Stack

IT Job Solution

(A Pattern Based IT Job Solution)

Computer Science
and
Information Technology

"Work Hard in Silence and Let Success Makes the Noise"

Pattern Analysis

Pattern Name	Unit for the Pattern	Remarks
BUET	Unit-0,1,2,3,4,5,6,7,8,9,10,11,12,13,16,17	DESCO, DPDC, PGCB, DTCA, NWPGCL, TGTDCL, DBBL etc. (Power and Gas Field)
KUET, RUET	Unit-0,1,2,3,4,7,8,9,10,11,12,13	NESCO, WZPDCL
MIST	Unit-0,1,2,3,4,7,8,9,10,11,12,13	DPDC
BREB(Own)	0,1,2,3,4,7,8,9,10,11,12,13	BREB
BPSC, NTRC	Unit-0,1,2,3,4,5,6,7,8,9,10,11,12,13,16,17	BPSC
Bank	Unit-0,1,2,3,4,5,6,7,8,9,10,11,12,13,16,17	Combined Bank
Bangladesh Bank	Unit-0,1,2,3,4,5,6,7,8,9,10,11,12,13,16,17	Bangladesh Bank
IBA	Unit-0,1,2,3,4,5,6,7,8,9,10,11,12,13,16,17	Tele talk, BCIC
Telecom Sector	Unit-14,15,18,19	BTCL, Tele talk
Combined with Electrical	Unit- 14,15	BTCL, BTRC etc.

গুরুত্বপূর্ন পয়েন্ট

- উপরের প্যাটানগুলোর মধ্যে কম্পিউটার সাইন্সের বেশিরভাগ পরীক্ষা বুয়েট নিয়ে থাকে, যেমন-ডেসকো, ডিপিডিসি, পিজিসিবি ইত্যাদি। তাই প্রস্তুতি শুরু করতে হবে বুয়েট প্যাটানে, আর বুয়েট প্যাটানে সবচেয়ে বেশি বিষয় থেকে প্রশ্ন করা হয়। তাই বুয়েট প্যাটানে প্রস্তুতি নিলেই অন্য সকল প্যাটান পড়া হয়ে যায়। একটা কথা বলে রাখা ভালো বুয়েট প্যাটানে লজিক্যাল প্রশ্ন বেশি করা হয়, অন্যদিকে কুয়েট বা রুয়েট প্যাটানে(যেমন-নেসকো, ওজোপাডিকো) থিওরি প্রশ্ন বেশি করা হয়।
- টেলিকম সেক্টরের জবের ক্ষেত্রে (বিটিআরসি, বিটিসিএল, টেলিটক) পরীক্ষা যে প্যাটানে হোক না কেন টেলিকম সেক্টরের জন্য অতিরিক্ত কিছু ইউনিট এই বইতে দেওয়া হয়েছে সেগুলো অবশ্যই পড়তে হবে।
- আইবিএ প্যাটান অন্য সকল প্যাটান থেকে তুলনামূলকভাবে কঠিন হয়, তাই উল্লিখিত ইউনিটগুলো ভালো করে বুঝে বুঝে পড়তে হবে।
- পিএসসি এর ক্ষেত্রে থিওরিটিক্যাল প্রশ্ন বেশি থাকে। তাই এই কথা মাথায় রেখে প্রস্তুতি নিতে হবে।
- যেসকল পরীক্ষা ইলেকট্রিক্যাল কিংবা ইলেকট্রনিক্স এর কম্বাইন্ড হয় (যেমন-বিটিসিএল, বিটিআরসি) সেসকল পরীক্ষার জন্য অতিরিক্ত হিসেবে উলিক্ষিত ইউনিটগুলো পড়তে হবে সাথে যারা পরীক্ষা নিবে তাদের প্যাটান অনুসরণ করতে হবে।

Contents

Programming and Data Structures and Algorithms

UNIT 1

PART A

Chapter 1

Programming in C

LEARNING OBJECTIVES

- *Basic concepts*
- *Character set*
- *Identifier*
- *Declaring a variable*
- *Visualization of declaration*
- *Constants*
- *Single character constants*
- *String constants*
- *Using const keyword*
- *Precedence decreases as we move from top to bottom*
- *Type conversion*
- *Documentation section*
- *Preprocessing*
- *Global declaration*
- *Control statements*
- *Selection/Decision making statement*
- *Looping statements*
- *Unconditional jump statements*

Basic Concepts

Character Set

A character refers to an alphabet, digit or a special symbol.
Alphabets: $A - Z$, $a - z$

Digits: 0 -9
Special symbols:
~ ! # % ∧ and * ( ) – + { } [ ] – < > , . | ? \ | : ; " ' White space

Identifi er

Identifi er is a user-define d name used for naming a variable or a function.
Rules for naming an identifier

- Consists only letters, digits and underscore
- Starts only with an alphabet or underscore
- Keywords cannot be used.
- Can be as long as you like, first 31 characters are significant.

Example: Valid identifiers: RollNo, Roll_No, _Roll_No
rollno, Name2;
Invalid: 2name, Roll No.

Variable

The name itself represents value, is not constant. Variable is a data name whose value varies/changes during program execution. Variable name is a name given to memory cell (may be one or multiple bytes).

Data Types

Represents type of data and set of operations to perform on data .

Data Type

Primitive/Basic	Derived	User defined	Valueless
– Char	– Array	– Structure	
– float	– pointer	– union	– void
– double		Enumeration	
– integer			

Type	Keyword	Number of Bytes
Integer	int	2
Floating	float	4
Double	double	8
Character	char	1

- Before using a variable, you must give some information to compiler about the variable. i.e., you must declare it.
- Declaration statement includes the type and variable name.

Syntax:
Datatype Var_name;
Example:

```
int roll_no;
char ch;
float age;
```

- When we declare a variable
  - memory space is allocated to hold a value of specified type.
  - space is associated with variable name
  - space is associated with a unique Address.

Table 1 *Visualization of declaration*

	roll no
int roll no;	garbage
	2002
	marks
int marks = 10;	10
	3008
	diameter
float diameter = 5.9	5.9
	4252
	ch → variable name
char ch : 'A'	A → value
	2820 → address

Note: The default value is garbage, i.e., an unknown value is assigned randomly.

Renaming data types with typedef Typedef is a keyword, which can form complex types from the basic type, and will assign some simpler names for such combinations. This is more helpful when some declaration is very tough, confus-ing or varies from one implementation to another.

For example, the data type unsigned long int is redefined as LONG as follows:

typedef unsigned long int LONG;

Uses of enumerated data types Enumerated data types are most useful when one is working over small, discrete set of values, in which each is having a meaning and it is not a number.

A best example can be given on months jan, feb, mar, …, dec, which are 12 in number, with assigning consecutive num-bers for it.

The main advantages are storage efficiency, the *c*-code can become readable

Constants

A constant value is one which does not change during the execution of a program.
C supports several types of constants:

1. Integer constants
2. Real constants
3. Single character constants
4. Strings constants

Integer constants

An integer constant is a sequence of digits. It consists of a set of digits 0 to 9 preceded by an optional + or − sign spaces, commas, and non-digit characters are not permitted between digits.
Examples for valid decimal integer constants are
123
−31
0
562321
+78
Examples for invalid integer constants are
20,000
₹1000

Real constants

Real constants consist of a fractional part in their representation. Integer constants are inadequate to represent quantities that vary continuously.
Examples of real constants are
0.0026
−0.97
435.29
+487.0

Single character constants

A single character constant represents a single character which is enclosed in a pair of quotation symbols.
Examples for character constants are
'5'
'x'
';'

String constants

A string constant is a set of characters enclosed in double quotation marks. The characters in a string constant sequence may be alphabet, number, special character and blank space.
Examples of string constants are
"VISHAL"
"1234"
"C language"
"!….?"

Naming constants

A name given to a constant value. Value of name does not change during program execution.

Using const keyword

When we use 'const' with data type, memory will be allocated to variable and the initialized value does not change.
const int $x = 10$;
const float pi = 3.141;

Using # de ine

define x 10
define pi 3.141
Where '# define' is instruction to preprocessor so memory is allocated. The preprocessor replace each occurrence of name with value in program before execution.

OPERATOR

An operator is a symbol which performs operations on given data elements.

Table 2 *Precedence and Associativity*

( ) Parenthesis [ ] Index → Member of • Member of	$L-R$
Pre ++, – – (unary) – , &(address of) * (Indirection)	$R-L$
Arithmetic * , /, %	$L-R$
Arithmetic: +, –	$L-R$
Bitwise shift : ≪, ≫	$L-R$
Relational: <, >, =. >, > = = =, ! =	$L-R$
Bitwise ex –OR : ∧	$L-R$
Logical AND : &&	$L-R$
Logical OR : ||	$L-R$
Conditional: ? :	$R-L$
Assignment & compound Assignment =, + =, – =, * =, / = ; % =	$R-L$
Separation operator: , (comma)	$L-R$

Note: For Assignment operator, only a variable is allowed on its left.

Precedence Decreases as We Move from Top to Bottom

Examples:

1. int a,b,c;
   $a = b = c = 0$;
   Assigns '0' to a,b,c;

2. int a, $b = 55$, $c = 10$;
   initializes 'b' with 55 and 'c' with '10'.
   $b + c = a$; // Invalid
   Only variable is allowed on left side of assignment.

3. int $a = 15$, $b = 20$, $c = 2$, $d = 5$, $e = 10$, f, g, h, i;
   $f = a \ll c$;
   'a' is left shifted for 'c' times and result stored in 'f'
   i.e.,
   $a = 15 = (1\ 1\ 1\ 1)_2$
   ↓ ↓ ↓ ↓
   1 1 1 1 0 (After first shift)
   ↓ ↓ ↓ ↓
   1 1 1 1 0 0 (After second shift)
   One left shift multiplies 15 by 2 = 30
   Again the 2nd left shift multiplies 30 by 2 = 60
   Thus $15 \times 2^2 = 60$, where the power of 2 is the number of times shift is made. Value of 'f' becomes 60.

Note: Left shift multiplies the value by 2. Right shift divides the value by 2.

$g = a$ and b;
a – 0 1 1 1 1
b – 1 0 1 0 0
0 0 1 0 0 = 4
'&' performs bitwise AND. So 'g' value is '4'.

$h = a|b$; a – 0 1 1 1 1
b – 1 0 0 0 0
1 1 1 1 1 = 31
"I" performs bitwise 'OR'. R value is '31'.

$i = a \hat{} d$: a – 1 1 1 1
b – 0 1 0 1
1 0 1 0 = 10
'^' performs bit-wise ex – OR. i value is '10'.

4. int $a = 100$, $b = 200$, $c = 300$, x;
   x = (a>b)?((a>c)?a:c):((b>c)? b:c);
   false c
   x = c
   so, x = 300

5. int $i = 10, j = 10, x, y$;
   x = i++++++i+i++++++i+++i
   executes as
   ++ i
   ++ i } pre-increments
   ++ i
   X = i + i + i + i + i
   i ++ ; } post-increments
   i ++ ;
   so x = 65, i = 15.
   y = j – – + – – j + j – – + – – j + – – j

EXERCISES

Practice Problems I

Directions for questions 1 to 15: Select the correct alternative from the given choices.

1. What will be the output of the following program?

```
void main()
{
int i;
char a[ ] =" \0 ";
if (printf("%s\n", a))
printf ("ok \n");
else
printf("program error \n");
}
```

(A) ok (B) progam error
(C) no output (D) compilation error

2. Output of the following will be

```
# define FALSE-1
# define TRUE 1
# define NULL 0
main( )
{
    if(NULL)
    puts("NULL");
    else if(FALSE)
    puts("TRUE");
    else
    puts("FALSE");
    }
```

(A) NULL (B) TRUE
(C) FALSE (D) 1

3. main( )

```
{
    printf("%x",-1 << 4) ;
}
```

For the above program output will be
(A) FFF0 (B) FF00
(C) 00FF (D) 0FFF

4. For the following program
# define sqr (*a*) *a***a*
main( )

```
    {
    int i;
    i = 64 / sqr(4);
    printf( "%d", i);
    }
```

output will be
(A) 4 (B) 16
(C) 64 (D) compilation error

5. #define clrscr ( ) 1000

```
main ( )
{
clrscr();
printf ( "%d \n", clrscr());
}
```

Output of the above program will be?
(A) error (B) No output
(C) 1000 (D) 1

6. Output of the following program is

```
main( )
{
    int i = -2;
    +i;
    printf("i = %d, +i = %d\n", i, +i);
```

(A) error (B) –2, +2
(C) –2, –2 (D) –2, 2

7. main( )

```
{
  int n;
  printf("%d", scanf ("%d", & n));
}
```

For the above program if input is given as 20. What will be the output?
(A) 20 (B) 1
(C) 2 (D) 0

8. How many times will the following code be executed?

```
{
    x = 10;
    while (x = 1)
          x ++;
}
```

(A) Never
(B) Once
(C) 15 times
(D) Infinite number of times

9. The following statement

```
printf("%d", 9%5); prints
```

(A) 1.8 (B) 1.0
(C) 4 (D) 2

10. int *a*;

```
printf("%d", a);
```

What is the output of the above code fragment?
(A) 0 (B) 2
(C) Garbage value (D) 3

11. printf("%d", printf("time"));
(A) syntax error
(B) outputs time 4
(C) outputs garbage
(D) prints time and terminates abruptly

12. The following program

```
main( )
{
    int i = 2;
    {
    int i = 4, j = 5;
```

Selected Questions

1. Which one of the following are essential features of an object-oriented programming language?
(i) Abstraction and encapsulation
(ii) Strictly-typedness
(iii) Type-safe property coupled with sub-type rule
(iv) Polymorphism in the presence of inheritance

(A) (i) and (ii) only
(B) (i) and (iv) only
(C) (i), (ii) and (iv) only
(D) (i), (iii) and (iv) only

2. Which of the following are true?

(i) A programming language which does not permit global variables of any kind and has no nesting of procedures/functions, but permits recursion can be implemented with static storage allocation

(ii) Multi-level access link (or display) arrangement is needed to arrange activation records only if the programming language being implemented has nesting of procedures/functions

(iii) Recursion in programming languages cannot be implemented with dynamic storage allocation

(iv) Nesting procedures/functions and recursion require a dynamic heap allocation scheme and cannot be implemented with a stack-based allocation scheme for activation records

(v) Programming languages which permit a function to return a function as its result cannot be implemented with a stack-based storage allocation scheme for activation records

(A) (ii) and (v) only (B) (i), (iii) and (iv) only (D)
(C) (i), (ii) and (v) only (ii), (iii) and (v) only

3. What will be the output of the following C program segment?

```
char inChar = 'A';
switch(inChar) {
case 'A': printf("choice A\n");
case 'B':
case 'C': printf("choice B");
case 'D'
case 'E':
default: printf("No Choice");}
```

(A) No choice
(B) Choice A
(C) Choice A
Choice B No choice
(D) Program gives no output as it is erroneous

4. Suppose n and p are unsigned int variables in a C program. We wish to set p to ${}^{n}C_{3}$. If n is large, which one of the following statements is most likely to set p correctly?

(A) $p = n * (n-1) * (n-2)/6;$
(B) $p = n * (n-1) /2 * (n-2)/3;$
(C) $p = n * (n-1) /3 * (n-2)/2;$
(D) $p = n * (n-1) * (n-2)/6.0;$

5. The secant method is used to find the root of an equation $f(x) = 0$. It is started from two distinct estimates x_a and x_b for the root. It is an iterative procedure involving linear interpolation to a root. The iteration stops if $f(x_b)$ is very small and then x_b is the solution. The procedure is given below. Observe that there is an expression which is missing and is marked by ?. Which is the suitable expression that is to put in place of ? so that it follows all steps of the secant method?

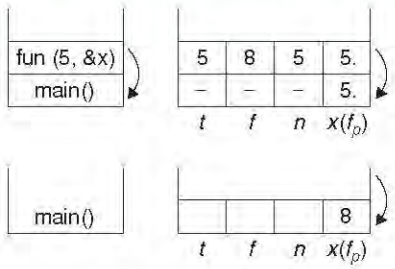
Secant

Initialize: x_a, x_b, ε, N // ε = convergence indicator
// N = maximum no. of iterations
$f_b = f(x_b)$
$i = 0$
while ($i < N$ and $|f_b| > \varepsilon$) do
$i = i + 1$ // update counter
x_t = ? // missing expression for
// intermediate value
$x_a = x_b$ // reset x_a
$x_b = x_t$ // reset x_b
$f_b = f(x_b)$ // function value at new x_b
end while
if $|f_b| > \varepsilon$ then // loop is terminated with $i = N$
write "Non-convergence"
else
write "return x_b"
end if

(A) $x_b - (f_b - f(x_a))\, f_b/(x_b - x_a)$
(B) $x_a - (f_a - f(x_a))\, f_a/(x_b - x_a)$
(C) $x_b - (x_b - x_a)\, f_b/(f_b - f(x_a))$
(D) $x_a - (x_b - x_a)\, f_a/(f_b - f(x_a))$

6. Consider the following C program:

```
#include<stdio.h>
int main( )
{
int i, j, k = 0;
j = 2 * 3 / 4 + 2.0 / 5 + 8 / 5;
k -= --j;
for (i = 0; i < 5; i ++)
{
switch(i + k)
```



Finally, x contains '8', so printf prints '8'.

