 0$  at  $x = 0$

(d) 0.2

value of  $f(x) < 0$  point of

select a nonlinear goal statement null

| File    | Format | Font            | Size | Color | Style  |
|---------|--------|-----------------|------|-------|--------|
| Normal  | Auto   | Times New Roman | 10   | Black | Normal |
| Page    | Normal | Times New Roman | 10   | Black | Normal |
| Section | Normal | Times New Roman | 10   | Black | Normal |
| Text    | Normal | Times New Roman | 10   | Black | Normal |

(e) 10.0

square of bottom of the circle

$(x-1)^2 = 100$  or  $x = 10$  or  $x = -10$

or  $x = 11$  or  $x = -9$  tell question is in  $\mathbb{Z}^+$

so  $x = 11$  or  $x = 10$  or  $x = 9$  or  $x = 8$  or  $x = 7$

or  $x = 6$  or  $x = 5$  or  $x = 4$  or  $x = 3$  or  $x = 2$

(f) 10.0

(d) 24.2

square

difference of  $100 - 100$

$10000 - 10000 = 0$

$1000000 - 1000000 = 0$

invention

one is of lesser value with higher is 1. Now will

high  $x$  is less, i. smaller val. in position

i. to value of the sum is less in 1. Then

summed up because there is no value in general one the

higher because off. So most probably odd values is in

In which year was the first computer built?

In which year was the first computer built?

ANSWER

## CONTROL STATEMENTS

**4.2**

### LEVEL-1

- Q.1** Which of the following is possible with the help of “goto” statement?
- Branching around statements or group of statements under certain conditions.
  - Jumping to the end of a loop under certain conditions, thus bypassing the remainder of the loop during the current pass
  - Jumping completely out of a loop under certain conditions, thus determining the execution of a loop.
- (a) (i), (ii), (iii)  
 (b) (i), (iii)  
 (c) only (i)  
 (d) (ii), (iii)
- Q.2** Consider for loop in a C program. If the condition is missing
- It is assumed to be present and taken to be true
  - It results in a syntax error
  - It is assumed to be present and taken to be false
  - Execution will be terminated abruptly
- Q.3** In the following ‘C’ program, find out the error in the ‘while’ loop, if any?
- ```
main ()
{
    int i = 1;
    while ( )
    {
        printf("%d", i++);
        if (i > 10)
            break;
    }
}
```
- (a) There should be at least a semicolon in the while ( )  
 (b) The while loop should be replaced by for loop  
 (c) No error  
 (d) The condition in the while loop is a must
- Q.4** A “switch” statement is used to
- switch between functions in a program
  - to choose from multiple possibilities which may arise due to different values of a single variable
  - switch from one variable to another variable
  - to use switching variable

**Q.5** Static variables are sometimes called

- (a) class variables
- (b) functional variables
- (c) dynamic variables
- (d) auto variables

**Q.6** Trace the o/p

```
main()
{
    int x=10,y=10,z=5;
    if(x<y<z);
    printf("/n%d",i);
}
```

- (a) 1
- (b) 0
- (c) Error
- (d) 5

**Q.7** Consider the following program fragment

```
sum = 0;
do { scanf("%d",&x);
      if (x>0) printf ("%d",x);
      else
          if (x==0) break;
      sum +=x;
} while (sum < 10);
```

What would be the output if input is 5,2,0,3,0?

- (a) 5 2
- (b) 5 2 0
- (c) 5 2 0 3
- (d) 5 2 3

**Q.8** The following program fragment:

```
for (i=1; i<5 ;i+=3), {
    print ("%d",i);
}
results in
(a) a syntax error
(b) an execution error
(c) printing of 7
(d) printing of 16
```

**Q.9** What is the output of the following program?

```
main()
{
    int x,y,z;
    x= 2;
    y=1;
    z=1;
    if (x>y+z)
        printf("Hello/n");
    else if (x<y+z)
        printf(Hi/n");
    else
        printf("Hey!/n");
    }
}
(a) Hello!
(b) Hi!
(c) Hey!
(d) none of these
```

**Q.10** Which of the following statements are correct about the C program given below?

```
#include<stdio.h>
int main()
{
    int x = 10, y = 100% 90;
    for (i = 1; i <= 10; i++);
    if (x != y);
        printf ("x = %d y = %dn", x,y),
    return 0;
}
```

- (a) The printf() function is called 10 times.
- (b) The program will produce the output x = 10 y = 10.
- (c) The ; after the if (x != y) will NOT produce an error.
- (d) The program will not produce any output.
- (e) The printf() function is called infinite times.

**CONTROL STATEMENTS**

**Q.11** Point out the error, if any, in the following program.

```
#include < stdio.h>
int main ()
{
    int i = 1;
    switch (i)
    {
        case 1 * 2 + 4:
            printf("\n Bottle per rent-inquired within.\n");
            break;
        case 1 :
            printf("\n Radioactive cats have 18 half-
lines.\n");
            break;
    }
}
```

(a) Bottle for rent-inquire within  
 (b) Radioactive cats have 18 half-lives.  
 (c) error  
 (d) 6

**Q.12** Rewriting following of statements using conditional operators will result into:

```
int a = 1, b;
if (a > 10)
    b = 20;
(a) int a = 1, b;
     a > 10? b = 20 : ;
(b) int a = 1, b;
     a > 10? b = 20 : (0);
(c) int a = 1, b;
     a > 10? (b = 20) : ;
(d) int a = 1, b, dummy;
     a > 10? b = 20: dummy = 1;
```

**Q.13** Choose the statements that are syntactically correct.

- (a) /\* Is /\* this a valid \*/ comment \*/
- (b) for (; ; );
- (c) return ;
- (d) return (5 + 2);

**Q.14** Consider the statements

```
putchar(getchar());
putchar(getchar());
```

If

a

b

is the input, the output will be

- (a) an error message
- (b) this can't be the input
- (c) ab
- (d) a b

**Q.15** Integer division results in

- (a) truncation
- (b) rounding
- (c) overflow
- (d) None of the above

**Q.16** Pick the operators that associate from the right.

- (a) ?:
- (b) +=
- (c) =
- (d) <

**Q.17** The following code fragment

```
int x, y = 2, z, a;
x = (y *= 2) + (z = a = Y),
print f("%d", x);
```

- (a) prints 8
- (b) prints 6
- (c) prints 6 or 8 depending on the compiler implementation
- (d) is syntactically wrong

**Q.18** The following statement

```
print f ("%f", 9/5);
```

prints

- (a) 1.8
- (b) 1.0
- (c) 2.0
- (d) none of the above

**Q.19** #include <stdio.h>  
 int main()  
 {  
 unsigned int ch = 0;  
 for (ch = 65; ch <= 225;)  
 printf ("%d %c\n", ch, ch++);  
 return 0;  
 }  
 (a) 65 A  
 66 B  
 ...  
 125 }  
 126 ~  
 ...  
 255  
 ...  
 65 A  
 66 B  
 ...  
 ...  
 (b) 66 A  
 67 B  
 ...  
 125 |  
 126 }  
 ...  
 255  
 256  
 (c) 65 A  
 66 B  
 ...  
 125 }  
 126 ~  
 ...  
 255  
 256  
 (d) 66 A  
 67 B  
 ...  
 125 |  
 126 }  
 ...  
 255  
 ...  
 66 A  
 67 B  
 ...  
 ...