Example 2: What does the following program prints?

```
#include < stdio.h>
void f (int *p, int *q)
{
p=q;
*p=12;
}
int i = 0, j=1;
int main()
{
f(&i, &j);
printf(" %d%d ", i, j);
return 0 ;
}
```

(A) 2 12 (B) 12 1
(C) 0 1 (D) 0 12

Solution: (D)

```
main( )
f (&i, &j)
address of 'i' is stored in to p.
and address of 'j' is stored into 'q'.
i.e., *p and*q refers i and j.
The statement:
p = q; updates pointer 'p', so that both
pointers refer to parameter 'j'.
*p = 12
Changes value of 'j' to '12' But 'i' does
not effected. So, prints 0 12.
```

Example 3: What is the value printed by the following program?

```
# include <stdio.h>
int f(int *a, int n)
{
if (n<=0) return 0 ;
else if(*a%2 = = 0)
return *a + f(a+1, n-1);
else
return *a - f(a+1, n-1);
}
int main ( )
{
int G [ ] = { 12, 7, 13, 4, 11, 6};
printf("%d", f (a,b));
return 0;
}
```

(a) –9 (b) 12
(c) 15 (d) 20

Solution: (C)

	0	1	2	3	4	5
a	12	7	13	4	11	6

$f(a, 6)$ is the first call to function $f( )$.

The array _ name refers to base address of array, i.e., address of first element.

Thus,

$F(a, 6)$

12 % 2 = 0. So,

$$12 + f\underset{\downarrow 7}{(\overline{a+1}),}\left(\frac{n-1}{5}\right)[*a \text{ is even}]$$

$$12 + \left(7 - \left(\underset{\downarrow 13}{f(a+1),}\left(\frac{n-1}{4}\right)\right)\right)[*a \text{is odd}]$$

$$12 + \left(7 - \left(13 - f\left(\underset{\downarrow 4}{(a+1)}\right),\left(\frac{n-1}{3}\right)\right)\right)[*a \text{ is odd}]$$

$$12 + \left(7 - \left(13 - \left(4 + f\left(\underset{\downarrow 11}{(a+1)}, \frac{(n-1)}{2}\right),\right)\right)\right)[*a \text{ is even}]$$

$$12 + \left(7 - \left(13 - \left(4 + \left(11 - f\left(\underset{\downarrow 6}{(a+1)}, \left(\frac{n-1}{1}\right)\right)\right)\right)\right)\right)[*a \text{ is odd}]$$

$$12 + \left(7 - \left(13 - \left(4 + \left(11 - \left(6 + \underset{\downarrow 0}{\frac{f(a+1), (n-1)}{0}}\right)\right)\right)\right)\right)[*a \text{ is even}]$$

$$12 + (7 \quad (13 - (4 + (11 - (6 + 0))))) = 15$$

Scope, Lifetime and Binding

Storage classes specify the scope, lifetime and binding of variables. To fully define a variable, one needs to mention not only its 'type' but also its 'storage class'.

A variable name identifies some physical location within computer memory where a collection of bits are allocated for storing value of variable.

Storage class tells us:

1. Where the variable would be stored (either in memory or CPU registers)?
2. What will be the initial value of a variable, if no value is specifically initialized?
3. What is the scope of a variable (where it can be accessed)?
4. What is the life of a variable?

```
    p = s1 + 2;
    *p = '0';
    printf("%s", s1);
}
```

What will be printed by the program?

(A) 12 (B) 120400
(C) 1204 (D) 1034

7. Consider the following C program

```
#include<stdio.h>
int main ( )
{
static int a[ ] = {10, 20, 30, 40, 50};
static int *p[ ] = {a, a+3, a+4, a+1, a+2};
int **ptr = p;
ptr++;
printf("%d%d", ptr-p, **ptr);
}
```

8. Consider the following C program.

```
void f(int, short);
void main( )
{
int i = 100;
short s = 12;
short *p = &s;
_____; // call to f( )
}
```

Which one of the following expressions, when placed in the blank above, will **NOT** result in a type checking error?

(A) $f(s,*s)$ (B) $i = f(i,s)$
(C) $f(i,*s)$ (D) $f(i,*p)$

9. Consider the following *C* program.

```
# include<stdio.h>
void mystery (int *ptra, int *ptrb) {
int *temp;
temp = ptrb;
ptrb = ptra;
ptra = temp;

}
int main ( ) {
int a = 2016, b = 0, c = 4, d = 42;
mystery (&a, &b);
if (a < c)
    mystery(&c, &a);
mystery (&a, &d);
printf("%d\n", a)
}
```

The output of the program is _____.

10. The following function computes the maximum value contained in an integer array $p[\,]$ of size n ($n >= 1$).

```
int max (int *p, int n) {
    int a = 0, b = n – 1;
    while (_____) {
            if (p [a] <= p [b]) {a = a+1;}
            else                { b = b – 1;}
    }
    return p[a];
}
```

The missing loop condition is

(A) $a != n$
(B) $b != 0$
(C) $b > (a+1)$
(D) $b != a$

11. The value printed by the following program is ___.

```
void f (int* p, int m) {
m = m +5;
*p = *p + m;
return;
}
void main () {
int i = 5, j = 10;
f(&i, j);
printf ("%d", i +j);
}
```

12. Consider the following program:

```
int f (int *p, int n)
{   if (n <= 1) return 0;
else return max (f (p +1, n – 1), p [0] – p [1] );
}
int main ()
{
int a[ ] = {3,5,2,6,4};
printf ("%d", f(a,5));
}
```

Note: *max (x,y) returns the maximum of x and y.*

The value printed by this program is _____

Chapter 5

Trees

LEARNING OBJECTIVES

- *Tree*
- *2-Tree*
- *Binary tree*
- *Properties of binary trees*
- *Complete binary tree*
- *Full binary tree*
- *Binary tree representation*
- *Linked representation*
- *Binary search tree*
- *Binary tree traversing methods*
- *AVL tree*
- *Binary heap*
- *Max-heap*
- *Min-heap*
- *Expression tree*

TREE

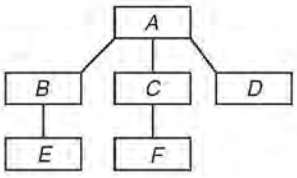
Tree is non-linear data structure designated at a special node called root and elements are arranged in levels without containing cycles.

(or)

The tree is

1. Rooted at one vertex
2. Contains no cycles
3. There is a sequence of edges from any vertex to any other
4. Any number of elements may connect to any node (including root)
5. A unique path traverses from root to any node of tree
6. Tree stores data in hierarchical manner
7. The elements are arranged in layers

Example:



- Root node is A.
- A's children are B, C and D.
- E, F and D are leaves.
- Nodes B, C are called as intermediate nodes.
- A is parent of B, C and D.
- B is parent of E and C is parent of F.
- Number of children of a node is called degree of node.

2-TREE

A tree in which every node contains either 0 or 2 children.

BINARY TREE

It is a special type of tree where each node of tree contains either 0 or 1 or 2 children.

(or)

Binary Tree is either empty, or it consists of a root with two binary trees called left-sub tree and right sub-tree of root (left or right or both the sub trees may be empty).

Properties of binary tree

- Binary tree partitioned into three parts.
- First subset contains root of tree.
- Second subset is called left subtree.
- Another subset is called right subtree.
- Each subtree is a binary tree.
- Degree of any node is 0/1/2.
- The maximum number of nodes in a tree with height 'h' is $2^{h+1}-1$.
- The maximum number of nodes at level 'i' is 2^{i-1}.
- For any non-empty binary tree, the number of terminal nodes with n_2, nodes of degree 2 is $N_0 = n_2 + 1$
- The maximum number of nodes in a tree with depth d is $2^d - 1$.

EXERCISES

Practice Problems I

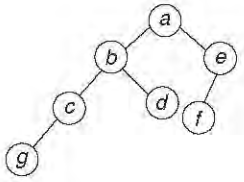
Directions for questions 1 to 15: Select the correct alternative from the given choices.

1. A binary tree T has n leaf nodes. The number of nodes of degree two in T is ____.
(A) n (B) $n-1$
(C) $\log n$ (D) $n+1$

2. How many numbers of binary tree can be created with 3 nodes which when traversed in post-order gives the sequence C, B, A?
(A) 3 (B) 5
(C) 8 (D) 15

3. A binary search tree contains the values 3, 6, 10, 22, 25, 30, 60, 75. The tree is traversed in pre-order and the values are printed out. Which of the following sequence is a valid output?
(A) 25 6 3 10 22 60 30 75
(B) 25 6 10 3 22 75 30 60
(C) 25 6 75 60 30 3 10 22
(D) 75 30 60 22 10 3 6 25

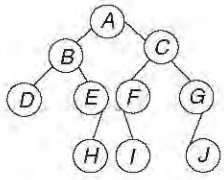
4. Figure shows a balanced tree. How many nodes will become unbalanced when a node is inserted as a child of the node 'g'?



(A) 7 (B) 2
(C) 3 (D) 8

5. A full binary tree with n non-leaf nodes contains
(A) $2n$ nodes (B) $\log_2 n$ node
(C) $n+1$ nodes (D) $2n+1$ nodes

6. Which of the following list of nodes corresponds to a post order traversal of the binary tree shown below?



(A) *A B C D E F G H I J* (B) *J I H G F E D C B A*
(C) *D H E B I F J G C A* (D) *D E H F I G J B C A*

7. Which of the following sequence of array elements forms as heap?
(A) {23, 17, 14, 6, 13, 10, 1, 12, 7, 5}
(B) {23, 17, 14, 6, 13, 10, 1, 5, 7, 12}
(C) {23, 17, 14, 7, 13, 10, 1, 5, 6, 12}
(D) {23, 17, 14, 7, 13, 10, 1, 12, 5, 6}

8. What is the maximum height of any AVL tree with 7 nodes? Assume that the height of a tree with a single node is 0.
(A) 2 (B) 3
(C) 4 (D) 5

9. A binary search tree is generated by inserting in order the following integers:

55, 15, 65, 5, 25, 59, 90, 2, 7, 35, 60, 23.

The number of nodes in the left subtree and right subtree of the root respectively are
(A) 8, 3 (B) 7, 4
(C) 3, 8 (D) 4, 7

10. In a complete binary tree of n nodes, how far are the most distant two nodes? Assume each in the path counts as 1.
(A) about $\log_2 n$ (B) about $2\log_2 n$
(C) about $3\log_2 n$ (D) about $4\log_2 (n)$

11. A complete binary tree of level 5 has how many nodes?
(A) 20 (B) 63
(C) 30 (D) 73

Common data for questions 12 and 13: A 3-ary max-heap is like a binary max-heap, but instead of 2 children, nodes have 3 children. A 3-ary heap can be represented by an array as follows:

The root is stored in the first location, $a[0]$, nodes in the next level from left to right is stored from $a[1]$ to $a[3]$ and so on. An item x can be inserted into a 3-ary heap containing n items by placing x in the location $a[n]$ and pushing it up the tree to satisfy the heap property.

12. Which one of the following is a valid sequence of elements in an array representing 3-ary max-heap?
(A) 1, 3, 5, 6, 8, 9 (B) 9, 6, 3, 1, 8 , 5
(C) 9, 3, 6, 8, 5, 1 (D) 9, 5, 6, 8, 3, 1

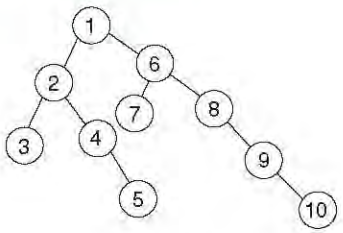
13. Suppose the elements 7, 2, 10 and 4 are inserted, in that order, into the valid 3-ary max-heap found in the above question. Which one of the following is the sequence of items in the array representing the resultant heap?
(A) 10, 7, 9, 8, 3, 1, 5, 2, 6, 4
(B) 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
(C) 10, 9, 4, 5, 7, 6, 8, 2, 1, 3
(D) 10, 8, 6, 9, 7, 2, 3, 4, 1, 5

14. Consider the nested representation of binary trees : (X Y Z) indicated Y and Z are the left and right subtrees respectively, of node X(Y and Z may be null (or) further nested) which of the following represents a valid binary tree?

Algorithms

UNIT 1

PART B



	Left child	Right child
1	2	6
2	3	4
3	0	0
4	0	5
5	0	0
6	7	8
7	0	0
8	0	9
9	0	10
10	0	0

Figure 5 A binary tree and its representation

- Vertex 3 is of depth '2', height '0' and the level is 2 (Height of tree – depth of '3' = 4 – 2 = 2).
- A binary tree is represented by 2 arrays: left child and right child.
- A binary tree is said to be complete if for some integer k, every vertex of depth less than k has both a left child and a right child and every vertex of depth k is a leaf. A complete binary tree of height k has exactly $(2^{k+1} - 1)$ vertices.
- A complete binary tree of height k is often represented by a single array. Position 1 in the array contains the root. The left child of the vertex in position 'i' is located at position '$2i$' and the right child at position '$2i + 1$'.

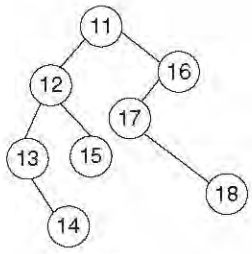
Tree Traversals

Many algorithms which make use of trees often traverse the tree in some order. Three commonly used traversals are pre-order, postorder and inorder.

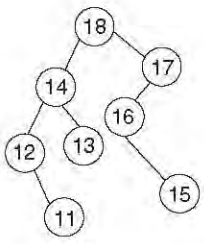
Pre-order Traversal

A pre-order traversal of T is defined recursively as follows:

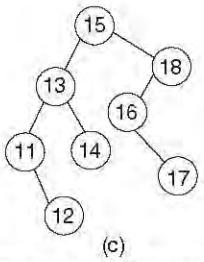
1. Visit the root.
2. Visit in pre-order the sub trees with roots $v_1, v_2 \ldots v_k$ in that order.



(a)



(b)



(c)

Figure 6 (a) Pre-order, (b) Post-order **(c)** In-order

Post-order traversal

A post-order traversal of T is defined recursively as follows:

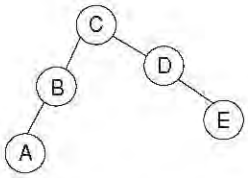
1. Visit in post-order the sub trees with roots v_1, v_2, v_3, ... v_k in that order.
2. Visit the root r.

In-order Traversal

An in-order traversal is defined recursively as follows:

1. Visit in-order the left sub tree of the root 'r'.
2. Visit 'r'.
3. Visit in inorder the right sub tree of r.

Example: Consider the given tree



What are the pre-order, post-order and in-order traversals of the above tree?

Solution: Pre-order – CBADE
Post-order – ABEDC
In-order – ABCDE

Data Structure

A data structure is a way to store and organize data in-order to facilitate access and modifications. No single data structure works well for all purposes, it is important to know the strengths and limitations of several data structures.

Efficiency

Algorithms devised to solve the same problem often differ dramatically in their efficiency. Let us compare efficiencies of Insertion sort and merge sort; insertion sort, takes time equal to $C_1 n^2$ to sort 'n' elements, where C_1 is a constant that does not depend on 'n'. It takes time proportional to n^2, merge sort takes time equal to $C_2 n \log n$, C_2 is another constant that also does not depend on 'n'. Insertion sort has a smaller constant factor than merge sort ($C_1 < C_2$) constant factors are far less significant in the running time.

Selected Questions

1. The median of n elements can be found in $O(n)$ time. Which one of the following is correct about the complexity of quick sort, in which median is selected as pivot?

(A) $\theta(n)$ (B) $\theta(n \log n)$
(C) $\theta(n^2)$ (D) $\theta(n^3)$

2. Given two arrays of numbers $a_1 \ldots a_n$ and $b_1 \ldots b_n$ where each number is 0 or 1, the fastest algorithm to find the largest span (i, j) such that $a_i + a_i + 1 + \cdots + a_j = b_i + b_i + 1 + \cdots + b_j$, or report that there is not such span,

(A) Takes $O(3^n)$ and $\Omega(2^n)$ time if hashing is permitted
(B) Takes $O(n^3)$ and $\Omega(n^{2.5})$ time in the key comparison model
(C) Takes $\Theta(n)$ time and space
(D) Takes $O(\sqrt{n})$ time only if the sum of the $2n$ ele-ments is an even number

3. 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 $n > 0$ is: **[2007]**

(A) $\lceil \log_2 n \rceil + 1$ (B) n
(C) $\lceil \log_2 + n \rceil$ (D) $\lfloor \log_2 n \rfloor + 1$

4. In the following *C* function, let $n \geq m$.

```
int gcd(n,m)
{
if (n%m = =0) return m;
    n = n%m;
return gcd(m,n);
}
```

How many recursive calls are made by this function?

(A) $\Theta(\log_2 n)$ (B) $\Omega(n)$
(C) $\Theta(\log_2 \log_2 n)$ (D) $\Theta(\sqrt{n})$

5. What is the time *complexity* of the following recursive function:

```
int DoSomething (int n) {
if (n <= 2)
    return 1;
  else
return(DoSomething (floor(sqrt(n)))+ n);}
```

(A) $\Theta(n^2)$ (B) $\Theta(n \log_2 n)$
(C) $\Theta(\log_2 n)$ (D) $\Theta(\log_2 \log_2 n)$

6. An array of n numbers is given, where n is an even number. The maximum as well as the minimum of these n numbers needs to be determined. Which of the following is TRUE about the number of comparisons needed?

(A) At least $2n - c$ comparisons, for some constant c, are needed.
(B) At most $1.5n - 2$ comparisons are needed.
(C) At least $n \log_2 n$ comparisons are needed.
(D) None of the above.

7. Consider the following *C* code segment:

```
int IsPrime(n)
    {
    int i,n;
    for(i=2; i<= sqrt(n); i ++)
      if(n%i = =0)
       {printf("Not Prime\n"); return 0;}
    return 1;
}
```

Let $T(n)$ denote the number of times the *for* loop is executed by the program on input n. Which of the following is TRUE?

(A) $T(n) = O(\sqrt{n})$ and $T(n) = \Omega(\sqrt{n})$
(B) $T(n) = O(\sqrt{n})$ and $T(n) = \Omega(1)$
(C) $T(n) = O(n)$ and $T(n) = \Omega(\sqrt{n})$
(D) None of the above

8. The most efficient algorithm for finding the number of connected components in an undirected graph on n vertices and m edges has time complexity

(A) $\Theta(n)$ (B) $\Theta(m)$
(C) $\Theta(m + n)$ (D) $\Theta(mn)$

9. Consider the following functions:

$f(n) = 2^n$
$g(n) = n!$
$h(n) = n^{\log n}$

Which of the following statements about the asymptotic behavior of $f(n)$, $g(n)$, and $h(n)$ is true?

(A) $f(n) = O(g(n)); g(n) = O(h(n))$
(B) $f(n) = \Omega(g(n)); g(n) = O(h(n))$
(C) $g(n) = O(f(n)); h(n) = O(f(n))$
(D) $h(n) = O(f(n)); g(n) = \Omega(f(n))$

10. The minimum number of comparisons required to determine if an integer appears more than $n/2$ times in a sorted array of n integers is

(A) $\Theta(n)$ (B) $\Theta(\log n)$
(C) $\Theta(\log * n)$ (D) $\Theta(1)$

11. We have a binary heap on n elements and wish to insert n more elements (not necessarily one after another) into this heap. The total time required for this is

Chapter 2

Sorting Algorithms

LEARNING OBJECTIVES

- *Sorting algorithms*
- *Merge sort*
- *Bubble sort*
- *Insertion sort*
- *Selection sort*
- *Selection sort algorithm*
- *Binary search trees*
- *Heap sort*
- *Sorting–performing delete max operations*
- *Max-heap property*
- *Min-heap property*
- *Priority queues*

Sorting Algorithms

Purpose of sorting

Sorting is a technique which reduces problem complexity and search complexity.

- Insertion sort takes $\theta(n^2)$ time in the worst case. It is a fast inplace sorting algorithm for small input sizes.
- Merge sort has a better asymptotic running time $\theta(n \log n)$, but it does not operate in inplace.
- Heap sort, sorts 'n' numbers inplace in $\theta(n \log n)$ time, it uses a data structure called heap, with which we can also implement a priority queue.
- Quick sort also sorts 'n' numbers in place, but its worst – case running time is $\theta(n^2)$. Its average case is $\theta(n \log n)$. The constant factor in quick sort's running time is small, This algorithm performs better for large input arrays.
- Insertion sort, merge sort, heap sort, and quick sort are all comparison based sorts; they determine the sorted order of an inputtarray by comparing elements.
- We can beat the lower bound of Ω ($n \log n$) if we can gather information about the sorted order of the input by means other than comparing elements.
- The counting sort algorithm, assumes that the input numbers are in the set $\{1, 2, \ldots. k\}$. By using array indexing as a tool for determining relative order, counting sort can sort n numbers in $\theta(k + n)$ time. Thus counting sort runs in time that is linear in size of the input array.
- Radix sort can be used to extend the range of counting sort. If there are 'n' integers to sort, each integer has 'd' digits, and each digit is in the set $\{1, 2, \ldots k\}$, then radix sort can sort the numbers in $\theta(d\,(n + k))$ time. Where 'd' is constant. Radix sort runs in linear time.
- Bucket sort, requires knowledge of the probabilistic distribution of numbers in the input array.

Merge Sort

Suppose that our division of the problem yields 'a' sub problems, each of which is $\left(\frac{1}{b}\right)$th size of the original problem. For merge sort, both a and b are 2, but sometimes $a \neq b$. If we take $D(n)$ time to divide the problem into sub problems and $C(n)$ time to combine the solutions of the sub problems into the solution to the original problem. The recurrence relation for merge sort is

$$T(n) = \begin{cases} \theta(1) \text{ if } n \leq c, \\ aT(n/b) + D(n) + C(n) \text{ otherwise} \end{cases}$$

Running time is broken down as follows:

Divide: This step computes the middle of the sub array, which takes constant time $\theta(1)$.

Conquer: We solve 2 sub problems of size $(n/2)$ each recursively which takes $2T(n/2)$ time.

Combine: Merge sort procedure on an n-element sub array takes time $\theta(n)$.

Chapter 3

Divide-and-conquer

LEARNING OBJECTIVES

- *Divide-and-conquer*
- *Divide-and-conquer examples*
- *Divide-and-conquer technique*
- *Merge sort*
- *Quick sort*
- *Performance of quick sort*
- *Recurrence relation*
- *Searching*
- *Linear search*
- *Binary search*

Divide-and-Conquer

Divide-and-conquer is a top down technique for designing algorithms that consists of dividing the problem into smaller sub problems hoping that the solutions of the sub problems are easier to find and then composing the partial solutions into the solution of the original problem.

Divide-and-conquer paradigm consists of following major phases:

- Breaking the problem into several sub-problems that are similar to the original problem but smaller in size.
- Solve the sub-problem recursively (successively and independently)
- Finally, combine these solutions to sub-problems to create a solution to the original problem.

Divide-and-Conquer Examples

- Sorting: Merge sort and quick sort
- Binary tree traversals
- Binary Search
- Multiplication of large integers
- Matrix multiplication: Strassen's algorithm
- Closest-pair and Convex-hull algorithm

Merge Sort

Merge sort is a sorting algorithm for rearranging lists (or any other data structure that can only be accessed sequentially, e.g., file streams) into a specifiedorder .

Merge sort works as follows:

1. Divide the unsorted list into two sub lists of about half the size.
2. Sort each of the two sub lists.

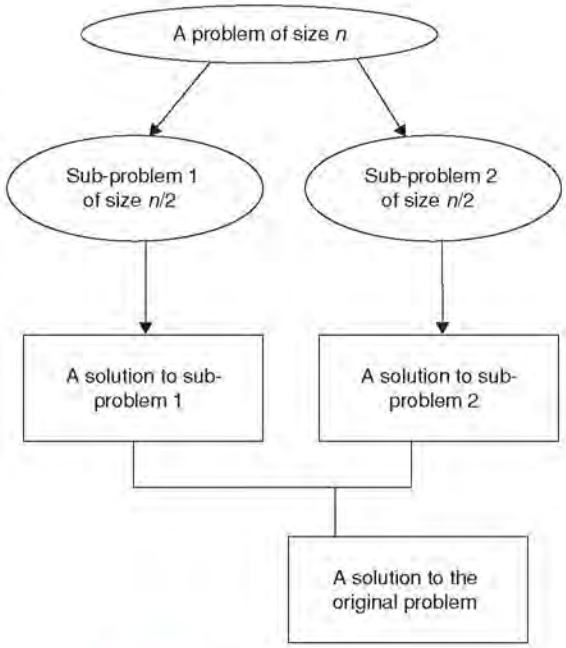


Figure 1 Divide-and-conquer technique.

3. Merge the two sorted sub lists back into one sorted list
4. The key of merge sort is merging two sorted lists into one, such that if we have 2 lists
   $X(x_1 \le x_2 \le x_3 \cdots \le x_m)$ and
   $Y(y_1 \le y_2 \le y_3 \cdots \le y_n)$ the resulting list is $z(z_1 \le z_2 \le \cdots \le z_{m+n})$

Example 1: $L_1 = \{3, 8, 9\}$, $L_2 = \{1, 5, 7\}$

Merge $(L_1, L_2) = \{1, 3,5,7,9\}$

Chapter 4

Greedy Approach

LEARNING OBJECTIVES

- *Greedy approach*
- *Knapsack problem*
- *Fractional knapsack problem*
- *Spanning trees*
- *Prim's algorithm*
- *Kruskal's algorithm*
- *Tree and graph traversals*
- *Back tracking*
- *Graph traversal*
- *Breadth first traversal*
- *Depth first search*
- *Huffman codes*
- *Task-scheduling problem*
- *Sorting and order statistics*
- *Simultaneous minimum and maximum*
- *Graph algorithms*

Greedy Approach

In a greedy method, we attempt to construct an optimal solution in stages.

- At each stage we make a decision that appears to be the best (under some criterion) at the time.
- A decision made at one stage is not changed in a later stage, so each decision should assure feasibility.
- Some problems that use greedy approach are:

1. Knapsack problem
2. Minimum spanning tree
3. Prims algorithm
4. Kruskals algorithm

Knapsack Problem

The knapsack problem is a problem in combinatorial optimization: given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. We have n kinds of items, 1 through n. Each kind of item i has a value V_i and a weight W_i, usually assume that all values and weights are non-negative. The maximum weight that we can carry in the bag is W.

Solved Examples

Example 1: (Making change)

Problem:	Accept n dollars, to return a collection of coins with a total value of n dollars.
Configuration:	A collection of coins with a total value of n.
Objective function:	Minimize number of coins returned.
Greedy solution:	Always return the largest coin you can.

- Coins are valued \$.30, \$.020, \$0.05, \$0.01 use a greedy choice property and make \$.40 by using 3 coins.

Solution: \$0.30 + \$0.05 + \$0.05 = \$0.40

Fractional Knapsack Problem

Given: A set S of n items, with each item i having

- b_i – a positive benefit
- w_i – a positive weight

Goal: Choose items with maximum total benefit but with weight atmost W.

If we are allowed to take fractional amounts, then this is the fractional knapsack problem.

- In this case, let x_i denote the amount we take of item i.
- Objective: Maximize $\sum_{i \in S} b_i (x_i / w_i)$
- Constraint: $\sum_{i \in S} x^i \leq W$

Chapter 5

Dynamic Programming

LEARNING OBJECTIVES

- *Dynamic programming*
- *Multi-stage graph*
- *All pairs shortest path problem*
- *Hashing methods*
- *Mid-square method*
- *Folding method*
- *Resolving collisions*
- *Matrix chain multiplication*
- *Longest common subsequence*
- *Optimal substructure of LCS*
- *NP-hard and NP-complete*
- *P-problem*
- *NP-problem*
- *P, NP, and Co-NP*
- *Cooks theorem*
- *Non-deterministic search*

Dynamic Programming

Dynamic programming is a method for solving complex problems by breaking them down into simpler sub problems. It is applicable to problems exhibiting the properties of overlapping sub problems which are only slightly smaller, when applicable; the method takes far less time than naive method.

- The key idea behind dynamic programming is to solve a given problem, we need to solve different parts of the problem (sub problems) then combine the solutions of the sub problems to reach an overall solution. Often, many of these sub problems are the same.
- The dynamic programming approach seeks to solve each sub problem only once, thus reducing the number of computations. This is especially useful when the number of repeating sub problems is exponentially large.
- There are two key attributes that a problem must have in order for dynamic programming to be applicable 'optimal sub structure' and 'overlapping sub-problems'. However, when the overlapping problems are much smaller than the original problem, the strategy is called 'divide-and-conquer' rather than 'dynamic programming'. This is why merge sort-quick sort are not classified as dynamic programming problems.

Dynamic programming is applied for:

- Multi stage graph
- All pairs shortest path

Principle of Optimality

It states that whatever the initial state is, remaining decisions must be optimal with regard to the state following from the first decision.

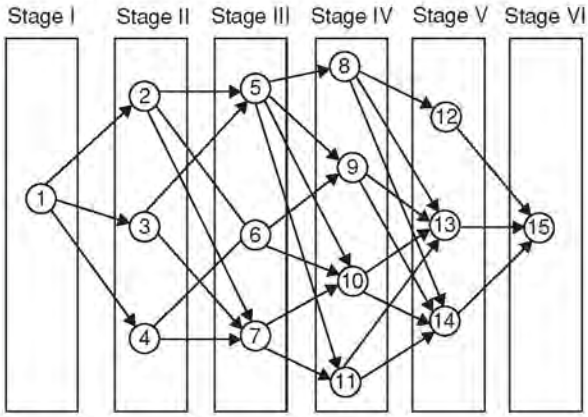
To solve a problem using dynamic programming strategy, it must observe the principle of optimality.

Multi-stage Graph

A multi-stage graph is a graph

- $G = (V, E)$ with V partitioned into $K >= 2$ disjoint subsets such that if (a, b) is in E, then a is in V_i, and b is in V_{i+1} for some sub sets in the partition;
- $|V_1| = |V_K| = 1$ the vertex S in V_1 is called the source; the vertex t is called the sink.
- G is usually assumed to be a weighted graph.
- The cost of a path from node V to node W is sum of the costs of edges in the path.
- The 'multi-stage graph problem' is to find the minimum cost path from S to t.

Example:



Costs of edges

1 – 2 → 10
1 – 3 → 20

Computer Organization and Architecture

UNIT 2

Chapter 1

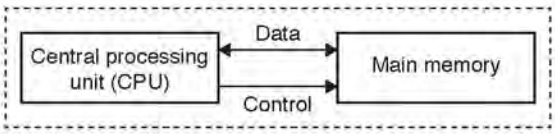
Machine Instructions, Addressing Modes

LEARNING OBJECTIVES

- *Computer*
- *Computer system*
- *Computer component*
- *Machine instruction*
- *Instruction types*
- *Types of operands*
- *Types of operations*
- *Procedure call instruction*
- *Addressing modes*
- *Computer performance*

COMPUTER

A computer is a data-processing machine which is operated automatically under the control of a list of instructions (called a program) stored in its main memory.



Computer System

- A computer system consists usually of a computer and its peripherals.
- Computer peripherals include input devices, output devices and secondary memories.

Computer Architecture: Computer Architecture refers to those attributes of a system visible to a programmer, i.e., the attributes that have direct impact on the logical execution of a program.

Example: Whether a computer will have a multiply instruction or not.

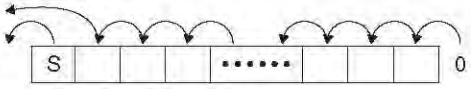
Computer Organization: Computer organization refers to operational units and their interconnections that realize the architectural specifi cations.

Example: Whether the multiply instruction will be implemented by a special multiply unit or by a mechanism that makes repeated use of the add unit of the system.

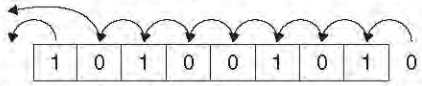
Computer Components

- $R_1, R_2 \ldots R_n$: General Purpose Registers.
- **PC:** Program counter. Holds address of next instruction to be executed. PC = PC + I (I = instruction length)
- **IR:** Instruction Register. It holds the instruction which is fetched from memory.
- **MAR:** Memory Address Register: MAR specifies the address in memory for the next read or write.
- **MBR:** Memory Buffer Register. It contains the data to be written into memory or receives the data read from memory.
- **Input–outputAR:** Input–output Address Register. It specifies a particular input–output device.
- **Input–outputBR:** Input–output Buffer Register. Used for the exchange of data between an input–output module and the CPU.
- **ALU:** Arithmetic and Logic Unit. Used to perform arithmetic and logical operations.

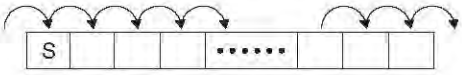
(c) Arithmetic left shift: Arithmetic shift operation treats the data as a signed integer and does not shift the sign bit. In Arithmetic left shift, a logical left shift is performed on all bits but the sign bit, which is retained.



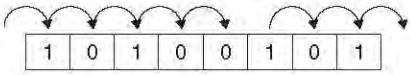
Example: $R_1 = 1010\ 0101$
Arithmetic Left shift $R_1 = 1100\ 1010$.



(d) Arithmetic right shift: Here, the sign bit is replicated into the bit position to its right.