**Q.20** Which of the following comments about **for** loop are correct?

- (a) Using 'break' is equivalent to using a 'goto' that jumps to the statement immediately following the loop.
- (b) Continue is used to by-pass the remainder of the current pass of the loop.
- (c) If comma operator is used, then the value returned is the value of the right operand.
- (d) It can always be replaced by a 'while' loop.

**Q.21** The 'switch' feature

- (a) can always be replaced by a nested if-then-else clause
- (b) enhances logical clarity
- (c) can't always be replaced by a nested if-then-else clause
- (d) none of the above

**Q.22** Choose the best answer.

Storage class defines

- (a) the data type
- (b) the scope
- (c) the scope and permanence
- (d) the scope, permanence and data type

**Q.23** Consider the following program segment.

```
i = 6720 ; j = 4 ;
while ((i%j) == 0)
{
  i = i / j ;
  j = j + 1;
}
```

On termination j will be the value

- (a) 4
- (b) 8
- (c) 9
- (d) 6720

**Q.24** The program

```
main ()
{
    int i = 5;
    i = (++i) / (i++);
    printf("%d", i);
}
```

prints

- 2
- 5
- 1
- 6

**Q.25** The process of transforming one bit pattern into another by bit-wise operations is called

- masking
- pruning
- biting
- chopping

**Q.26** The operation of a staircase switch best explain the

- OR operation
- AND operation
- XNOR operation
- XOR operation

**Q.27** The number of possible values of m, such that m & 0x3f equals 0x23 is

- 1
- 2
- 3
- 4

## LEVEL-2

**Q.28** If a = 9, b = 5 and c = 3, then the expression (a - a/b \* b%c) > a%b%c evaluates to

- true
- false
- invalid
- 0

**Q.29** In following code, find out the error in 'while' loop, if any

```
main ()
{
    int i = 1;
    while ()
    {
        printf("%d", i++);
        if (i > 10)
            break;
    }
}
```

- The while loop should be replaced by for loop
- There should be at least a semicolon in the while
- The condition in the while loop is a must
- No error

**Q.30** Consider a following declaration

```
main ()
{
    sum = 0;
    do { scanf("%d", &i);
        if (i < 0)
            i = -1;
        ++ flag;
        } sum += i;
    } while (i != 0);
}
```

which of the following statement is correct?

- The program segment itself is a compound statement
  - The do-while statement, which is embedded in the program segment, contains a compound statement.
  - The if statement, which is embedded in the do-while statement, contains a compound statement.
- only (ii) & (iii)
  - (i), (ii), (iii)
  - only (i) & (ii)
  - neither of (i), (ii), (iii)

**Q.31** Consider the following declaration for (x = 1; x

```

<=10; x++)
{
    for (y = x; y <= 10; y++)
        printf ("%d", x),
        printf ("\n");
}

```

The (sum of all elements of 4<sup>th</sup> row) - (sum of all elements of 7<sup>th</sup> row) = \_\_\_\_\_

- (a) 7
- (b) 0
- (c) 5
- (d) 3

**Q.32** Consider the following declarations:

count = 0;

```

do
{
    printf("%d", text [count]);
    ++count;
}

```

while (text [count] != '\*');

Which of the following statement is incorrect?

- (a) If we enter '\*' as first character then also this program will execute atleast once.
- (b) For the above code, if we replace do-while by while then program execution is slightly faster.
- (c) We can replace text [count] in output statement as text [count ++], without writing afterwards.
- (d) (a), (b) & (c) are wrong.

### Common Data For Questions 33 & 34:

```

main ()
{
    int n = 1, sum = 0;
    while (n <= 10)
    {
        sum += n * n++;
    }
    printf("sum = %d\n", sum);
}

```

**Q.33** What is the output of the above code?

- (a) 2121
- (b) 385
- (c) 440
- (d) Indefinite loop

**Q.34** If the statement 4 is replaced by

Sum += n ++ \* n ++;

What will be the difference between the new and old output?

- (a) 0
- (b) 218
- (c) 220
- (d) -218

**Q.35** void main ()

```

{
    int stud [ ] [2] = {1234, 56, 1212, 33, 1434,
80, 1312, 778, 1203, 75};
    // let stud [ ] [2] start at 4001.
    int i, j, sum = 0, j=0;
    for (i = 0; i <= 4; i++, j++)
    {
        if (j == 2) j = 0;
        sum += *(stud + i) + j);
    }
    printf("%d", sum),
}

```

- (a) 2735
- (b) 6717
- (c) 3880
- (d) 4682

**Q.36** What is the output of the following 'C' fragment?

```
for (i = 1, j = 10; i < 6; ++i, -j)
```

```
printf("%d%d", i, j);
```

- (a) 1 1 1 1 1 9 9 9 9 9

- (b) none of these

- (c) 1 10 2 9 3 8 4 7 5 6

- (d) 1 2 3 4 5 10 9 8 7 6

**Q.37** Match the following:

Group I		Group II	
1.	Type declaration instructions	a.	$k = i * 234 + n - 7;$
2.	Input/Output instructions	b.	for( $i = 0; i \leq 10; i++$ );
3.	Arithmetic instructions	c.	gets();
4.	Control instruction	d.	char name, code;

- (a) 1 – b, 2 – a, 3 – d, 4 – c
- (b) 1 – d, 2 – c, 3 – a, 4 – b
- (c) 1 – c, 2 – b, 3 – a, 4 – d
- (d) 1 – a, 2 – d, 3 – b, 4 – c

**Q.38** How many times will the printf statement be executed?

```
main ( )
{
    int n;
    n = 10;
    while (n < 10)
        printf("hello");
    -- n;
}
```

- (a) once
- (b) 9
- (c) 10
- (d) never

**Q.39** Consider the program fragment

```
switch(choice){
    case 'R' : printf("RED");
    case 'W' : printf("WHITE");
    case 'B' : printf("BLUE");
    default : printf("ERROR");
    break;
}
```

- What would be the output if choice = 'R'?
- (a) RED WHITE BLUE
  - (b) RED
  - (c) RED ERROR
  - (d) RED WHITE BLUE ERROR

**Q.40** Following program returns

```
void summation (n)
```

```
int n;
```

```
{
```

```
int i; sum = 0;
```

```
i = 1;
```

```
for (i = 1; i <= n, i++)
```

```
sum += i;
```

```
}
```

(a) sum of 1, 2, ..... , n

(b) nth number

(c) sum of the n number

(d) none of these

**Q.41** Consider the following program:

```
main ( )
```

```
{
```

```
int x = 2, y = 5;
```

```
if (x < y) return (x = x + y);
```

```
else printf("z1") ;
```

```
printf("z2") ;
```

```
}
```

Then

- (a) this will result in compilation error
- (b) output is z2
- (c) output is z1z2
- (d) none of these

**Q.42** The following program fragment:

```
int k = -7,
```

```
printf ("%d", 0 < !k);
```

(a) prints 0

(b) prints a non zero value

(c) is illegal

(d) prints an unpredictable value

**Q.43** # include<stdio.h>

```
#include<conio.h>
void main()
{
    int i;
    clrscr();
    i=1;
    while (i<=5)
    {
        print("%d",i);
        i++;
    }
}
```

- (a) 11145
- (b) 12145
- (c) 12345
- (d) none of these

**Q.44** # include<stdio.h>

```
#include<conio.h>
void main()
{
    int i;
    clrscr();
    i=1;
    while (i<=5)
    {
        printf("%d",i);
        if(i==3)
            continue;
        i++;
    }
}
```

- (a) 113...
- (b) 123...
- (c) 133...
- (d) none of these

**Q.45** Find out the error in the following program:

```
main()
{
    int mark;
    char grade;
    switch (mark)
    {
        case 5: grade = 'A'; break;
        case4: grade = 'B'; break;
        case4: grade = 'B' break;
        defualt: grade = 'C'; break;
    }/*switch*/
}
```

- (a) switch statement cannot have more than three labels
- (b) case labels cannot be numbers
- (c) no two labels may be identical
- (d) none of the above

**Q.46** Consider the following declaration:

```
switch (color)
{
    case 'r':
    case 'R': printf("RED");
                break;
    case 'g':
    case 'G': printf("GREEN");
                break;
    default: printf ("BLACK");
              break;
}
```

Which of the following statement is correct.