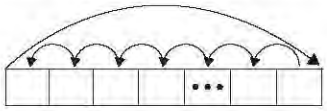
Example: $R_1 = 1010\ 0101$
Arithmetic Right shift $R_1 = 1101\ 0010$



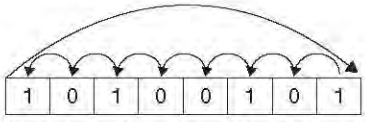
Notes:

1. With numbers in 2's complement form, a right arithmetic shift corresponds to a division by 2, with truncation for odd numbers.
2. Both arithmetic left shift and logical left shift correspond to a multiplication by 2 when there is no overflow.

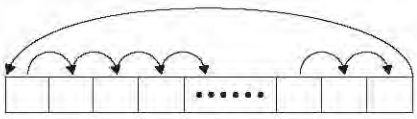
(e) Left rotate (Cyclic left shift): Rotate operations preserve all the bits being operated on. Here the bits from LSB will move one bit position to the left and MSB will placed in LSB position.



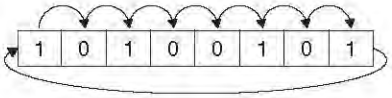
Example: $R_1 = 1010\ 0101$
Left Rotate $R_1 = 0100\ 1011$



(f) Right rotate (Cyclic right shift): Here the bits from MSB will be shifted to one bit position right and LSB is placed in MSB.



Example: R_1: 1010 0101
Right Rotate $R_1 = 1101\ 0010$.



(v) Transfer of control: This type of operations updates the program counter. Used for subroutine call/return, manage parameter passing and linkage.

Example: Jump, jump unconditional, return, execute, skip, skip conditional, Halt, Wait, NOP, etc.

(vi) Input–output operations: These are used to issue a command to input–output module.

Example: input, output, start input–output, test input–output etc.

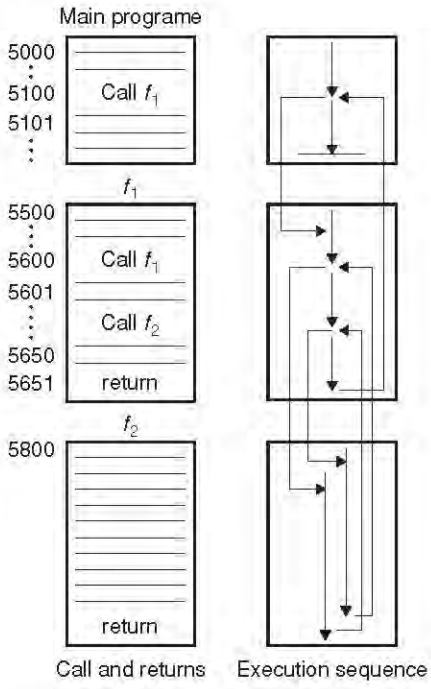
(vii) Conversion operations: These are similar to arithmetic and logical operations. May also involve special logic to perform conversion.

Procedure Call Instruction

A procedure is a self-contained computer program that is incorporated into a large program. It allows us to use the same piece of code many times. The procedure mechanism involves two basic instructions:

1. A call instruction that branches from the present location to the procedure.
2. A return instruction that returns from the procedure to the place from where it was called.

Example:



We can call a procedure from a variety of points, so the processor must somehow save the return address to return

Chapter 2

ALU and Data Path, CPU Control Design

LEARNING OBJECTIVES

- *Arithmetic and logic unit*
- *Fixed-point arithmetic operation*
- *Floating point arithmetic operation*
- *BCD*
- *Data path*
- *CPU control design*
- *Instruction cycle*
- *Control unit*
- *Control of processor*
- *Function of control unit*
- *Design of control unit*
- *Types of micro-instructions*
- *Micro-instruction sequencing*
- *RISC and CISC*
- *RISC characteristic*
- *CISC characteristic*

ALU (Arithmetic and Logic Unit)

ALU performs arithmetic and logical operations on data (*see* Figure 1).

Figure 1 ALU inputs and outputs

- Data are presented to ALU in registers and the results of an operation are stored in registers.
- Registers are temporary storage locations within the processor that are connected by signal paths to ALU.
- The control unit provides signals that control the operation of ALU and the movement of data into and out of the ALU.
- Here we will discuss
  1. Fixed-point arithmetic operations
  2. Floating-point arithmetic operations
  3. BCD data arithmetic operations

Fixed-point Arithmetic Operations

Fixed-point representation

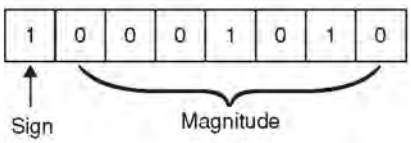
The numbers may be positive, zero or negative. So we have two types of numbers:

Unsigned numbers Only zero and positive integers can be repre-sented. All bits represent magnitude and no need of sign.

Signed numbers In signed representation, the most signifi cant bit represents the sign. If the number is positive, the MSB is 0 and remaining bits represent magnitude. If the number is negative, we have three techniques to represents that number:

1. **Signed magnitude representation:** In signed magnitude representation, the MSB represents sign and remaining bits represents magnitude. If the number is negative then the MSB is 1.

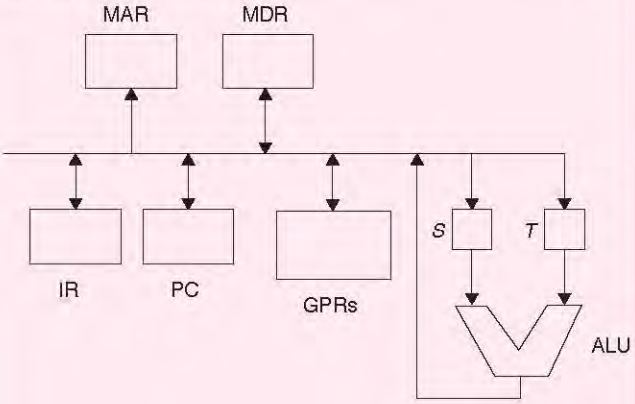
   Example: Signed magnitude representation of –10 =



2. **Signed 1's complement representation:** In signed 1's complement representation, the MSB bit is 1. The remaining bits of its signed magnitude bits are inverted i.e., convert 0's to 1's and 1's to 0's to obtain 1's complement.

Selected Questions

Common data for questions 1 and 2: 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 cycles 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 bus into the MDR.

1. The instruction 'add R_0, R_1' has the register transfer interpretation $R_0 \leftarrow R_0 + R_1$. The minimum number of clock cycles needed for execution cycle of this instruction is
   (A) 2 (B) 3
   (C) 4 (D) 5

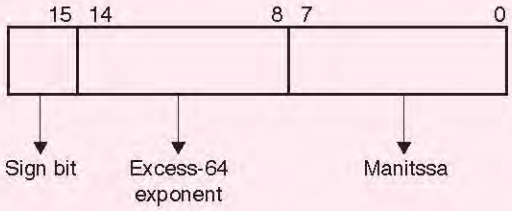
2. The instruction 'call *Rn*, sub' is a two word instruction. Assuming that PC is incremented during the fetch cycle of the first word of the instruction, its register transfer interpretation is

   $Rn \leftarrow PC + 1$;

   $PC \leftarrow M[PC]$

   The minimum number of CPU clock cycles, needed during the execution cycle of this instruction is
   (A) 2 (B) 3
   (C) 4 (D) 5

Data for question 3: Consider the following floating-point format.



Mantissa is a pure fraction in sign-magnitude form.

3. The normalized 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 normalized representation of the above number (0.239×2^{13}) is:
   (A) 0*A* 20 (B) 11 34
   (C) 49 *D*0 (D) 4*A E*8

4. In the IEEE floating point representation the hexa-decimal value 0x00000000 corresponds to
   (A) The normalized value 2^{-127}
   (B) The normalized value 2^{-126}
   (C) The normalized value + 0
   (D) The special value + 0

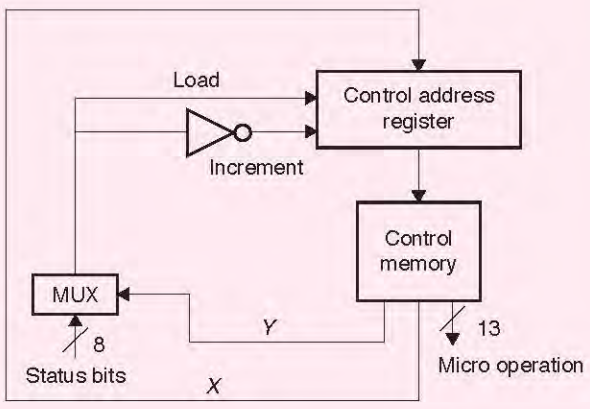
5. *P* is a 16-bit signed integer. The 2's complement rep-resentation of *P* is $(F87B)_{16}$. The 2's complement rep-resentation of 8**P* is
   (A) $(C3D8)_{16}$ (B) $(187B)_{16}$
   (C) $(F878)_{16}$ (D) $(987B)_{16}$

6. The decimal value 0.5 in IEEE single precision float-ing point representation has
   (A) fraction bits of 000 ... 000 and exponent value of 0
   (B) fraction bits of 000...000 and exponent value of –1
   (C) fraction bits of 100...000 and exponent value of 0
   (D) no exact representation

7. The smallest integer that can be represented by an 8-bit number in 2's complement form is
   (A) –256 (B) –128
   (C) –127 (D) 0

8. Let $A = 1111\ 1010$ and $B = 0000\ 1010$ be two 8-bit 2's complement numbers. Their product in 2's complement is
   (A) 1100 0100 (B) 1001 1100
   (C) 1010 0101 (D) 1101 0101

9. The microinstructions stored in the control memory of a processor have a width of 26 bits. Each micro-instruction 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



Chapter 3

Memory Interface, I/O Interface

LEARNING OBJECTIVES

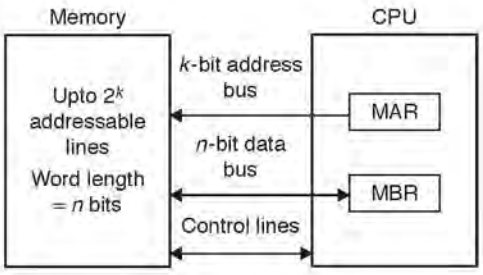
- *Memory interface*
- *RAM*
- *ROM*
- *Memory interfacing*
- *Input–output interfacing*
- *Handshaking*
- *Data transfer mode*
- *Design techniques for interrupts*
- *Direct memory access*
- *Input–output processor*

Memory Interface

Basic Concepts

Computer memory is used to store programs and data. The maximum size of a memory that can be used in any computer is determined by the addressing scheme.

Example: If the memory address has 16-bits, then the size of memory will be 2^{16} Bytes.



If MAR is k-bits long and MDR is n-bits long, then the memory may contain up to 2^k addressable locations and the n-bits of data are transferred between the processor and memory. This transfer takes place over processor bus. The processor bus has

1. Address line
2. Data line
3. Control line

Control line is used for coordinating data transfer.

Processor reads the data from the memory by loading the address of the required memory location into MAR and setting the $R/\overline{W}$ line to 1.

The memory responds by placing the data from the addressed location onto the data lines and confirms the actions. Upon confirmation, the processor loads the data onto the data lines, into MDR register. The processor writes the data into the memory location by loading the address of this location into MAR and loading the data into MDR sets the $R/\overline{W}$ line to 0.

- **Memory Access Time:** It is the time that elapses between the initiation of an operation and the completion of that operation.
- **Memory Cycle Time:** It is the minimum time delay that required between the initiations of two successive memory operations.

RAM (Random Access Memory)

In RAM, if any location that can be accessed for a read/write operation in fixed amount of time, it is independent of the location's address:

- Memory cells are usually organized in the form of array, in which each cell is capable of storing one bit of information.
- Each row of cells constitutes a memory word and all cells of a row are connected to a common line called as word line.
- The cells in each column are connected to sense/write circuit by two bit lines.

The data input and data output of each sense/write circuit are connected to a single bidirectional data line that can be connected to a data bus.

Chapter 4

Instruction Pipelining

LEARNING OBJECTIVES

- *Flynn's classification*
- *Pipelining*
- *Six-stages of pipelining*
- *Pipeline performance*
- *Pipeline hazards*
- *Structural hazards*
- *Data hazards*
- *Control hazards*
- *Conditional branch*
- *Dealing with branches*

Flynn's Classification

In parallel processing, the system is able to perform concurrent data processing to achieve faster execution time. A classifi cation introduced by M.J. Flynn considers the organization of a computer system by the number of instructions and data items that are manipulated simultaneously. According to this classifi cation, there are four major groups of computers:

1. **Single instruction stream, single data stream (SISD):**
   - Single computer containing a control unit, a processor unit and a memory unit
   - Instructions executed sequentially; parallel processing may be achieved by multiple functional units or by pipeline processing.
2. **Single instruction stream, Multiple data stream (SIMD):**
   - Include many processing units under the supervision of a common control unit.
   - All processors receive the same instruction from the control unit but operate on different items of data.
3. **Multiple instruction stream, Single data stream (MISD):**
   - No practical system has been constructed.
4. **Multiple instruction stream, Multiple data stream (MIMD):**
   - Capable of processing several programs at the same time.

One type of parallel processing that does not fit Flynn's classification is pipelining.

Pipelining

Pipelining is a technique of decomposing a sequential process into sub operations, with each subprocess being executed in special dedicated segment that operates with all other segments.

In pipelining, new inputs are accepted at one end before previously accepted inputs appear as outputs at the other end.

Two-stage Pipeline

As a simple approach, consider subdividing instruction processing into two stages:

1. Fetch instruction
2. Execute instruction
   - There are times during the execution of an instruction when main memory is not being accessed. This time can be used to fetch the next instruction in parallel with the execution of the current one. This is called instruction prefetch or fetch overlap.
   - This process will speed up instruction execution. If the fetch and execute stages were of equal duration, the instruction cycle time would be halved.
   - But there are some problems in this technique:
     (i) The execution time is generally longer than the fetch time.
     (ii) A conditional branch instruction makes the address of the next instruction to be fetched unknown.
   - These two factors reduce the potential effectiveness of the two-stage pipeline, but some speed up occurs. To gain further speed up, the pipeline must have more stage(s).

Six-stage Pipeline

Let the six stages be

F: Fetch Instruction
D: Decode Instruction
C: Calculate Operand Address
O: Operand Fetch

17. Consider the execution of 1000 instructions on a five-stage pipeline machine. Then the speed-up due to the use of pipelining given that the probability of an instruction being a branch is 0.2.
(A) 1.77 (B) 2.6
(C) 2.77 (D) 3.2

18. If an instructions following a branch (taken or not taken) have a dependency on the branch and cannot be executed until the branch is executed, then the dependency is
(A) True data dependency
(B) Procedural dependency
(C) Resource conflict
(D) Output dependency

19. 'A two-stage instruction pipeline unlikely to cut the instruction cycle time in half, compared with the use of no pipeline.' The statement is
(A) Always true (B) Always False
(C) Can't predict (D) Some times true

20. Write after read dependency is also known as
(A) True dependency (B) Anti-dependency
(C) Output dependency (D) Inverse dependency

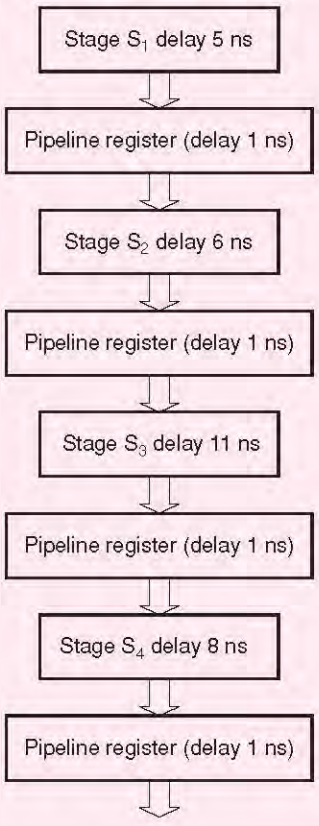
Selected Questions

1. Consider a 6-stage instruction pipeline, where all stages are perfectly balanced. Assume that there is no cycle time overhead of pipelining. When an application is executing on this 6-stage pipeline, the speedup achieved with respect to non-pipelined execution if 25% of the instructions incur 2 pipeline stall cycles is _____.

2. Consider the following processors (ns stands for nano-seconds). Assume that the pipeline registers have zero latency.
P1: Four-stage pipeline with stage latencies 1 ns, 2 ns, 2 ns, 1 ns.
P2: Four-stage pipeline with stage latencies 1 ns, 1.5 ns, 1.5 ns, 1.5 ns.
P3: Five-stage pipeline with stage latencies 0.5 ns, 1 ns, 1 ns, 0.6 ns, 1 ns.
P4: Five-stage pipeline with stage latencies 0.5 ns, 0.5 ns, 1 ns, 1 ns, 1.1 ns.
Which processor has the highest peak clock frequency?
(A) P1 (B) P2
(C) P3 (D) P4

3. An instruction pipeline has five stages, namely, instruction fetch (IF), instruction decode and register fetch (ID/RF), instruction execution (EX), memory access (MEM), and register write back (WB) with stage latencies 1 ns, 2.2 ns, 2 ns, 1 ns, and 0.75 ns, respectively (ns stands for nano seconds). To gain in terms of frequency, the designers have decided to split the ID/RF stage into three stages (ID, RF1, RF2) each of latency 2.2/3 ns. Also, the EX stage is split into two stages (EX1, EX2) each of latency 1 ns. The new design has a total of eight pipeline stages. A program has 20% branch instructions which execute in the EX stage and produce the next instruction pointer at the end of the EX stage in the old design and at the end of the EX2 stage in the new design. The IF stage stalls after fetching a branch instruction until the next instruction pointer is computed . All instructions other than the branch instruction have an average CPI of one in both the designs. The execution times of this program on the old and the new design are P and Q nanoseconds, respectively. The value of P/Q is _____.

4. Consider an instruction pipeline with four stages (S_1, S_2, S_3 and S_4) each with combinational circuit only. The pipeline registers are required between each stage and at the end of the last stage. Delays for the stages and for the pipeline registers are as given in the figure.



Digital Logic

UNIT 3

Chapter 2

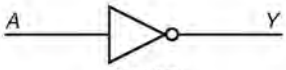
Boolean Algebra and Minimization of Functions

LEARNING OBJECTIVES

- *Logic gates*
- *Boolean algebra*
- *AXIOMS and Laws of Boolean algebra*
- *Properties of Boolean algebra*
- *Conversion from Min term to Max term*
- *Minimization of Boolean function*
- *K-map method*
- *Prime implicant*
- *Implementation of function by using NAND-NOR Gates*
- *EX-OR, EX-NOR GATE*

Logic Gates

1. **Inverter or NOT gate (7404 IC):** The inverter performs a basic logic operation called inversion or complementation. The purpose of the inverter is to change one logic level to the opposite level. In terms of digital circuits, it converts 1 to 0 and 0 to 1.



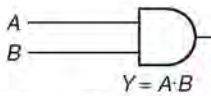
Symbol

$Y = \overline{A}$

Table 1 *Truth Table*

Input	Output
A	Y
0	1
1	0

2. **AND gate (logical multiplier 7408 IC):** The AND gate performs logical multiplication more commonly know as AND function. The AND gate is composed of 2 or more inputs and a single output



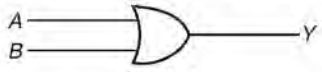
$Y = A \cdot B$

Figure 1 2 input AND gate

Table 2 *Truth Table*

Input		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

3. **OR gate (logical adder 7432 IC):** The OR gate performs logical, addition commonly known as OR function.



Symbol

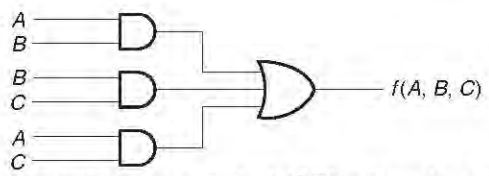
$Y = A + B$

Figure 2 2 input OR gate

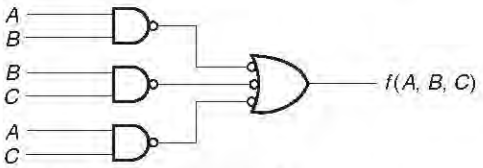
Table 3 *Truth Table*

Input		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Solution:



By considering bubbles at output of AND gate and input of OR gate.

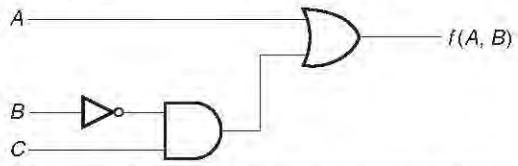


So four NAND gates are required.

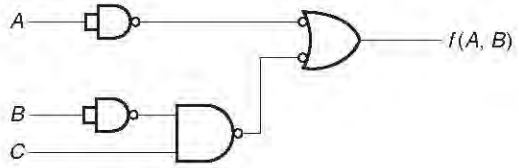
Example 17: Number of NAND gates required for implementation of $f(A, B) = A + \overline{B}C$ is

(A) 3 (B) 4 (C) 5 (D) 6

Solution:



To convert the all gates into NAND gates, place bubble output of AND, and inputs of OR gates. Now, the circuit can be drawn as



Four NAND gates are required.

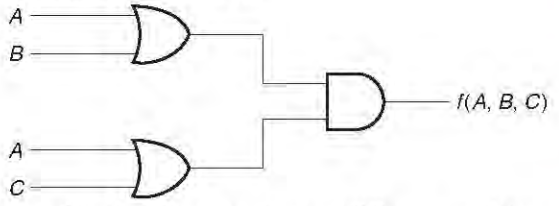
Example 18: $f = A + BC$, the number of NOR gates required to implement f, are?

(A) 3 (B) 4 (C) 5 (D) 2

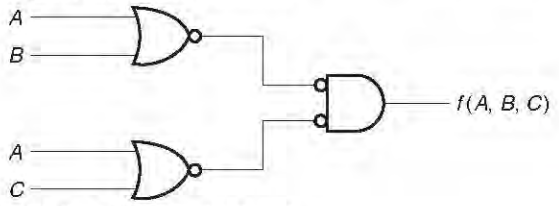
Solution: $A + BC$ is in SOP form.

To implement this function by using NOR gates, we can write $f(A, B, C) = A + BC = (A + B)(A + C)$

Which is in the form of POS?



By including bubbles at output of OR gate, and input of AND gate, the circuit becomes



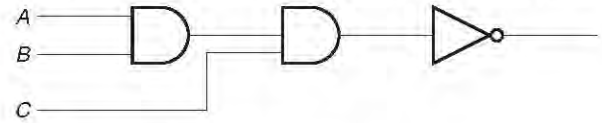
Now the circuit consists of all NOR gates. Three NOR Gates are required.

Example 19: How many number of two-input NAND–NOR gates are required to implement three-input NAND–NOR gates respectively?

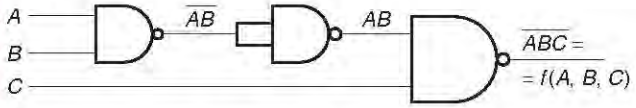
(A) 2, 2 (B) 2, 3
(C) 3, 2 (D) 3, 3

Solution: $f(A, B, C) = \overline{ABC} = \overline{AB} + \overline{C}$

(1) Implement above function by using two-inputs gates

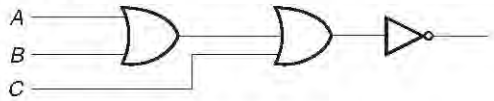


Now convert each gate to NAND gate

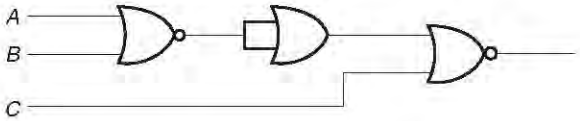


Three two-input NAND gates are required.

(2) $G(A, B, C) = \overline{A + B + C}$ Implement it by using two-input gates



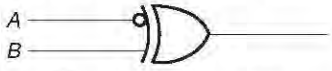
Now convert each gate to NOR gate



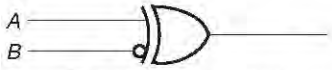
Three two-input, NOR gates are required.

EX-OR, EX-NOR Gates

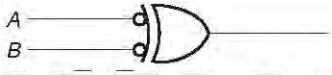
Inverted inputs for EX OR, EX-NOR gates



$$\overline{A} \oplus B = \overline{\overline{A}B} + \overline{A}\,\overline{B} = AB + \overline{A}\,\overline{B} = A \odot B$$



$$A \oplus \overline{B} = A\overline{\overline{B}} + \overline{A}\,\overline{B} = AB + \overline{A}\,\overline{B} = A \odot B$$



$$\overline{A} \oplus \overline{B} = \overline{A}\,\overline{\overline{B}} + \overline{\overline{A}}\,\overline{B} = \overline{A}B + A\overline{B} = A \oplus B$$

$$\overline{A} \odot B = \overline{A}B + \overline{\overline{A}}\,\overline{B} = \overline{A}B + A\overline{B} = A \oplus B$$

(C) Both S_1 and S_2 are true
(D) Neither S_1 nor S_2 are true

22. Given the function $F = P^1 + QR$, where F is a function in three Boolean variables P, Q and R and $P^1 = !P$, consider the following statements.
(S_1) $F = \Sigma(4, 5, 6)$
(S_2) $F = \Sigma(0, 1, 2, 3, 7)$
(S_3) $F = \pi(4, 5, 6)$
(S_4) $F = \pi(0, 1, 2, 3, 7)$
Which of the following is true?

(A) (S_1) – False, (S_2) – True, (S_3) – True, (S_4) – False
(B) (S_1) – True, (S_2) – False, (S_3) – False, (S_4) – True
(C) (S_1) – False, (S_2) – False, (S_3) – True, (S_4) – True
(D) (S_1) – True, (S_2) – True, (S_3) – False, (S_4) – False

23. The total number of prime implicants of the function $f(w, x, y, z) = \Sigma(0, 2, 4, 5, 6, 10)$ is _____

24. Consider the Boolean operator # with the following properties:
$x \# \underline{0} = x$, $x \# 1 = \times$, $x \# x = 0$ and
$x \# \times = 1$. Then $x \# y$ is equivalent to
(A) $x\ \overline{y} + \overline{\times}\ y$ (B) $x\ \overline{y} + \overline{\times}\ \overline{y}$
(C) $\overline{\times}\ y + x\ y$ (D) $x\ y + \overline{\times}\ \overline{y}$

25. Consider the Karnaugh map given below, where X represents *"don't care"* and blank represents 0.

dc \ ba	00	01	11	10
00		X	X	
01	1			X
11	1			1
10		X	X	

Assume for all inputs (a, b, c, d), the respective complements $(\overline{a}, \overline{b}, \overline{c}, \overline{d})$ are also available. The above logic is implemented using 2-input NOR gates only. The minimum number of gates required is _____.

26. If w, x, y, z are Boolean variables, then which one of the following is INCORRECT?
(A) $wx + w(x + y) + x(x + y) = x + wy$
(B) $\overline{w\overline{x}(y + \overline{z})} + \overline{w}x = \overline{w} + x + \overline{y}z$
(C) $\left(w\overline{x}\left(y + x\overline{z}\right) + \overline{w}\overline{x}\right)y = \overline{x}\,\overline{y}$
(D) $(w + y)(wxy + wyz) = wxy + wyz$

27. Given $f(w, x, y, z) = \sum m(0,1,2,3,7,8,10) + \sum d(5,6,11,15)$, where d represents the *don't-care* condition in Karnaugh maps. Which of the following is a mini-mum product-of-sums (POS) form of $f(w, x, y, z)$?

(A) $f = (\overline{w} + \overline{z})(\overline{x} + z)$
(B) $f = (\overline{w} + z)(x + z)$
(C) $f = (w + z)(\overline{x} + z)$
(D) $f = (w + \overline{z})(\overline{x} + z)$

28. Let $\oplus$ and $\odot$ denote the Exclusive OR and Exclusive NOR operations, respectively. Which one of the following is NOT CORRECT?
(A) $\overline{P \oplus Q} = P \odot Q$
(B) $\overline{P} \oplus Q = P \odot Q$
(C) $\overline{P} \oplus \overline{Q} = P \oplus Q$
(D) $(P \oplus \overline{P}) \oplus Q = (P \odot \overline{P}) \odot \overline{Q}$

29. Consider the minterm list form of a Boolean function F given below.

$$F(P, Q, R, S) = \sum m(0, 2, 5, 7, 9, 11) + d(3, 8, 10, 12, 14)$$

Here, m denotes a minterm and d denotes a don't care term. The number of essential prime implicants of the function F is _____.

Chapter 3

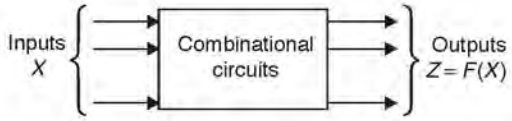
Combinational Circuits

LEARNING OBJECTIVES

- *Combinational logic design*
- *Arithmetic circuit*
- *Half adder*
- *Full adder*
- *Half subtractor*
- *Full subtractor*
- *n-bit comparator*
- *Parity bit generator and parity bit checker*
- *Code converter*
- *Decoder*
- *Designing high order decoders from lower order decoder*
- *Combinational logic implementation*
- *Encoders*
- *Multiplexer*
- *Demultiplexer*

Introduction

Combinational logic is a type of logic circuit whose output is a function of the present input only.



Combinational Logic Design

The design of combinational circuit starts from the problem, statement and ends with a gate level circuit diagram.

The design procedure involves the following steps:

1. Determining the number of input variables and output variables required, from the specifications.
2. Assigning the letter symbols for input and output.
3. Deriving the truth table that defines the required relationship between input and output.
4. Obtaining the simplified Boolean function for each output by using K-map or algebraic relations.
5. Drawing the logic diagram for simplified expressions.

We will discuss combinational circuits under the following categories:

- Arithmetic circuits
- Code converters
- Data processing circuits

Arithmetic Circuits

Arithmetic circuits are the circuits that perform arithmetic operation. The most basic arithmetic operation is addition.

Half Adder

Addition is an arithmetic operation, and here to implement addition in digital circuits we have to implement by logical gates. So the addition of binary numbers will be represented by the logical expressions. Half adder is an arithmetic circuit which performs the addition of two binary bits, and the result is viewed in two output—sum and carry.

The sum 'S' is the X-OR of 'A' and 'B' where A and B are inputs.

$$\therefore \quad S = A\overline{B} + B\overline{A} = A \oplus B$$

The carry 'C' is the AND of A and B.

$$\therefore C = AB$$

Table 1 *Truth Table*

Inputs		Outputs	
A	**B**	**S**	**C**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

So, half adder can be realized by using one X-OR gate and one AND gate.

the function, together with an external OR gate that forms their logical sum, provides a hardware implementation of the function.

Similarly, any function with n inputs and m outputs can be implemented with $n \times 2^n$ decoders and m OR gates.

Example 3: Implement full adder circuit by using 2 × 4 decoder.

Sum = Σ (1, 2, 4, 7), Carry = Σ (3, 5, 6, 7)

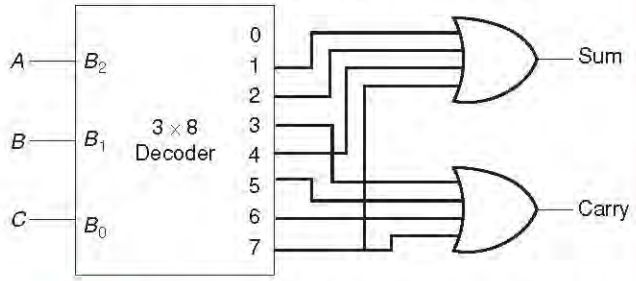


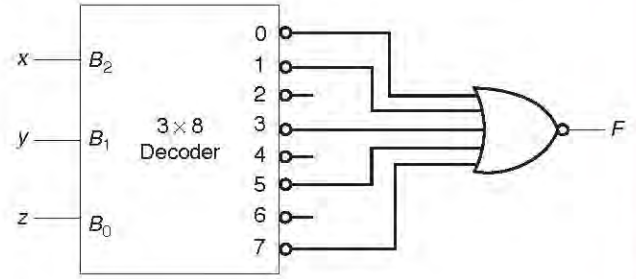
Figure 14 Implementation of full adder circuit with decoder

The 3 × 8 decoder generates the 8 minterms for A, B, and C. The OR gate for output sum forms the logical sum of minterms 1, 2, 4 and 7. The OR gate for output carry forms the logical sum of minterms 3, 5, 6 and 7.