- (a) The above declaration is incorrect due to first case value
- (b) The above declaration is incorrect due to first and third case values
- (c) The above declaration performs only default condition whether input is r, R, g, G
- (d) The above declaration works properly for g, r, R, G and any other input value also.

**Q.47** The following statement:

```
if (a>b)
if(c>b)
printf("one");
else
if(c==a) printf("two");
else printf("there");
else printf("four");
(a) results in a syntax error
(b) prints four if c<=b
(c) prints two if c<=b
(d) prints four if a <=b.
```

**Q.48** What is the value of "average" after the following program is executed?

```
main ( )
```

```
{
```

```
int sum, index;
```

```
index = 0;
```

```
sum = 0;
```

```
for (;;) {
```

```
sum = sum + index;
```

```
++ index;
```

```
if (sum >= 100) break;
```

```
}
```

```
average = sum/index;
```

```
(a) 91/14
```

```
(b) 105/14
```

```
(c) 91/13
```

```
(d) 105/15
```

**Q.49** What will be the output of the following program?

```
{
int i = 1234; j = 0177, k = 0xa08c;
printf("%8d%8o%8x/n", i, j, k);
}
(a) 12340777ao80
(b) 00001234 01aa ax
(c) 1234 1 abc a08c
(d) 1234177 a08c
```

**Q.50** The O/P of following program will be

```
#include <stdio.h>
int main()
{
    int a = 10, b;
    a >= 5? b = 100 : b = 200;
    printf ("%d\n", b);
    return 0;
}
```

- (a) 100
- (b) 200
- (c) error
- (d) compile error

**Q.51** Which of the following is the correct output for the program given below?

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
char j = 1;
```

```
while (j <= 255)
```

```
{
```

```
printf ("%d", j);
```

```
j = j + 1;
```

```
}
```

```
printf ("\n");
```

```
return 0;
```

```
}
```

(a) 1 2 3 ... 127

(b) 1 2 3 ... 255

(c) 1 2 3 ... 254 255 0 1 2 3 ... 254 255... infinite times

(d) 1 2 3 ... 127 128 0 1 2 3 ... 127 128 ... infinite times

(e) 1 2 3 ... 127 -128 -127 -126 ... -2 -1 0 1 2 ... 127 -128 -127 ... infinite times

**Q.52** What will be the output of the following program?

```
#include <stdio.h>
{
    char ch;
    ch = 'A';
    printf ("The letter is");
    printf ("%c", ch >= 'A' && ch <= 'Z' ? ch
    + 'a' - 'A' : ch);
    printf ("Now the letter is");
    printf ("%c\n", ch >= 'A' && ch <= 'Z' ? ch
    + 'a' - 'A');
    return 0;
}
```

- (a) The letter is  
Now the letter is
- (b) The letter is a  
Now the letter is A.
- (c) Error
- (d) No output

**Q.53** Which of the following statements are correct about the program given below?

```
#include <stdio.h>
int main()
{
    unsigned int num;
    int i;
    scanf ("%u", &num);
    for (i = 0; i < 16; i++)
        printf ("%d", (num << i & 1 << 15) ? 1 : 0);
    printf ("\n");
    return 0;
}
```

- (a) It prints all even bits from num.
- (b) It prints all odd bits from num.
- (c) It prints binary equivalent of num.
- (d) None of the above

**Q.54** #include <stdio.h>

```
int function (int, int);
int main ()
{
    int i = 135, a = 135, k;
    k = function (!++i, !a++);
    printf ("i=%d a=%d k=%d\n", i, a, k);
    return 0;
}
```

int function (int j, int b)

```
{
    int c;
    c = j + b;
    return (c);
}
```

- (a) i = 136 a = 135 k = 271
- (b) i = 136 a = 136 k = 0
- (c) i = 136 a = 136 k = 272
- (d) None of the above

**Q.55** #include <stdio.h>

```
#define ISUPPER(x) (x>=65 && x<=90)
#define ISLOWER(x) (x>=97 && x<=122)
#define ISALPHA(x) (ISUPPER(x) ||
```

```
ISLOWER(x))

int main ( )
{
    char ch='+';
    if (ISALPHA(ch))
        printf("ch contains an alphabet\n");
    else
        printf("ch doesn't contain an alphabet\n");
    return 0;
}
```

- (a) ch contains an alphabet
- (b) ch doesn't contain an alphabet
- (c) error
- (d) no output

**Q.56** #include <stdio.h>  
 int main ()  
 {  
 char s[] = "C it yourself";  
 int i = 0;  
 while (s[i])  
 {  
 if (s[i] != '')  
 s[i] = s[i] + 1;  
 i++;  
 }  
 printf ("%s\n", s);  
 return 0;  
 }

- (a) C it yourself
- (b) B hs xntqrde
- (c) Error
- (d) D ju zpvstfng

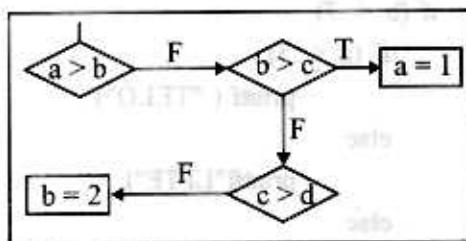
**Q.57** The following program fragment

```
int i = 5 ;
do {putchar (i + 100); print f("%d", i --);}
while (i) ;
results in the printing of
(a) i5h4g3f2el
(b) i4h3g2fle0
(c) an error message
(d) None of the above
```

**Q.58** What will be value of count after the following program is executed?

```
main ()
{
  int count, digit=0;
  count=1;
  while(digit<=9)
  {
    printf("%d", count);
    ++ digit;
  }
}
(a) 1001011101
(b) 1111111011
(c) 0111111111
(d) 1111111111
```

**Q.59** Consider the following flow chart.



Which of the following, correctly implements the above flow chart?

- (a) if (a > b)  
 if (b > c)  
 a = 1;  
 else if (c > d)  
 b = 2;
- (b) if (a <= b)  
 if (b > c)  
 a = 1;  
 else if (c <= d)  
 b = 2;
- (c) if (a > b);  
 else if (b > c)  
 a = 1;  
 else if (c <= d)  
 b = 2;
- (d) if (a > b);  
 else if (b > c)  
 a = 1;  
 else if (c > d);  
 else b = 2;

**Q.60** Consider the program fragment

```
j = 2;
while ((i % j) != 0)
  j = j + 1;
if (j < i) printf ("%d", j);
If i >= 2, then the value of j, will be printed only if
(a) i is prime
(b) j does not divide i
(c) j is odd
(d) i is not prime
```

**Q.61** The following program

```
main()
{
```

```

float a = .5, b = .7;
if (b < .7)
    if (a < .5)
        printf ("TELO");
    else
        printf("LTTE");
else
    printf ("JKLF");
}
outputs
(a) LTTE
(b) TELO
(c) JKLF
(d) PLO

```

**Q.62** Consider the following program.