Example 4: The minimized SOP form of output $F(x, y, z)$ is

(A) $x'y + z'$ (B) $x'y' + z'$

(C) $x'y' + z'$ (D) $x' + y'z$



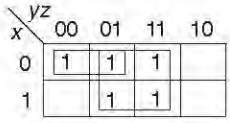
Solution: (C)

The outputs of decoder are in active low state. So, we can express outputs as $\overline{Y_7}, \overline{Y_6} \cdots \overline{Y_0}$

Outputs 0, 1, 3, 5, 7 are connected to NAND gate to form function $F(x, y, z)$

So
$$F = \overline{\overline{Y_0} \cdot \overline{Y_1} \cdot \overline{Y_3} \cdot \overline{Y_5} \cdot \overline{Y_7}}$$
$$= Y_0 + Y_1 + Y_3 + Y_5 + Y_7$$
$$= \Sigma(0, 1, 3, 5, 7)$$

By using K-maps

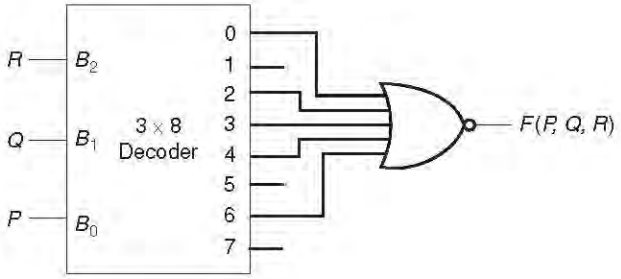


$F = z + x'y'$

Example 5. The minimal *POS* form of output function $f(P, Q, R)$ is

(A) $P\overline{Q} + PR$ (B) $P + \overline{Q}R$

(C) $P(\overline{Q} + R)$ (D) $Q(\overline{P} + R)$



Solution: (C)

The outputs of decoder are in normal form. 0, 2, 3, 4, 6 outputs are connected to NOR gate to form $F(P, Q, R)$

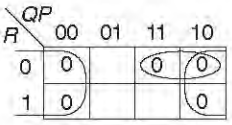
So
$$F = \overline{Y_0 + Y_2 + Y_3 + Y_4 + Y_6}$$
$$= \overline{Y_0} \cdot \overline{Y_2} \cdot \overline{Y_3} \cdot \overline{Y_4} \cdot \overline{Y_6}$$

$Y_0, Y_1, \ldots, Y_7$ indicate minterms, whereas $\overline{Y_0}, \overline{Y_1}, \cdots, \overline{Y_7}$ are maxterms.

So $F = \pi(0, 2, 3, 4, 6)$

Here, from the decoder circuit MSB is R, LSB is P.

By using K-map



$$F(P, Q, R) = P(R + \overline{Q})$$

ENCODERS

It is a digital circuit that performs the inverse operation of a decoder.

An encoder has 2^n (or fewer) input lines and n output lines.

It is also known as an octal to binary converter.

Consider an 8–3 line encoder:

Table 8 *Truth Table*

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	1	0	0	1	1
0	0	0	0	1	0	0	1	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

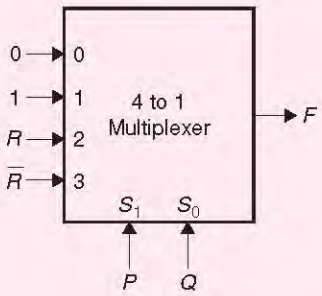
10. The amount of ROM needed to implement a 4-bit multiplier is
(A) 64 bits
(B) 128 bits
(C) 1K bits
(D) 2K bits

11. In the following truth table, $V = 1$ if and only if the input is valid.

Inputs				Outputs		
D_0	D_1	D_2	D_3	X_0	X_1	V
0	0	0	0	x	x	0
1	0	0	0	0	0	1
x	1	0	0	0	1	1
x	x	1	0	1	0	1
x	x	x	1	1	1	1

What function does the truth table represent?
(A) Priority encoder
(B) Decoder
(C) Multiplexer
(D) Demultiplexer

12. Consider the 4-to-1 multiplexer with two select lines S_1 and S_0 given below.



The minimal sum-of-products form of the Boolean expression for the output F of the multiplexer is

(A) $\overline{P}Q + Q\overline{R} + P\overline{Q}R$
(B) $\overline{P}Q + \overline{P}Q\overline{R} + PQ\overline{R} + P\overline{Q}R$
(C) $\overline{P}QR + \overline{P}Q\overline{R} + Q\overline{R} + P\overline{Q}R$
(D) $PQ\overline{R}\ \overline{P}QR + \overline{P}Q\overline{R} + Q\overline{R} + P\overline{Q}R$

13. Consider the following combinational function block involving four Boolean variables x, y, a, b, where x, a, b are inputs and y is the output.

$f(x, y, a, b)$
{
if (x is 1) $y = a$;
else $y = b$;
}

Which one of the following digital logic blocks is the most suitable for implementing this function?
(A) Full adder
(B) Priority encoder
(C) Multiplexer
(D) Flip–flop

14. Let $\oplus$ denote the Exclusive OR (X-OR) operation. Let '1' and '0' denote the binary constants. Consider the following Boolean expression for F over two variables P and Q:

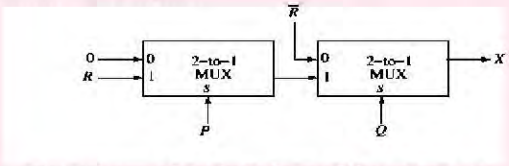
$F(P, Q) = ((1 \oplus P) \oplus (P \oplus Q)) \oplus ((P \oplus Q) \oplus (Q \oplus 0))$

The equivalent expression for F is

(A) $P + Q$
(B) $\overline{P + Q}$
(C) $P \oplus Q$
(D) $\overline{P \oplus Q}$

15. A half adder is implemented with XOR and AND gates. A full adder is implemented with two half adders and one OR gate. The propagation delay of an XOR gate is twice that of an AND/OR gate. The propagation delay of an AND/OR gate is 1.2 microseconds. A 4-bit ripple-carry binary adder is implemented by using four full adders. The total propagation time of this 4-bit binary adder in microseconds is _____

16. Consider the two cascaded 2-to-1 multiplexers as shown in the figure.



minimal sum of products form of the output x is

The minimal sum of products form of the output X is

(A) $\overline{P}\ \overline{Q} + PQR$
(B) $\overline{P}\ Q + QR$
(C) $PQ + \overline{P}\ \overline{Q}\ R$
(D) $\overline{P}\ \overline{Q} + PQR$

17. When two 8-bit numbers $A_7 \ldots A_0$ and $B_7 \ldots B_0$ in 2's complement representation (with A_0 and B_0 as the least significant bits) are added using a **ripple-carry adder**, the sum bits obtained are $S_7 \ldots S_0$ and the carry bits are $C_7 \ldots C_0$. An overflow is said to have occurred if
(A) the carry bit C_7 is 1
(B) all the carry bits $(C_7, \ldots, C_0)$ are 1
(C) $(A_7 \cdot B_7 \cdot \overline{S_7} + \overline{A_7} \cdot \overline{B_7} \cdot S_7)$ is 1
(D) $(A_0 \cdot B_0 \cdot \overline{S_0} + \overline{A_0} \cdot \overline{B_0} \cdot S_0)$ is 1

Chapter 4

Sequential Circuits

LEARNING OBJECTIVES

- *Sequential circuit*
- *Basic storage elements*
- *Latches (SR Latch, D Latch, JK Latch)*
- *Flip-flops (JK flip-flop, T flip-flop, D flip-flop)*
- *Counters*
- *Asynchronous counter design*
- *Synchronous counter design*
- *Registers*
- *Various types of registers*
- *Application of shift register*

Sequential Circuits

In sequential circuits the output depends on the input as well as on the previous history of output, i.e., they contain memory elements.

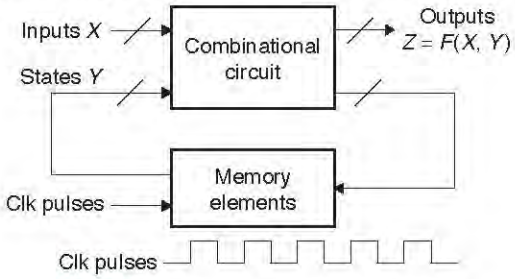


Figure 1 Block diagram of sequential circuit

Table 1 *Comparison between combinational and sequential circuits*

Combinational Circuits	Sequential Circuits
1. Output at any time depends on the combine set of input applied to it simultaneously at that instant of time	Output depends on the present input as well as on the previous history of output
2. Contains no memory element	Contains at least one memory element
3. Easy to design due to absence of memory	Difficult to design
4. Totally described by the set of output values	Its performance is totally described by the set of subsequent values as well as set of output values
5. Faster in speed because all inputs are primary inputs and applied simultaneously	Slower in speed because secondary inputs are also needed which are applied after delay
6. It need more hardware for realization	Less hardware required
7. Expensive in cost	Cheap in cost

Sequential circuits are of two types:

1. Clocked or synchronous
2. Unclocked or asynchronous

In synchronous sequential circuits the logic circuits action is allowed to occur in synchronization with the input clock pulse from a system clock.

In asynchronous sequential circuits the logic sequential action is allowed to occur at any time.

Basic Storage Elements

Latches and fl ip-fl ops

A storage element in digital circuit can maintain a binary state indefi n itely until directed by an input signal to switch states. Storage elements that operate with signal levels (i.e., level triggering of signal inputs) are referred to as latches. Those controlled by a clock transition (i.e., edge triggering) are fl ip-fl ops.

Latches and fi lp-fl ops are related because latches are basic circuits from which all fl ip-fl ops are constructed. Latches are useful for storing binary information and for the design of asynchronous sequential circuits. But latches are not practical for use in synchronous sequential circuits, so we use fl ip-fl ops.

Flip-fl ops

They are also known as bistable multivibrators. This is a basic memory element to store 1-bit of information 0 or 1 and is used in storage circuits, counters, shift register, and many other computer applications. It has two stable states: 1 and 0. The high state is called set state and zero as reset.

It has two outputs one being the complement of the other usually designated by Q and $\overline{Q}$.

Networks, Information Systems, Software Engineering and Web Technology

UNIT 4

Part A Network

Has arbitrary number of bits and allows character codes with an arbitrary number of bits per character. Every frame begins and ends with a special bit pattern, 01111110, called a flag byte.

As soon as the sender's data link layer encounters five consecutive one's in the data, it stuffs a 0 bit into the outgoing bit stream.

Receiver de-shifts the 0 bit of the five consecutive incoming 1 bits, followed by a 0 bit.

If the user data is 01111110, transmitted as 011111010 but stored at receiver as 01111110.

Physical layer coding violations Applied to the networks in which the encoding on the physical medium contains some redundancy.

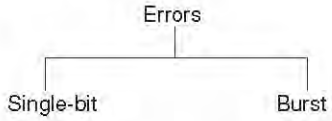
1 → high – low pair

0 → low – high pair

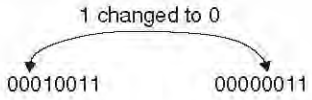
Here high-high, low-low not used for data.

Every data bit has a transition in the middle, thus easy for the receiver to locate the bit boundaries.

Types of Errors



Single bit error The term single bit error means that only one bit in the data unit has changed, it can either be from 1 to 0 or from 0 to 1.



Single bit error correction

A single bit error occurs when a bit changes in value from 0 to 1 (or) from 1 to 0 while storing (or) while performing read (or) write operation. If that error bit is identified, that can be corrected by complementing.

Hamming codes

In hamming codes, K parity bits are added to an n-bit data word, that forms a new word of $(n + k)$ bits. The bit positions are numbered in sequence from 1 to $n + k$. These positions numbered with powers of 2 are reserved for the parity bits; the remaining bits are the data bits.

Example: Consider the given 8-bit data word 11000100, we include four party bits with this word and arrange the bits as follows.

Bit position

1	2	3	4	5	6	7	8	9	10	11	12
$P1$	$P2$	1	$P4$	1	0	0	$P8$	0	1	0	0

The parity bits are in positions, 1, 2, 4, 8. Each parity bit is calculated as

$P1$ = XOR of bits (3, 5, 7, 9, 11) = $1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 0$

$P2$ = XOR of bits (3, 6, 7, 10, 11) = 0

$P4$ = XOR of bits (5, 6, 7, 12) = 1

$P8$ = XOR of bits (9, 10, 11, 12) = 1

⇒ If there is odd number of 1s, XOR gives 0

⇒ If there is even number of 1s, XOR gives 1

The values $P1 = 0$, $P2 = 0$, $P4 = 1$, $P8 = 1$ are substituted in 12-bit composed word

Bit position

1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	1	1	0	0	1	0	1	0	0

Check for errors:

$C1$ = XOR of bits (1, 3, 5, 7, 9, 11)

$C2$ = XOR of bits (2, 3, 6, 7, 10, 11)

$C4$ = XOR of bits (4, 5, 6, 7, 12)

$C8$ = XOR of bits (8, 9, 10, 11, 12)

Since the bits were written with even parity, the result $C = C8\ C4\ C2\ C1 = 0000$

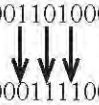
∴ Indicates that no error has occurred.

- The code can be used with words of any length.

Burst Error The term burst means that two or more bits in the data unit have changed, either changed, from 1 to 0 or changed from 0 to 1.

Sent:

010011010000-sent bits corrupted by burst error



010001111000 Received

Parity bit

Parity bit is an error detecting code. This bit is added to data words depending on number of 1's in the data word; It could be even parity and odd parity.

n-bit data word is transformed to $(n + 1)$ bit code word with the addition of a bit. Even parity makes even number of 1's in a code word, similarly odd parity makes odd number of 1's in a code word.

Let us illustrate with example

Data word – 1 0 1 1 Parity bit

Code word – 1 0 1 1 1 parity bit (even parity)

Code word: 1 0 1 1 0 (odd parity)

At the receiver side, when the code is received, the receiver checks the same as it is done by the generator. But here it adds all the bits which results in syndrome. If the syndrome is 0 then the number of 1's in code word is even, else number of 1's is odd.

Decision logic analyzer will decide, whether the code word is correct or not, based on syndrome value.

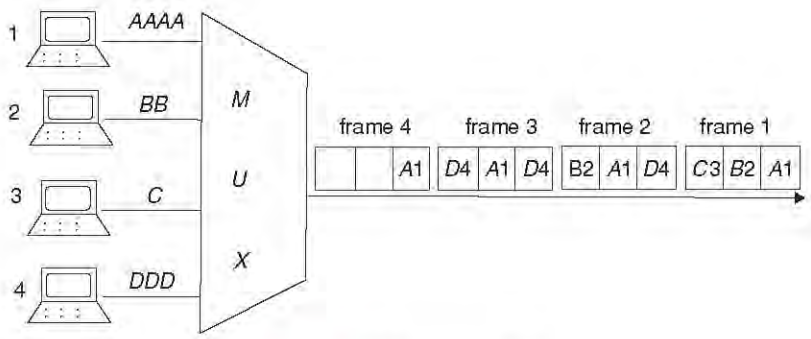


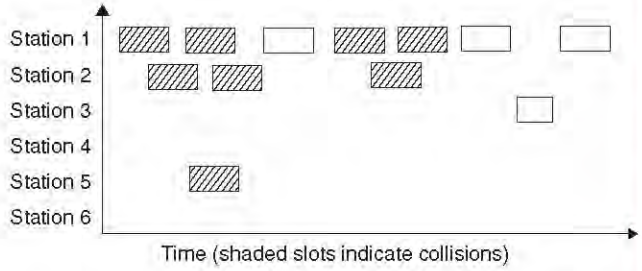
Figure 6 Asynchronous TDM

Aloha Protocols

The Aloha Protocol was designed to provide data transmission between computers on several islands using radio transmission.

Pure aloha

Pure Aloha is an unslotted, fully decentralized protocol. It is extremely simple and trivial to implement. The ground rule is 'when you want to talk, just talk!' So, a node which wants to transmit, will go ahead and sends the packet on its broadcast channel, with no consideration of who so ever to any body else is transmitting (or) (not).

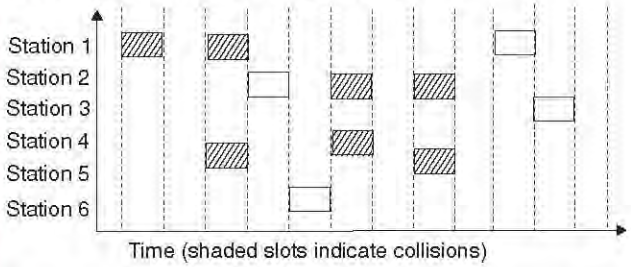


One serious drawback here is that, you don't know whether what you are sending, has been received properly or not. To resolve this in pure Aloha, when one node finishes speaking it expects an acknowledgement in a finite amount of time otherwise it simply retransmits the data. This scheme works well in small networks where the load is not high. But in large, load intensive networks where many nodes may want to transmit at the same time, this scheme fails miserably. This led to the development of slotted Aloha.

Slotted Aloha

This is quite similar to pure Aloha, differing only in the way transmissions take place. Instead of transmitting right at the demand time, the sender waits for some time. This delay is specified as follows–the timeline is divided into equal slots and then it is required that transmission should take place only at slot boundaries. To be more precise, the slotted Aloha makes the following assumptions.

- All frames consist of exactly L bits.
- Time is divided into slots of size L/K seconds. (i.e., a slot equals the time to transmit one frame)
- Nodes start to transmit frames only at the beginning of slots.
- The nodes are synchronized so that each node knows when the slots begin.
- If two or more frames collide in a slot, then all the nodes detect the collision event before slot ends.



In this, way the number of collisions that can possibly take place is reduced by a huge margin. And hence, the performance became much better compared to pure Aloha. Collisions may only take place with nodes that are ready to speak at the same time.

Virtual private network

Virtual Private Networking (VPN) Internet protocol security (IP sec) is one of the most complete, secure, standards-based protocol developed for transporting data.

A VPN is a shared network, where private data can be accessed only by the intended recipient.

The term VPN is used to describe a secure connection over the Internet.

VPN is also used to describe private networks such as Frame Relay and Asynchronous Transfer Mode (ATM).

The purpose of data security is that the data flowing across the network is protected by encryption technologies.

IP sec-based VPNs use encryption to provide data security, that increases the networks resistance to data tampering.

IP sec-based VPNs can be created over any type of IP Network, including Internet, ATM, Frame Relay, among all only Internet is inexpensive.

Uses of VPN

Intranets Intranets connect an organization's locations. These locations could be head quarters offices, branch offices, Employees home which is located in some Remote area.

Chapter 2

Routing Algorithms

LEARNING OBJECTIVES

- *Routing algorithm basics*
- *Flooding*
- *Multipath routing*
- *Distance vector routing*
- *Link state routing*
- *Hierarchical routing*
- *Rip*
- *Ospf*
- *Congestion control techniques*
- *Traffic shaping*

Routing Algorithms Basics

The main function of network layer is routing packets from the source machine to the destination machine. The routing algorithms are part of the network layer software, responsible for deciding which output line an incoming packet should be transmitted on.

Routing algorithms can be grouped into two major classes: Non-adaptive and Adaptive.

1. Non-adaptive algorithms do not base their routing decisions on measurements or estimates of the current traffic and topology. Instead, the choice of the route to use is downloaded to the routers when the network is booted. This procedure is called static routing.
2. Adaptive algorithms, in contrast, change their routing decisions to reflect changes in the topology, and the traffic as well.

Store and Forward Packet Switching

In this Technique, the data packet will be stored at the node and it is forwarded to its next appropriate intermediate node. The next intermediate node will first store the packet in the buffer, based on the router decision, it selects an interface, and forwards to receiver.

The technique is most suitable for the networks with unsteady connectivity.

The length of the packet we take shows effect on the file transfer, if the data packet is small, in the store the forward, delay will be less at each node, but causes extra overhead with headers. So, the packet size selection should be done appropriately.

Flooding

Static algorithms, in which every incoming packet is sent out on every outgoing line except the one on which it is arrived. Header contains the hop count of each packet. Hop counter is decremented at each hop, with the packet being discarded when the counter reaches zero.

Another way for damming the flood is to keep track of which packets have been flooded, to avoid sending them out a second time. A variation of flooding that is slightly more practical is selective flooding. In this algorithm the routers do not send every incoming packet out on every line, only on those lines that are going approximately in the right direction.

Multipath Routing

Multipath routing is routing the packets from the source, on multiple paths to the destination. It is nothing but spreading the traffic.

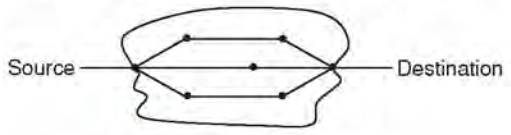


Figure 1 Multipath routing model

Single path routing causes QOS, throughput and delay problems, and multipath routing, improves network performance with sharing of available resources of network.

The components of multipath routing are

1. Multipath calculation algorithm
2. Multipath forwarding algorithm
3. End-Host protocol

Chapter 3

TCP/UDP

LEARNING OBJECTIVES

- *Transport layer*
- *User Diagram Protocol (UDP)*
- *TCP/IP*
- *TCP/IP vs OSI reference model*
- *TCP state transition diagram*
- *TCP flow control*
- *Application layer*
- *ICMP, SMTP, POP3, IMAP 4, HTTP, FTP*
- *DNS*
- *Network devices*

Transport Layer

Real communication takes place between two applications programs i.e., processes. For this, process-to-process delivery is needed. A mechanism is required in order to deliver data from one of these processes running on the source host to the corresponding process running on the destination host.

The transport layer is responsible for process-to-process delivery.

Addressing in Transport Layer

Port addresses

- A transport layer address is a port number.
- The destination port number is needed for delivery and the source port number is needed for reply.
- The port numbers are 16-bit integers ranging from 0 to 65535.

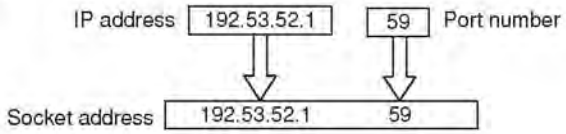
The IANA (Internet Assigned Number Authority) has divided the port numbers as:

- Well-known ports (0 to 1023)
- Registered ports (1024 to 49,151)
- Dynamic or private or ephemeral ports (49,152 to 65,535)

Socket address

Process to process delivery needs two identifiers, IP address and port address at each end to make a connection.

The combination of an IP address and a port number is socket address.



Protocols at transport layer

1. UDP
2. TCP
3. SCTP

User Datagram Protocol (UDP)

- UDP is connectionless protocol.
  - There is no mechanism for connection establishment or connection termination.
  - The packets may be delayed or lost or may arrive out of sequence, i.e., there is no acknowledgement.
  - Each user datagram sent by UDP is an independent program. Even if the user datagram's are coming from the same source program and going to the same destination process, there is no relationship between the different datagrams.

  Thus, user datagrams can travel on a different path.
- Multicasting capability is embedded in UDP.
- It is a simple, unreliable transport protocol.
  - There is no flow control, no window mechanism.
  - There is no error control as well except for the checksum. The sender does not know if a message has been lost or duplicated. When the receiver detects an error through the checksum, the datagram is discarded silently.
- It is used in real-time applications.
  - The header length is fixed, of 8 bytes. Real time applications require a constant flow of data. Moreover, the unreliability (fast and less complex service) of UDP aids in real-time applications like voice over IP, online games etc.
- It encapsulates and decapsulates messages in an IP datagram.

Chapter 4

IP(v4)

LEARNING OBJECTIVES

- *IP addressing*
- *Class A*
- *Class B*
- *Class C*
- *Class D*
- *Subnet mask*
- *Classless Inter Domain Routing (CIDR)*
- *Network address*
- *Network Address Translation (NAT)*
- *IP-Protocol*

IP Addressing

Every machine on the internet has a unique identific ation number, called an IP Address. A typical IP address looks like this:

216.27.61.137

To make it easier for humans to remember, IP addresses are normally expressed in decimal format as a "*dotted decimal number*" like the one above. But computers communicate in binary form. Look at the same IP address in binary:

11011000.00011011.00111101.10001001

The four numbers in an IP address are called octets, because they each have eight positions when viewed in binary form. If you add all the positions together, you get 32, which is why IP addresses are considered 32-bit numbers. Since each of the eight positions can have two different states (1 or 0) the total number of possible combinations per octet is 2^8 or 256. So each octet can contain any value between 0 and 255. Combine the four octets and you get 2^{32} or a possible 4,294,967,296 unique values.

Out of the almost 4.3 billion possible combinations, certain values are restricted from use as typical IP addresses. For example, the IP address 0.0.0.0 is reserved for the default network and the address 255.255.255.255 is used for broadcasts.

The octets serve a purpose other than simply separating the numbers. They are used to create classes of IP addresses that can be assigned to a particular business, government or other entities based on size and need. The octets are split into two sections: Net and Host. The Net section always contains the fi rst octet. It is used to identify the network that a computer belongs to. Host (sometimes referred to as Node) identifies the actual computer on the network. The Host section always contains the last octet. There are five IP classes plus certain special addresses. They are

1. Class A
2. Class B
3. Class C
4. Class D
5. Class E

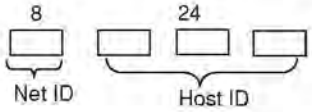
Default Network: The IP address of 0.0.0.0 is used for the default network.

Class A

This class is for very large networks, such as, a major international company. IP addresses with a fi rst octet from 1 to 126 are part of this class. The other three octets are used to identify each host. This means that there are 126 Class A networks each with 16,777,214 ($2^{24} - 2$) possible hosts for a total of 2,147,483,648 (2^{31}) unique IP addresses. Class A networks account for half of the total available IP addresses. In Class A networks, the high order bit value (the very fi rst binary number) in the fi rst octet is always 0.

Example:

Net	Host or Node
115.	24.53.107



Selected Questions

1. Suppose computers A and B have IP addresses 10.105.1.113 and 10.105.1.91 respectively and they both use the same net mask *N*. Which of the values of N given below should not be used if A and B should belong to the same network?

(A) 255.255.255.0 (B) 255.255.255.128
(C) 255.255.255.192 (D) 255.255.255.224

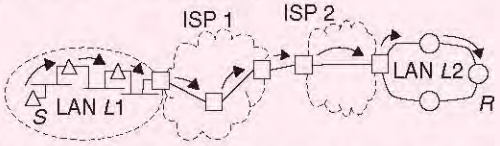
2. In the IPv4 addressing format, the number of networks allowed under class C addresses is

(A) 2^{14} (B) 2^{7}
(C) 2^{21} (D) 2^{24}

3. In an IPv4 datagram, the *M*-bit is 0, the value of HLEN is 10, the value of total length is 400 and the fragment offset value is 300. The position of the datagram, the sequence numbers of the first and the last bytes of the payload, respectively are

(A) Last fragment, 2400 and 2789
(B) First fragment, 2400 and 2759
(C) Last fragment, 2400 and 2759
(D) Middle fragment, 300 and 689

4. In the diagram shown below, *L*1 is an Ethernet LAN and *L*2 is a Token-Ring LAN. An IP packet originates from sender S and traverses to *R*, as shown. The links within each ISP and across the two ISPs, are all point-to-point optical links. The initial value of the TTL field is 32. The maximum possible value of the TTL field when R receives the datagram is _____.



5. Host *A* (on TCP/IPv4 network *A*) sends an IP datagram *D* to host *B* (also on TCP/IPv4 network *B*). Assume that no error occurred during the transmission of *D*. When *D* reaches *B*, which of the following IP header field(s) may be different from that of the original datagram *D*?

(i) TTL (ii) Checksum
(iii) Fragment Offset
(A) (i) only (B) (i) and (ii) only
(C) (ii) and (iii) only (D) (i), (ii), and (iii)

6. An IP router implementing Classless Inter-Domain Routing (CIDR) receives a packet with address 131.23.151.76. The router's routing table has the following entries:

Prefix	Out Interface Identifier
131.16.0.0/12	3
131.28.0.0/14	5
131.19.0.0/16	2
131.22.0.0/15	1

The identifier of the output interface on which this packet will be forwarded is _____.

7. Every host in an IPv4 network has a 1-second resolution real-time clock with battery backup, each host needs to generate up to 1000 unique identifiers per second. Assume that each host has a globally unique IPv4 address. Design a 50-bit globally unique ID for this purpose. After what period (in seconds) will the identifiers generated by a host wrap around?

8. Which one of the following fields of an IP header is NOT modified by a typical IP router?

(A) Checksum (B) Source address
(C) Time to Live (TTL) (D) Length

9. Consider the following routing table at an IP router:

Network No.	Net Mask	Next Hop
128.96.170.0	255.255.254.0	Interface 0
128.96.168.0	255.255.254.0	Interface 1
128.96.166.0	255.255.254.0	R2
128.96.164.0	255.255.252.0	R3
0.0.0.0	Default	R4

For each IP address in Group I identify the correct choice of the next hop from Group II using the entries from the routing table above.

Group I	Group II
i) 128.96.171.92	a) Interface 0
ii) 128.96.167.151	b) Interface 1
iii) 128.96.163.151	c) R2
iv) 128.96.165.121	d) R3
	e) R4

(A) i–a, ii–c, iii–e, iv–d (B) i–a, ii–d, iii–b, iv–e
(C) i–b, ii–c, iii–d, iv–e (D) i–b, ii–c, iii–e, iv–d

10. Host A sends a UDP datagram containing 8880 bytes of user data to host B over an Ethernet LAN. Ethernet frames may carry data upto 1500 bytes (i.e., MTU = 1500 bytes). Size of UDP header is 8 bytes and size of IP header is 20 bytes. There is no option field in IP header. How many total number of IP fragments will be transmitted and what will be the contents of offset field in the last fragment?

(A) 6 and 925 (B) 6 and 7400
(C) 7 and 1110 (D) 7 and 8880

11. In the network 200.10.11.144/27, the fourth octet (in decimal) of the last IP address of the network which can be assigned to a host is _____

12. An IP datagram of size 1000 bytes arrives at a router. The router has to forward this packet on a link whose MTU (maximum transmission unit) is 100 bytes.

Assume that the size of the IP header is 20 bytes.

The number of fragments that the IP datagram will be divided into for transmission is _____.

13. For the IEEE 802.11 MAC protocol for wireless communication, which of the following statements is/ are **TRUE**?
    I. At least three non-overlapping channels are available for transmissions.
    II. The RTS-CTS mechanism is used for collision detection.
    III. Unicast frames are ACKed.
    (A) All I, II and III (B) I and III only
    (C) II and III only (D) II only

14. The maximum number of IPv4 router addresses that can be listed in the record route (RR) option field of an IPv4 header is_____.

15. Match the following:

	Field		Length in bits
P.	UDP Header's Port Number	I.	48
Q.	Ethernet MAC Address	II.	8
R.	IPv6 Next Header	III.	32
S.	TCP Header's Sequence Number	IV.	16

(A) P-III, Q-IV, R-II, S-I
(B) P-II, Q-I, R-IV, S-III
(C) P-IV, Q-I, R-II, S-III
(D) P-IV, Q-I, R-III, S-II

16. Consider an IP packet with a length of 4,500 bytes that includes a 20-byte IPv4 header and a 40-byte TCP header. The packet is forwarded to an IPv4 router that supports a Maximum Transmission Unit (MTU) of 600 bytes. Assume that the length of the IP header in all the outgoing fragments of this packet is 20 bytes. Assume that the fragmentation offset value stored in the first fragment is 0.

The fragmentation offset value stored in the third fragment is _____.

Answer Keys

Exercises

Practice Problems 1

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

Practice Problems 2

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

Selected Questions

1. D	**2.** C	**3.** C	**4.** 26	**5.** D	**6.** 1	**7.** 256	**8.** B	**9.** A	**10.** C
11. 158	**12.** 13	**13.** B	**14.** 9	**15.** C	**16.** 144				

Chapter 5

Network Security

LEARNING OBJECTIVES

- *Network security basics*
- *Terminologies*
- *Cryptographic techniques*
- *Encryptions*
- *Types of keys*
- *Traditional cipher algorithms*
- *Substitution cipher*
- *Traditional cipher*
- *Symmetric key encryption*
- *Asymmetric key encryption*
- *Diffie-hellman*
- *Digital signatures and certificates*

Network Security Basics

It is necessary to defi ne some fundamental terms relating to network security and are the elements used to measure the security of a network. These terms are used to measure the security of a network. To be considered suffi ciently advanced along the spectrum of security, a system must adequately address identifi cation, integrity, accountability, non-repudiation, authentication, availability, confi dentiality each of which is defi ned in the following sections:

Identi ication

Identifi c ation is simply the process of identifying one's self to another entity or determining the identity of the individual or entity, with whom you are communicating.

Authentication

Authentication serves as proof that you are who you say you are or what you claim to be. Authentication is critical if there is to be any trust between parties. Authentication is required when communicating over a network or logging into a network. When communicating over a network you should ask yourself two questions.

1. With whom am I communicating?
2. Why do I believe this person or entity is who he claims to be?

Access Control (Authorization)

This refers to the ability to control the level of access that individuals or entities have to a network or system and how much information they can receive. Level of authorization basically determines what you're allowed to do once you are authenticated and allowed access to a network, system or some other resource such as data or information. Access control is the determination of the level of authorization to a system, network or information.

Availability

This refers to whether the network, system, hardware and software are reliable and can recover quickly and completely in the event of an interruption in service. Ideally, these elements should not be susceptible to denial of service attacks.

Con identiality

This is also be called privacy or secracy to the protection of information from unauthorized disclosure. Usually achieved either by restricting access to the information or by encrypting the information so that it is not meaningful to unauthorized individuals or entities.

Integrity

This can be thought of as accuracy, this refers to the ability to pro-tect information, data, or transmissions from unauthorized, uncon-trolled, or accidental alterations.

Accountability

This refers to the ability to track or audit what an individual or entity is doing on a network or system.

Non-repudiation

The ability to prevent individuals or entities from denying (repudiating) that information, data or fi les were sent or received or that information or fi les were accessed or altered, when infact they were. This capability is crucial in e-commerce, without if an individual or

UNIT 4

Part B Information Systems

Chapter 1

Process Life Cycle

LEARNING OBJECTIVES

- *Introduction*
- *Process vs program*
- *Software component and elements*
- *Information gathering*
- *Requirement analysis*
- *Feasibility study*
- *Data flow diagram*
- *Process specification*
- *Input/output design*
- *Software process life cycle*
- *Software process model*

Introduction

A system can be defi ned as an orderly grouping of interdependent components linked together according to a plan to achieve a specifi c objective.

Example: Telephone system, transportation system, accounting system, etc.

Process Versus Program

A software process gives all steps used to create a software application, from the customer's requirements to the finishedproduct.