```

main ( )
{ putchar ( 'M') ;
first ( );
putchar ( 'm') ;
first ( )
{ _____ }
second ( )
{ putchar ( 'd'); }

```

If Madam is the required output, then the body of first ( ) must be

- (a) empty
- (b) second ( ); putchar ( 'a') ;
- (c) putchar ( 'a'); second ( ); printf ("%c", 'a');
- (d) None of the above

**Q.63** #include <stdio.h>

```

#define a 10
main ( )
{ #define a 50
    print f ("%d", a);
}

```

The output will be

- (a) 10
- (b) Error
- (c) 50
- (d) Redef location # define not allowed within main.

## LEVEL-3

**Q.64** Trace the O/P

```

sum = 0;
for (i=1;i<=10;i++)
if (i%2==0) sum+=i;
printf("%d", sum);
(a) 40
(b) 45
(c) 0
(d) 30

```

**Q.65** Consider the following declaration

(i) for (j = 2, j <= 13; ++j)

```

{
    sum = 0;
    i = 2;
    while ( i < 100)
    {
        sum += i;
        i += j;
    }
    printf("%d", sum);
}

```

(ii) for (j = 2, j <=13; ++j)

```

{
    sum = 0;
    i = 2;
    do
    {
        sum += i;
        i += j;
    } while (i < 100);
    printf("%d", sum);
}

```

- (a) The (A) and (B) both finds the series  $2 + 4 + 6 + \dots + 98$
- (b) The (A) is find the sum of series  $2 + 4 + 6 + \dots + 98$  but (B) is not logically correct
- (c) The (A) is find the sum of series  $2 + 4 + 6 + \dots + 98$  and (B) is find the sum of series  $2 + 4 + 8 + \dots + 98$
- (d) The (A) and (B) performs the same operation of finding the sum of series  $2 + 4 + 8 + \dots + 98$

**Q.66** Trace the output

```
main ()
{ int i = 0, x = 0;
do {
    if (i%5==0)
        { x++;
    print f ("%d",x);
} ++i;
} while (i<20);
print (" %d",x)
}
(a) 1 2 3 4 5
(b) 1 2 3 5 7
(c) 1 2 3 4 4
(d) None of these
```

**Q.67** Trace the output

```
main ()
{
    int i, j, k, x = 0;
    for (i = 0; i < 5; ++i)
    {
        for (j = 0; j < i; ++j)
        {
            k = (i + j - 1);
            if (k%2 == 0)
                x += k;
            else
                if (k%3 == 0)
                    x += k + 2;
            printf("%d", x);
        }
        printf("%d", x); }
```

- (a) 0 0 2 4 5 9 10 14 14 20 20
- (b) 0 0 2 4 5 9 10 13 13 19 19
- (c) 0 0 2 4 5 7 9 11 13 15 15
- (d) 0 0 2 4 6 8 10 12 14 16 18
- (e) 0 0 2 2 4 9 13 18 22 22 28 28

**Q.68** Trace the output

```
main ()
{
    int i, j, k, x = 0;
    for (i = 0; i < 5; ++i)
        for (j = 0; j < i; ++j)
    {
        switch ( i + j - 1 )
        {
            case 0 : x += 1;
            case 1 :
            case 2 :
            case 3 : x += 2;
            default : x += 3;
        }
        printf("%d", x);
    }
}
```

- (a) 1 4 7 10 13 16 19 22 25 28 28
- (b) 1 6 11 16 21 24 29 32 35 38 41
- (c) 1 6 11 16 21 24 29 32 35 38 38
- (d) 1 4 7 12 15 20 23 28 31 36 41

**Q.69** Consider the following program fragment:

```
if (a>b)printf("a>b");
else printf("else part");
printf("a<=b");
a<=b will be printed if
(a) a>b
(b) a<b
(c) a==b
(d) none of the above.
```

**Q.70** Consider the following declaration

```
main( )
{
    int n, r, x = 0, a;
    do
    {
        scanf("%d", &n);
    }
    while (n <= 0);
    a = n;
    for ( ; a > 0;)
    {
        r = a % 10;
        x = x * 10 + r;
        a = a/10;
    }
    if (x == n)
}
```

The above code can be used for \_\_\_\_\_

- (a) finding palindrome number
- (b) finding the reverse of the number
- (c) finding Armstrong number
- (d) we cannot use due to errors

**Q.71** void main( )

```
{ int n, x, y;
printf("Enter n");
scanf("%d", &n);
for (x = 1; x <= n; x++)
{
    for (y = 1; y <= x; y++)
        printf("%d", y);
    printf("\n");
}
```

Due to above declaration \_\_\_\_\_

- (a) Due to stack overflow this program did not work.
- (b) Starting from 1 to n, each line contains 1 to digits which is equals to the number of line i.e. 1 is printed on first line, 1 2 is printed on second line and so on.
- (c) Starting from 1 to n digits are printed on one line and such a n lines are printed.
- (d) Starting from 1 to n digits are printed on n different lines.

**Q.72** # include <stdio.h>

```
#include<conio.h>
void main( )
{
    int i;
    clrscr();
    for(i=1;i<=5;i++)
    {
        if(i==3)
            break;
        printf("%d",i);
    }
}
```

- (a) 1245
- (b) 12345
- (c) 1234
- (d) 12

**Q.73** # include <stdio.h>

```
#include<conio.h>
void main()
{
    int i;
    clrscr();
    for (i=1 ;i<=5;i++)
    {
        if(i==3)
            continue ;
        printf("%d",i);
    }
}
```

- (a) 1145
- (b) 1245
- (c) 1345
- (d) 12345

**Q.74** Consider the following declaration:

```
void main ()
{
    int n, x, y, s = 40;
    scanf("%d", &n);
    for (x = 1, x <= 2 * n - 1; x += 2)
    {
        for (y = 1; y <= s; y++)
            printf(" ");
        for (y = 1; y <= x; y++)
            printf("%d", y);
        for (y = x - 1; y >= 1; y--)
            printf("%d", y);
        printf("\n");
        s = s - 4;
    }
}
```

The sum of all elements in the second row, which is printed by above code is (take  $n > 1$ )

- (a) 7
- (b) 9
- (c) 18
- (d) 4

**Q.75** #include<stdio.h>

```
#include<conio.h>
```

```
void main()
```

```
{
    int i;
    clrscr();
    for(i=1;i<=5;i++)
    {
        if(i==3)
            goto end;
        printf("%d",i);
    }
end:
```

- (a) 12345
- (b) 1234
- (c) 1245
- (d) 123

**Q.76** Consider a following declaration:

```
main ( )
{
    int i = 2, n = 3;
    if (i < 5)
    {
        for ( ; n <= 4; n++)
        {
            printf("%d", n);
        }
    }
}
```

considering the complexity for the condition is  $O(2)$

What will be the complexity for above code

- (a)  $O(4)$
- (b)  $O(2)$
- (c)  $O(6)$
- (d)  $O(5)$

**Q.77** #include <stdio.h>

```
#define P(format, var) printf("var=%format\n", var)
```

```
int main ()
```

```
{
    int i = 3;
    float a = 3.14;
    P(d, i);
    P(f, a);
    return 0;
}
```

- (a) Unexpected end of file in conditional error.
- (b) var = 0.000000 format
- (c) var = 3.140000 format
- (d) No output

**Q.78** The following program fragment:

```
int i=5;
do {
    putchar(i+100);
    printf("%d",i--),
}
```

while (i);  
results in the printing of

- (a) i5h4g3f2el
- (b) i4h3g2fle0
- (c) an error message
- (d) none of the above

### Common Data For Question 79 & 80:

Consider the following program fragment:

```
d = 0;
for (i = 1; i < 31; ++i)
    for (j = 1; j < 31; ++j)
        for(k = 1; k < 31; ++k)
            if(((i + j + k) % 3) == 0)
                d=d+1;
printf("%d", d);
```

**Q.79** The output will be

- (a) 9000
- (b) 27000
- (c) 3000
- (d) none of the above

**Q.80** The number of additions performed by the above program fragment is

- (a) 27000
- (b)  $27000 \times 3$
- (c)  $9000 + 3 \times 27000$
- (d)  $9930 + 27000 \times 3$

## GATE QUESTIONS

**Q.81** An unrestricted use of the “go to” statement is harmful because of which of the following reason (s): [GATE 1989]

- (a) It makes it more difficult to verify programs.
- (b) It makes program more inefficient.
- (c) It makes it more difficult to modify existing programs.
- (d) It results in the compiler generating longer machine code.

**Q.82** Given the programming constructs (i) assignment (ii) for loops where the loop parameter cannot be changed within the loop (iii) If then else (iv) forwards goto (v) arbitrary goto (vi) non recursive procedure call (vii) recursive procedure/function call (viii) repeat loop, which constructs will you not include in a programming language such that it should be possible to program the terminates (i.e. halting) function in the same programming language. [GATE 1999]

[2-Marks]

- (a) (ii),(iii),(iv)
- (b) (v),(vii),(viii)
- (c) (vi),(vii),(viii)
- (d) (iii),(vii),(viii)

**Q.83** Consider the following C function definition

```
int Trial (int a, int b, int c)
{
    if ((a >= b) && (c < b)) return b;
    else if (a >= b) return Trial (a,c,b)
    else return Trial (b,a,c)
}
```

The function Trial:

[GATE 1999]

[2-Marks]

- (a) finds the maximum of a,b and c
- (b) finds the minimum of a,b and c
- (c) finds the middle number of a,b,c
- (d) none of the above

**Q.84** In the following C program fragment j, k n and TwoLog\_n are integer variables, and A is an array of integers. The variable n is initialized to an integer  $\geq 3$ , and TwoLog\_n is initialized to the value of  $2^{\lceil \log_2(n) \rceil}$

```
for (k = 3; k <= n; k++)
    A[k] = 0;
for (k = 2; k <= TwoLog_n; k++)
    for (j = k + 1; j <= n; j++)
        A[j] = A[j] || (j % k);
for (j = 3; j <= n; j++)
    if (!A[j]) printf("%d", j);
```

The set of numbers printed by this program fragment is [GATE 2003]

[2-Marks]

- (a) {}
- (b) {m | m  $\leq$  n, ( $\exists i$ ) [m = i!]}
- (c) {m | m  $\leq$  n, ( $\exists i$ ) [m =  $i^2$ ]}
- (d) {m | m  $\leq$  n, m is prime}

- Q.85** What does the following algorithm approximate? (Assume  $m > 1$ ,  $\epsilon > 0$ ).

```

x = m ;
y = 1 ;
while (x - y > ε) {
    {      x = (x + y)/2;
          y = m/x;
    }
}
print(x);

```

[GATE 2004]  
[2-Marks]

- (a)  $m^{1/3}$
- (b)  $m^{1/2}$
- (c)  $\log m$
- (d)  $m^2$

- Q.86** Consider the following program fragment for reversing the digits in a given integer to obtain a new integer. Let  $n = d_1d_2 \dots d_m$

```

int n, rev ;
rev = 0 ;
while (n > 0) {
    rev = rev * 10 + n % 10 ;
    n = n / 10 ;
}

```

The loop invariant condition at the end of the  $i^{\text{th}}$  iteration is [GATE 2004]

[2-Marks]

- (a)  $n = d_1 d_2 \dots d_m$  or  $rev = d_m \dots d_2 d_1$
- (b)  $n \neq rev$
- (c)  $n = d_{m-i+1} \dots d_{m-1} d_m$  or  $rev = d_{m-i} \dots d_2 d_1$
- (d)  $n = d_1 d_2 \dots d_{m-i}$  and  $rev = d_m d_{m-1} \dots d_{m-i+1}$

- Q.87** Consider the following C function

```

int f(int n)
{ static int i = 1 ;
  if (n >= 5) return n ;
  n = n + i ;
  i++;
  return f(n) ;
}

```

The value of returned by  $f(1)$  is [GATE 2004]  
[2-Marks]

- (a) 6
- (b) 8
- (c) 7
- (d) 5

- Q.88** Consider the following C program

```

main ( )
{
    int x, y, m, n;
    scanf("%d%d", &x, &y);
    /* Assume x > 0 and y > 0 */
    m = x;           n = y;
    while (m != n)
    {
        if (m > n)
            m = m - n;
        else
            n = n - m;
    }
}

```

printf("%d", n);

}

The program computes [GATE 2004]  
[2-Marks]

- (a) the least common multiple of  $x$  and  $y$
- (b) the greatest common divisor of  $x$  and  $y$
- (c)  $x \div y$ , using repeated subtraction
- (d)  $x \bmod y$  using repeated subtraction

- Q.89** What is the output of the following program?

```

#include <stdio.h>
int funcf(int x);
int funcg(int y);
main ( )
{
    int x = 5, y = 10, count;
    for (count = 1, count <= 2; ++count){
        y += funcf(x) + funcg(x);
        printf("%d", y);
    }
}
funcf(int x){
    int y;
    y = funcg(x);
    return(y);
}

```

```
funcg(int x){  
    static int y = 10;  
    y+=1;  
    return(y + x);  
}
```

- (a) 43 80  
 (b) 42 74  
 (c) 33 37  
 (d) 32 32

**Q.90** Consider the following C-program

```
void foo (int n, int sum) {  
    int k = 0, j = 0;  
    if (n == 0) return;  
    k = n % 10; j = n/10;  
    sum = sum + k;  
    foo (j, sum);  
    printf("%d", k);  
}  
  
int main () {  
    int a = 2048, sum = 0;  
    foo (a, sum);  
    printf("%d/n", sum);  
}
```

What does the above program print?

[GATE 2005]

[2-Marks]

- (a) 2, 0, 4, 8, 0  
 (b) 8, 4, 0, 2, 0  
 (c) 2, 0, 4, 8, 14  
 (d) 8, 4, 0, 2, 14

**Q.91** Consider the following C-function in which  $a[n]$  and  $b[m]$  are two sorted integer arrays and  $c[n+m]$  be another integer array.

```
void xyz (int a[ ], int b[ ], int c[ ]) {  
    int i, j, k;  
    i = j = k = 0;  
    while ((i < n))&&(j < m)  
        if (a[i] < b[j]) c[k++] = a[i++];  
        else c[k++] = b[j++];  
    }
```

Which of the following condition(s) hold(s) after the termination of the while loop?

[GATE 2006]

[2-Marks]

- (i)  $j < m$ ,  $k = n + j - 1$ , and  $a[n - 1] < b[j]$  if  $i = n$   
 (ii)  $i < n$ ,  $k = m + i - 1$ , and  $b[m - 1] \leq a[i]$  if  $j = m$   
 (a) neither (i) nor (ii)  
 (b) either (i) or (ii) but not both  
 (c) only (i)  
 (d) only (ii)

**Q.92** Consider the following segment of C-code

```
int j, n;  
j = 1;  
while (j <= n)  
    j = j*2;
```

The number of comparisons made in the execution of the loop for any  $n \geq 0$  is

[GATE 2007]

[1-Mark]

- (a)  $\lfloor \log_2 n \rfloor + 1$   
 (b)  $n$   
 (c)  $\lceil \log_2 n \rceil$   
 (d)  $\lceil \log_2 n \rceil + 1$

# ANSWER KEY

1	a	2	b	3	d	4	b	5	a
6	a	7	a	8	c	9	c	10	b, c
11	b	12	d	13	b, c, d	14	b	15	a
16	a, b, c	17	c	18	d	19	b	20	all
21	a, b	22	c	23	c	24	a	25	a
26	d	27	d	28	a	29	c	30	b
31	b	32	d	33	b	34	c	35	d
36	c	37	b	38	d	39	d	40	a
41	d	42	a	43	c	44	b	45	c
46	d	47	d	48	d	49	d	50	b
51	e	52	b	53	c	54	b	55	b
56	d	57	a	58	d	59	b, c, d	60	d
61	a	62	c	63	c	64	d	65	a
66	c	67	e	68	c	69	a, b, c	70	a
71	b	72	d	73	b	74	b	75	c
76	b	77	c	78	b	79	a	80	d
81	a, b	82	b	83	c	84	a	85	b
86	d	87	c	88	b	89	a	90	a
91	b	92	a						

## SOLUTIONS

**S.1 (a)**

We can perform (i), (ii), (iii) with the help of "goto" statement which is actually an unconditional branching statement.

**S.6 (a)**

$x = 10, y = 10, z=5$   
 $\Rightarrow (x < y) < z = 0 < z = \text{True}$

As it is left to right associativity.

**S.7 (a)**

```
i/p: 5,2,0,3,0
if (5>0) => print 5
sum = 0+5=5
if (2>0) => print 2
sum = 5+2 = 7
```

**S.8 (c)**

1st iteration:

$$i = 1 + 3 = 4$$

2nd iteration:

$$i = 4 + 3 = 7$$

Condition false. Thus out of loop prints 7.

**S.9 (c)**

Neither if condition nor else if condition is true hence it will print the else print.

**S.11 (b)**

Constant expression like  $1 * 2 + 4$  are acceptable in cases of a switch.

**S.12 (d)**

```
int a = 1, b, dummy;
a > 10? b = 20: dummy = 1;
```

Note that the following would not have worked:

```
a > 10? b = 20 : ;;
```

**S.13 (b, c, d)**

Comment starting with /\* and must end with \*/ combination

**S.14 (b)**

The input is actually a\nb. Since we are reading only two characters, **only a and \n will be read and printed.**

**S.17 (c)**

$y *= 2$  means  $y = y * 2$  i.e.,  $y = 4$ , in this problem. So, the expression is equivalent to  $x = 4 + 4$ , which is 8. So, **8 will be printed**. However, the order in which the operands are evaluated is implementation-dependent. If the right operand is evaluated first, the result will be 6. Don't take things for granted.

**S.18 (d)**

$9/5$  yields integer 1. Printing 1 as a floating point number prints garbage.

**S.19 (b)**

Output

66 A

67 B

...

125 ]

126 }

...

255

256

**Explanation:**

The **for** loop begins with 65 and goes up till 255, printing each of these numbers along with their corresponding characters. But note that the arguments are passed to the **printf()** function from right to left. Hence the second **ch** is passed

first, then it is incremented and then the first **ch** is passed. Hence when the first **ch** is passed, by that time value of **ch** already stands incremented.

Since **ch** has been declared as an **unsigned int** there is no question of exceeding the range, as the range of an **unsigned int** is 0 to 4294967295.

**S.23 (c)**

i % j	i = i/j	j = j + 1
true	1680	5
true	336	6
true	56	7
true	8	8
false	1	9

**S.24 (a)**

"++i" provides pre-increment, i.e.  $i = 6$ , then  $(++i)/(i++)$  provide  $i = 1$  with post increment resulting  $i = 2$ .

**S.26 (d)**

First form the truth table of the exclusive OR operation. If both the switches are off i.e., 0, 0 then the light will be off i.e., 0. So, 0, 0 yields 0. If you switch on either of the two switches i.e., 0 1 or 1 or 1 0, the light will be on. So, 0 1 yields 1 (so does 1 0). Now, if you switch on the other one, which is currently off, it will be 1 1. This should yield a 0. Compare these results with the truth table of **XOR**.

**S.27 (d)**

Hexadecimal representation is 4, i.e., m can have 4 possible values.

**S.28 (a)**

$$(a - a/b * b \% c) > a \% b \% c$$

$$(9 - 9/5 * 5 \% 3) > 9 \% 5 \% 3$$

which is true.

**S.29 (c)**

The condition in the while loop is a must, the while loop cannot take if condition.

**S.30 (b)**

All statements (i), (ii) & (iii) are correct statements.

**S.31 (b)**

The output printed by above code is

1 1 1 1 1 1 1 1 1 1

2 2 2 2 2 2 2 2 2

3 3 3 3 3 3 3 3

4 4 4 4 4 4 4 4       $\Rightarrow 28$  (4<sup>th</sup> row sum)

5 5 5 5 5 5

6 6 6 6 6

7 7 7 7       $\Rightarrow 28$  (7<sup>th</sup> row sum)

8 8 8

9 9

10

**Sum (4<sup>th</sup> row) – Sum (7<sup>th</sup> row) = 0**

**S.32 (d)**

(a)  $\Rightarrow$  we can replace count as count ++ in output statement.

(b)  $\Rightarrow$  while is faster than do-while.

(c)  $\Rightarrow$  due to do-while program will run atleast once.

**S.33 (b)**

Sum of squares of first 10 numbers = 385

**S.35 (d)**

After each pass, n is incremented by 2.

$\therefore$  sum = 1 + 9 + 25 + 49 + 81

$1234 + 33 + 1434 + 778 + 1203 = 4682$

**S.36 (c)**

1<sup>st</sup> iteration i= 1 i = 10

2<sup>nd</sup> iteration i=2 i = 9

3<sup>rd</sup> iteration 3 8

4<sup>th</sup> iteration 4 7

5<sup>th</sup> iteration 5 6

**S.37 (b)**

1.	Type declaration instructions	d.	char name, code;
2.	Input/Output instructions	c.	gets( );
3.	Arithmetic instructions	a.	$k = i * 234 + n - 7;$
4.	Control instruction	b.	for(i=0;i<=10;i++);

**S.38 (d)**

Initially the condition n<10 is false. Hence printf will not be executed.

**S.39 (d)**

If choice = 'R' then the case 'R' is matched and all the case will be executed including default as there is no break between any cases.

**S.40 (a)**

The statement sum + i produces the sum of n number starting from 1 to n.

**S.41 (d)**

Return always terminates the function that executed it. main() being a function, will be terminated when it executes the return statement. The return value will be returned to the calling environment, which is the operating system in this case.

**S.42 (c)**

k=7. So if 'k' is used as a boolean variable will be treated as a true condition. So, !k will be false i.e., so, 0<!k is actually 0<0, which is false. So, 0 will be printed.

**S.46 (c)**

The above declaration works properly for r, R,g,G and any other input value also.

Case 1: Suppose input value is other than r,R,g and G then it will execute default.

Case 2 If input value is no statements corresponding to 'r' and also there is no 'break' statements.

Hence it will execute case R: printf ("RED")

Similar for case 'g' :

Case 3: If the input value is 'R' or 'G' it will execute corresponding statements.

### S.50 (b)

The second assignment should be written in parentheses as follows:

$a \geq 5? b = 100 : (b = 200);$

else always second assignment gets executed.

### S.51 (e)

variable 'j' is a character type. Hence, value of 'j' will be printed upto 127. After that for  $j = 128$ , value of character becomes negative i.e. -128 - 127 ..... 1 0 1 2 and so on, for while loop upto 255.

### S.54 (b)

Output

i = 136 a = 136 k = 0

**Explanation:**

Observe the function call in **main()**. Since **++** precedes **i** its value is incremented to 136, and then the **!** operator negates it to give 0. This 0 is however not stored in **i** but is passed to **function()**. As against this while evaluating the expression **!a++**, since **++** follows **a**, firstly **a** is negated to 0, this 0 is passed to **function()** and **a** is incremented to 136. Thus what get passed to **function()** are 0 and 0, which are collected in **j** and **b**, added to give another 0 and finally returned to **main()**, where it is collected in **k** and then printed out.

### S.55 (b)

Output

ch doesn't contain an alphabet

**Explanation:**

The first and second macros have the criteria for checking whether their argument **x** is an upper or a lower case alphabet. These two criteria have been combined in the third macro,

**ISALPHA**. Thus when the program goes for compilation, the **if** statement has been converted to the form:

```
if (ch >= 65 && ch <= 90 || ch >= 97 && ch <= 122)
```

As **ch** has been initialized to character '+' , the conditions in the **if** fail and the control rightly passes to the **else** block, from where the output is obtained.

### S.56 (d)

Output

D ju zpvstfmg

**Explanation:**

No, your computer hasn't caught a virus! It has done just what you instructed it to. The **while** loop tests the value of the **i** element of the string. Since **i** has been initialised to 0, the first time **s[0]**, i.e. C is used. Since 'C' has a non-zero ASCII value, the condition evaluates to true, and control passes to the **if** statement. The condition to be satisfied here is that the **i** element is not a blank space. **s[0]** satisfied this condition hence the contents of the 0 elements are incremented by 1. Thus **s[0]**, i.e. 'C' having ASCII value 67, is incremented to 68, which is the ASCII value of upper case 'D'. The new value of **s[0]** is therefore 'D'. Next **i** is incremented to 1, and the while repeats for **s[1]**. However, the if condition fails this time, as **s[1]** is a blank space, so this element remains unchanged. Similarly, all non-blank elements are incremented, and the while ends when the '\0' is reached. lastly, the **printf()** outputs the changed string.

### S.57 (a)

**putchar** (105) will print the ASCII equivalent of 105 i.e., 'l'. The **printf** statement prints the current value of **i**, i.e., 5 and then decrements it. So, h4 will be printed in the next pass. This continues until 'l' becomes 0, at which point the loop gets terminated.

### S.58 (d)

The value of count is one greater than the variable digit.

**CONTROL STATEMENTS**

Hence when the last value of digit (digit $\leq 9$ ) is then the value of digit inside the loop will be 10. Hence the value of count will be one greater than count i.e. 11.

**S.62 (c)**

Since Madam is the required output, the function first () , should print 'a', call the function second ( ) that prints the 'd' and print 'a' again.

**S.63 (c)**

The preprocessor directive can be redefined anywhere in the program. So the most recently assigned value will be taken.

**S.64 (d)**

```
sum = 0
for (i=1;i<=10;i++)
    if(i%2==0) → false
    for (i = 2; i <= 10; i++)
        if(2%2 == 0) → True
        sum = 0 + 2 = 2
    for (i=3;3<=10;i++)
        if (3%2==0) =false

```

that means, we are adding only even values to sum

$$\therefore \text{sum} = 2+4+6+8+10=30$$

**S.65 (a)**

when  $j = 2$

sum = 0

$i = 2 < 100$

sum =  $0+2=2 \rightarrow$

$i = i+j = 2+2 = 4$

printf ("%d",sum) → 2

when  $j = 3$

sum =  $2 + 4$

$i = 2 + 4 = 6$

when  $j = 4$

$s = 2 + 4 + 6 + \dots$  Like that we find the series

$$2 + 4 + 6 + \dots + 98.$$

**S.66 (c)**

initially  $i = 0, x = 0; \rightarrow i > x$

$$0 \% 5 == 0$$

$$\therefore x = 1$$

then  $i$  is incremented to 1

$$\text{when } i = 5$$

$$5 \% 5 == 0$$

$$\therefore x = 2$$

when  $i$  is incremented to 10

$$10 \% 5 == 0$$

$$\therefore x = 3$$

when  $i$  is incremented to 19

$$19 < 20$$

$$\text{but } 19 \% 5 \Rightarrow 0$$

$$\therefore x = 4, \text{ only}$$

when  $i = 20$  condition in while is false

$\therefore$  control come out of loop,

$$x = 4$$

$\therefore \text{Final output} = 1 \ 2 \ 3 \ 4 \ 4$

**S.68 (c)**

initially  $i = 0$  and  $i < 5$

$$j = 0 \quad 0 < 0$$

Now  $i = 1, \quad i < 5$

$j = 0$  and  $0 < 1$

{

switch ( $0 + 1 - 1$ )

case 0 :  $x = 0 + 1 = 1$

$$\therefore x = 1$$

}

Now  $j = 1$  and  $1 < 1$

Now,  $i = 2 \quad 2 < 5$

$j = 0, \text{ and } 0 < 2$

{

switch ( $2 + 0 - 1$ )

case 1 :

```

case 2:          (d) 88.2
case 3: x = 1 + 2 = 3
∴ x = 3
default : x = 3 + 3
x = 6

```

Similarly final o/p as

1 6 11 16 21 24 29 32 35 38 38

### S.69 (a, b, c)

The else clause has no brackets i.e., element. So, printf ("a<=b"); will be executed anyway if  $a > b$  or  $a <= b$ . Hence the answer.

### S.70 (a)

If we enter number less than 0, then it won't accept, we have to enter number greater than zero.

In for loop; we are finding mod of that number

Let's assume,  $n = 121$ ,  $a = n = 121$

- 1)  $r = a \% 10 = 121 \% 10 = 1$
- 2)  $x = x * 10 + r = 0 * 10 + r = 1$
- 3)  $a = 121/10 = 12$   
 $a = 12$
- 1)  $r = 12 \% 10 = 2$
- 2)  $x = 1 * 10 + 2 = 12$
- 3)  $a = 12/10 = 1$   
 $a = 1$
- 1)  $r = 1 \% 10$
- 2)  $x = 12 * 10 + 1 = 121$
- 3)  $a = 1/10 \Rightarrow$  condition false

and finally we are comparing,  $(x == n) \Rightarrow (121 == 121)$

i.e. we can use the code for finding palindrome number.

### S.71 (b)

Let's assume that  $n = 2$

In for loop,  $(x = 1; x \leq 2; x++) \Rightarrow$  True

In inner for loop ( $y = 1; y \leq x; y++$ )  $\Rightarrow$  True

print  $y = 1 \Rightarrow 1$

$\text{print} ("\\n") \Rightarrow$  control is transfer to new line  
then  $y = y + 1 = 2 \Rightarrow$  Inner for loop condition false therefore, control is transfer to  $x = x + 1 \Rightarrow x = 2$ .

then also condition is true, again control goes to inner for loop and 1 2 is printed on second line again condition becomes false and finally outer for loop condition become false and control comes out. Therefore, we are getting output as

1	2	3	2	1												
1	2	3	4	3	2	1										
1	2	3	4	5	6	7	6	5	4	3	2	1				
1	2	3	4	5	6	7	8	9	8	7	6	5	4	3	2	1
:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:

### S.74 (b)

Due to given code, we obtain a output as follows

1	2	3	2	1												
1	2	3	4	5	4	3	2	1								
1	2	3	4	5	6	7	6	5	4	3	2	1				
1	2	3	4	5	6	7	8	9	8	7	6	5	4	3	2	1

From this we can find the, sum of the second row as  $= 1 + 2 + 3 + 2 + 1 = 9$

### S.76 (b)

For condition checking by default,  $O(1)$  is required

$\text{if } (i < 5) \Rightarrow \text{True} \Rightarrow O(2)$

$\text{for } ( ; n \leq 4) \Rightarrow \text{True} \Rightarrow O(2)$

$\therefore O(n) = O(2) + O(2) + O(1) + O(1) = O(2)$   
 $\Rightarrow$  By sum Rule.

### S.77 (c)

Output

`var = 0.000000format`

`var = 3.140000format`

Explanation

During preprocessing, the `format` in the `printf()` statement does not get replaced by the argument `d`. Thus the `printf()` statements, after preprocessing, look like this:

```
printf ("var = %format\n", i);
printf ("var = %format\n", a);
```

This is only in keeping with what the previous example illustrated, that the macro template within the quotes doesn't get replaced by the macro expansion. When these `printf()`s are executed, the material within quotes get dumped on to the screen as it is, except when a format specification or an escape sequence (like '`\n`') is encountered.

Look at the first output. Where did 'f' go? And why were the numbers printed at all? This has the simple explanation that our argument `format` happened to have as its first letter, an 'f'. The `printf()` interpreted '%f' as the format specification, and 'ormat' as something we wanted to write on the screen literally.

Note that we got the expected value for `a`, a `float`, but an absurd one for `int i`, since the `printf()` attempted to print out an `int` using `%f`.

### S.78 (b)

`Putchar (105)` will print the ASCII equivalent of 105 i.e., 'i'. The `printf` statement prints the correct value of `i`, i.e. 5 and then decrements it. Some `h4` will be printed in the next pass. This continues until 'i' becomes 0, at which point the loop gets terminated.

### S.79 (a)

$a + b + c \% 3$  will be 0 if  $a + b + c$  is a multiple of 3. This will happen in one of the following ways. All three -`a`, `b`, and `c` are multiples of 3. This can only happen if `a`, `b`, and `c` take one of the 10 values, -3, 6, 9, ..., 30, independent of one another. So, there are  $10 \times 10 \times 10 = 1000$  ways this can happen. Another possibility is that `a`, `b`, and `c` all leave a remainder 1 so that  $a + b + c$  is evenly divisible by 3. Considering all the different possibilities and adding, we get 9000. That will be the integer that gets printed.

### S.80 (d)

#### Refer S.79

The result can be analytically reasoned out. It can also be programmatically verified by having an integer variable 'countAddition' (initialized to 0) and incrementing this variable each time an addition is performed. With these changes the program fragment looks like.

```
int countAddition = 0;
d = 0;
for (i = 1; i < 31; ++i, ++countAddition) // To account for the addition in ++i
for (j = 1; j < 31; ++j, ++countAddition) // To account for the addition in ++j
for (k = 1, k < 31; ++k, ++countAddition) // To account for the addition in ++k
if (((i + j + k) % 3) == 0)
{
    d = d + 1;
    ++countAddition; // To account for the addition in d = d + 1
    ++countAddition; // To account for the addition in i + j
    ++countAddition; // To account for the addition in j + k
}
else
{
    ++countAddition; // To account for the addition in i + j
    ++countAddition; // To account for the addition in j + k
}
printf("%d", d);
printf("\n%d", countAddition);
```

The value of the variable `countAddition` that is printed by the last statement is the answer.

**S.81 (a, b)**

Disadvantages of go to statement.

**S.82 (b)**

An arbitrary goto can lead to infinite loops as follows:

Label 1: GOTO Label 2;

Label2: GOTO Label 1;

**Recursive procedure/ functional calls may be non terminating** if the terminating condition is never satisfied. A repeat loop can also lead to infinite loops if the condition to exit the loops is never satisfied.

**S.83 (c)**

In this, we are comparing ( $a \geq b$ ) & ( $c < b$ ), if both are true then only we return b, that means we are finding middle number of a,b,c. Again by calling Trial function with different parameters, we are finding middle number of a,b,c.

**S.84 (a)**

If( $\neg A[j]$ ) condition

If( $A[j] == 0$ )

Prints the value. But no zero is the array.

**S.85 (b)**

Let  $x = m = 9$ . The loop will be terminated when  $x - y = 0$  or  $x - y < 0$ . Consider the following iteration for  $x = m = 9$ ,  $y = 1$

$x-y > 0$	$x = (x+y)/2$	$y = m/x$
9-1=8	$x=(9+1)/2 = 5.0$	$y = 9/5.0 = 1.8$
5.0-1.8=3.2	$x=(5.0+1.8)/2=3.4$	$y=9/3.4=2.6$
3.4-2.6>0	$x=(3.4+2.6)/2=3$	$y=9/3.0=3.0$

$3.0 - 3.0 = 0$  loop terminated

So,  $m = 9$  then  $x = 3$

$$(m)^{1/2} = (3)^{1/2}$$

$$m = 9$$

$\Rightarrow x = 3$

So the algorithm compute  $m^{1/2}$ .

**(b) 08.2 S.86 (d)**

Loop invariant is the part of code motion. Loop invariant for while loop is a condition, if we assign this condition before the while loop then there is no effect in the code and produces same output

Consider the given code

int n, rev;

rev = 0;

while ( n > 0 ) {

    rev = rev \* 10 + n%10;

    n = n/10;

}

Let input  $n = 12345$

where  $d_1 = 1$ ,  $d_2 = 2$ ,  $d_3 = 3$ ,  $d_4 = 4$  and  $d_5 = 5$

Iteration	$n = n/10$	$rev = rev * 10 + n \% 10$
0	$d_1 d_2 d_3 d_4 d_5$ 1 2 3 4 5	rev = 0
1	$d_1 d_2 d_3 d_4$ 1 2 3 4	$rev = 0 * 10 + 5 = 5$
2	$d_1 d_2 d_3$ 1 2 3	$rev = 5 * 10 + 4 = 54$
3	$d_1 d_2$ 1 2	$rev = 54 * 10 + 3 = 543$
4	$d_1$ 1	$rev = 543 * 10 + 2 = 5432$
5	0	$rev = 5432 * 10 + 5 = 54325$

Let  $n = d_1 d_2 \dots d_m$  in the  $i^{\text{th}}$  iteration  $rev = d_m d_{m-1} \dots d_{i+1}$

For example if  $m = 5$  and  $i = 3$  the above example produces

$n = d_1 d_5 \dots d_2$        $rev = d_5 d_4 d_5 \dots d_1$   
1 2                            5 4 3

**S.87 (c)**

1. int f(int n)

2. { static int i = 1 ;