- The software process determines the organization and flexibility of the project.
- There are several different software processes and each describes their own solution to develop a valid software.
- Software programs are written programs or rules with associated documentation pertaining to the operation of a computer system.

Software Components and Elements

Software Component

It is a software element that can be independently deployed and composed without modifi c ation according to a composition standard.

- A component model implementation is the dedicated set of executable software elements required to support the execution of components.
- A component has clearly defined interfaces.
- An interface standard is the mandatory requirement enforced to enable software elements to directly interact with other software elements.
- An interface standard declares, when an interface comprises.

Standard An object or measure serving as a basis to which others should conform, by which the quality of others is judged.

Software Element

A sequence of abstract program statements that describe computations, which has to be performed by a machine.

Interface

It describes the behavior of a component that is obtained by con-sidering only the interactions of that interface and by hiding all other interactions.

- An abstraction of the behavior consists of subset of the interactions of one component together with a set of constraints.

Interaction

It is defined as action between 2 or more software elements.

Composition It is a combination of 2 or more software components, the newly formed component, behaviour will be at a diff erent level of abstraction.

The characteristics of new component is determined the components combined and the way in which they are combined.

Chapter 2

Project Management and Maintenance

LEARNING OBJECTIVES

- *Project management*
- *Software design*
- *Modeling component level design*
- *SRS*
- *Software testing*
- *White-box testing*
- *Black box testing*
- *Implementation maintenance*
- *Software quality assurance*
- *Software Re-engineering*
- *COCOMO MODEL*

Project Management

Project management is a technique used to ensure successful completion of a project by the project managers.

The functions included in project management are:

- Estimating resource requirements
- Scheduling tasks and events
- Providing training and site preparation
- Selecting qualified staff and supervising their work
- Monitoring the projects program
- Documenting
- Periodic evaluation
- Contingency planning

Project management involves planning, organization and control projects. It uses tools and software packages for planning and managing projects.

Project planning involves plotting project activities against time frame.

Project Planning Tools

- Tools used during software planning
- Helps the top level managers to take critical decisions during planning stage

Gantt Charts

This activity scheduling method introduced in 1914 by Henry L. Gantt, uses horizontal bars to show the duration of actions or tasks. The left end marks the beginning of the task and the right end its finish. Earlier tasks appear in the upper left and later ones in the lower right.

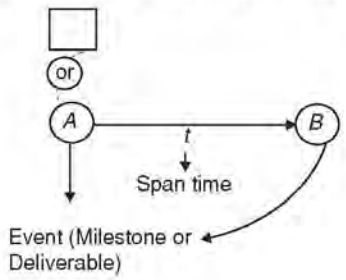
In real-life applications, an allowance for contingencies is provided. This is called **slack time.** Each project allows between 5 to 25 percent slack time for completion.

Program Evaluation and Review Technique (Pert)

Gantt charts do not show precedence relationships among the tasks and milestones of a project.

A PERT chart is a project management tool used to schedule, organize and coordinate tasks within a project.

A PERT chart presents a graphic illustration of a project as a network diagram consisting of numbered nodes (either circles or rectangles) representing events, or milestones in the project linked by labelled vectors (directional lines) representing tasks in the project. The direction of the arrows on the lines indicates the sequence of tasks.



Selected Questions

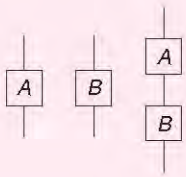
1. The coupling between different modules of a software is categorized as follows:

I. Content coupling
II. Common coupling
III. Control coupling
IV. Stamp coupling
V. Data coupling

Coupling between modules can be ranked in the order of strongest (least desirable) to weakest (most desir-able) as follows:

(A) I-II-III-IV-V (B) V-IV-III-II-I
(C) I-III-V -II-IV (D) IV-II-V -III-I

2. The cyclomatic complexity of each of the modules *A* and *B* shown below is 10. What is the cyclomatic complexity of the sequential integration shown below?



(A) 19 (B) 21
(C) 20 (D) 10

3. A company needs to develop digital signal processing software for one of its newest inventions. The software is expected to have 40000 lines of code. The company needs to determine the effort in person-months needed to develop this software using the basic COCOMO model. The multiplicative factor for this model is given as 2.8 for the software development on embedded systems, while the exponentiation factor is given as 1.20. What is the estimated effort in person months?

(A) 234.25 (B) 932.50
(C) 287.80 (D) 122.40

4. The following is the comment written for a *C* function.

/ * This function computes the roots of a quadratic equation $ax \wedge 2 + bx + c = 0$. The function stores two real roots in * root 1 and * root 2 and returns the status of validity of roots. It handles four different kinds of cases.

(i) When coefficient '*a*' is zero irrespective of discriminant.
(ii) When discriminant is positive.
(iii) When discriminant is zero.
(iv) When discriminant is negative.

Only in case (ii) and (iii), the stored roots are valid. Otherwise 0 is stored in the roots. The function returns 0 when the roots are valid and –1 otherwise.

The function also ensures root1 > = root2

int get_QuadRoots (float *a*, float *b*, float *c*, float * root1, float * root 2); */

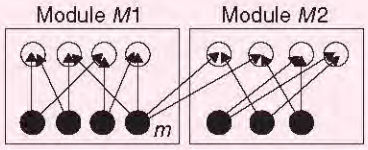
A software test engineer is assigned the job of doing black box testing. He comes up with the following test cases, many of which are redundant.

Test Case	Input Set			Expected Output Set		
	a	*b*	*c*	root1	root2	Return value
*T*1	0.0	0.0	7.0	0.0	0.0	–1
*T*2	0.0	1.0	3.0	0.0	0.0	–1
*T*3	1.0	2.0	1.0	–1.0	–1.0	0
*T*4	4.0	–12.0	9.0	1.5	1.5	0
*T*5	1.0	–2.0	–3.0	3.0	–1.0	0
*T*6	1.0	1.0	4.0	0.0	0.0	–1

Which one of the following options provide the set of non-redundant tests using equivalence class partitioning approach from input perspective for black-box testing?

(A) *T*1, *T*2, *T*3, *T*6 (B) *T*1, *T*3, *T*4, *T*5
(C) *T*2, *T*4, *T*5, *T*6 (D) *T*2, *T*3, *T*4, *T*5

5. The following figure represents access graphs of two modules *M*1 and *M*2. The filled circles represent methods and the unfilled circles represent attributes. If method *m* is moved to module *M*2 keeping the attributes where they are, what can we say about the average cohesion and coupling between modules in the system of two modules?



(A) There is no change
(B) Average cohesion goes up but coupling is reduced
(C) Average cohesion goes down and coupling also reduces.
(D) Average cohesion and coupling increase.

Common data for questions 6 and 7: The procedure given below is required to find and replace certain characters inside an input character string supplied in array *A*. The characters to be replaced are supplied in array 'oldc', while their respective replacement characters are supplied in array 'newc'. Array *A* has fixed length of five characters, while arrays 'oldc' and 'newc' contain three characters each.

UNIT 4

Part C Software Engineering and Web Technology

Chapter 1: Markup Languages 4.111

Theory of Computation

UNIT 5

Chapter 2

Context Free Languages and Push Down Automata

LEARNING OBJECTIVES

- *Context free grammar*
- *Context free language*
- *Ambiguity in context free grammars*
- *Removing ?-productions*
- *Removing unit productions*
- *Normal forms*
- *Chomsky's normal form*
- *Greiback normal form*
- *Closure properties of CFL's*
- *Push down automata*
- *PDAs accepting by final state and empty stack are equivalent*
- *Converting CFG to PDA*
- *Deterministic PDA*

Context Free Grammar

- A context free grammar (CFG) is a finite set of variables (non-terminals) each of which represents a language. The language represented by variables is described recursively in terms of each other. The primitive symbols are called terminals.
- The rules relating variables are called productions. A typical production states that the language associated with a given variable contains strings that are formed by concatenating strings from languages of certain other variables.
- CFG is a collection of three things;
  An alphabet Z of letters called terminals.
  A set of symbols called non-terminals, one of which is a start symbol, S.
  A finite set of productions of the form:
  One terminal $\rightarrow$ finite set of terminals and/or non-terminals.
- A CFG is defined as: $G = (V, T, P, S)$

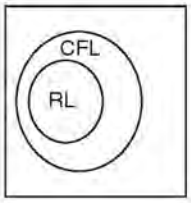
  Where

  $V \rightarrow$ Finite set of variables (non-terminals)
  $T \rightarrow$ Finite set of terminals (symbols)
  $P \rightarrow$ Finite set of productions, each, production is of the form, $A \rightarrow \alpha, A \in V, \alpha \in (V \cup T)^*$
  $S \rightarrow$ Start symbol

Context Free Language (CFL)

The language generated by CFG is a set of all strings of terminals that can be produced from start symbols, using the productions as substitutions. A language generated by a CFG is called context free language (CFL).

Note: Every regular grammar is context free, so a regular language (RL) is also context free.
Family of RL's is proper subset of CFL's.
i.e., RL $\subset$ CFL



Solved Examples

Example 1: What is the language that is generated by CFG, $G = S \rightarrow AB|A \rightarrow +/-|B \rightarrow CB/C|C \rightarrow 0/1/2/ \ldots 9$.
(A) Set of all rational numbers
(B) Set of all integers
(C) Set of all natural numbers
(D) Set of all complex numbers

Solution: (B)
$S \rightarrow AB|A \rightarrow +/-|B \rightarrow CB/C|C \rightarrow 0/1/2/ \ldots 9$
Consider-18 (integer)

$B \rightarrow SU_3|V_2V_2|V_1U_2|a|V_1S$
$U_1 \rightarrow SB$
$U_2 \rightarrow AS$
$U_3 \rightarrow V_2S$
$V_1 \rightarrow a$
$V_2 \rightarrow b$
$\therefore$ Nine non-terminals.

Greiback Normal Form (GNF)

- A CFG, $G = (V, T, R, S)$ is said to be in GNF, if every production is of form $A \rightarrow a\alpha$ where $a \in T$, $\alpha \in V^*$, i.e., α is a string of zero or more variables.
- Left recursion in R can be eliminated by following schema: If $A \rightarrow A\alpha_1 | A\alpha_2 | \ldots | A\alpha_r | \beta_1 | \beta_2 | \ldots | \beta_s$, then replace the above rules by
  (i) $A \rightarrow \beta_i | \beta_i Z, 1 \le i \le s$
  (ii) $Z \rightarrow \alpha_i | \alpha_i Z, 1 \le i \le r$
- If $G = (V, T, R, S)$ is a CFG, then another CFG, $G_1 = (V_1, T, R_1, S)$ can be constructed in GNF $\exists L(G_1) = L(G) - \{\varepsilon\}$.

The step wise algorithm is as follows:

1. Eliminate null production, unit productions and useless symbols from the grammar G and then construct a $G^1 = (V^1, T, R^1, S)$ in CNF generating the language $L(G^1) = L(G) - \{\varepsilon\}$.
2. Rename the variables like $A_1, A_2, \ldots A_n$ starting with $S = A_1$.
3. Modify the rules in R^1, so that if $A_i \rightarrow A_j\gamma \in R^1$ then $j > i$.
4. Starting with A_1 and proceeding to A_n, can be obtained as:
   (a) Assume that productions have been modified so that for $1 \le i \le k$, $A_i \rightarrow A_j\gamma \in R^1$ only if $j > i$
   (b) If $A_k \rightarrow A_j\gamma$ is a production with $j < k$, generate a new set of productions substituting for A_j, the body of each A_j production.
   (c) Repeating (b) atmost $k-1$ times, obtains rules of the form $A_k \rightarrow A_p\gamma, p \ge k$.
   (d) Replace rules $A_k \rightarrow A_k\gamma$ by removing left-recursion.
5. Modify the $A_i \rightarrow A_j\gamma$ for $i = n-1, n-2, \ldots 1$ in desired form at same time change z production rules.

Example 7: A grammar G is defined with rules $S \rightarrow XA|BB$, $B \rightarrow b|SB$, $X \rightarrow b$, $A \rightarrow a$. The normalized GNF of G contains ____ productions.
(A) 17 (B) 19
(C) 5 (D) 16

Solution: (B)
1. The Grammar, G is already in CNF.
2. Re-label with variables
   S with A_1
   X with A_2
   A with A_3
   B with A_4
   Grammar, G now is:
   $A_1 \rightarrow A_2A_3|A_4A_4$
   $A_4 \rightarrow b|A_1A_4$
   $A_2 \rightarrow b$
   $A_3 \rightarrow a$
3. Identify all productions which do not conform to any of the types listed below:
   $A_i \rightarrow A_j x_k \exists j > i$
   $Z_i \rightarrow A_j x_k \exists j \le n$
   $A_i \rightarrow a x_k \exists x_k \in V^*$ and $a \in T$
4. $A_4 \rightarrow A_1 A_4 \ldots$ identified
5. $A_4 \rightarrow A_1 A_4|b$
   To eliminate A_1, use substitution rule, $A_1 \rightarrow A_2 A_3|A_4 A_4$
   $\therefore \quad A_4 \rightarrow A_2 A_3 A_4|A_4 A_4 A_4|b$
   Substitute $A_2 \rightarrow b$
   $\therefore \quad A_4 \rightarrow b A_3 A_4|A_4 A_4 A_4|b$
   $A_4 \rightarrow A_4 A_4 A_4$ is left recursive. So, remove left recursion i.e., $A_4 \rightarrow b A_3 A_4|b|bA_3A_4Z|bZ$
   $Z \rightarrow A_4A_4|A_4 A_4Z$
6. Now, $G = A_1 \rightarrow A_2 A_3|A_4 A_4$
   $A_4 \rightarrow b A_3A_4|b|b A_3 A_4 Z|b Z$
   $Z \rightarrow A_4 A_4|A_4A_4Z$
   $A_2 \rightarrow b$
   $A_3 \rightarrow a$
7. A_1, Z are not in GNF. So,
   For $A_1 \rightarrow A_2A_3|A_4A_4$:
   Substitute for A_2 and A_4 to convert it to GNF
   $A_1 \rightarrow b A_3| b A_3 A_4A_4|b A_4|b A_3 A_4Z A_4|b Z A_4$
   For $Z \rightarrow A_4 A_4|A_4 A_4Z$
   substitute for A_4 to convert it to GNF
   $Z \rightarrow b A_3 A_4A_4|b A_4|b A_3 A_4Z A_4|b Z A_4|b A_3 A_4A_4$
   $Z|b A_4Z|b A_3A_4 Z A_4Z|b Z A_4 Z$
   $\therefore$ Final GNF is:
   $A_1 \rightarrow b A_3|b A_3A_4A_4|b A_4| b A_3A_4Z A_4|b Z A_4$
   $A_4 \rightarrow b A_3 A_4|b|b A_3 A_4Z|b Z$
   $A_2 \rightarrow b$
   $A_3 \rightarrow a$
   $Z \rightarrow b A_3 A_4A_4|b A_4|b A_3A_4 Z A_4|b Z A_4|b A_3 A_4 A_4$
   $Z| b A_4 Z|b A_3 A_4 Z A_4Z|b Z A_4 Z$
   $\therefore$ 19 productions.

Pumping Lemma for Context Free Languages

Let 'L' be context free language. There exists some integer, m $\exists \forall w$ in L, with $|w| \ge m$, $w = uvxyz$ with $|vxy| \le m$ and $|vy| \ge 1$ $\exists u\, v^i\, x\, y^i\, z \in L \ \forall i = 0, 1, 2, 3, \ldots$

Note: Pumping lemma is used to show that a language is Not context free.

Example 8: The language $\{a^n b^m c^n d^{(n+m)}: m, n \ge 0\}$ is
(A) Regular
(B) Context free but not regular
(C) Neither context free nor regular
(D) Cannot be determined

(A) $L = \{a^n b^m : m = n\}$
(B) $L = \{a^n b^m : m = n + 2\}$
(C) $L = \{a^n b^m : m \geq n + 2\}$
(D) $L = \{a^n b^m : m \leq n + 2\}$

11. The CFG, G is defined with rules:
$S \to AB|CD, A \to A00|\varepsilon, B \to B11|1, C \to C00|0, D \to D11|\varepsilon$. The language generated by G is
(A) $L = \{0^n 1^n | n \geq 0\}$
(B) $L = \{0\ 0^n\ 1\ 1^n | n > 0\}$
(C) $L = \{0^n 1^m | n + m \text{ is odd}\}$
(D) $L = \{0^n 1^m | n + m \text{ is even}\}$

12. The languages, L_1, L_2, L_3 are defined as:
$L_1 = \{a^n b^m c^{n+m} | n, m \geq 0\}$, $L_2 = \{a^n b^n c^m | n, m \geq 0\}$, $L_3 = \{a^n b^n c^{2n} | n \geq 0\}$, Which of the following statements are true?
(i) L_1, L_2 are context free
(ii) L_1, L_3 are context free
(iii) $L_3 = L_1 \cap L_2$
(iv) L_1, L_3 are context free but not L_2
(A) (i), (ii) (B) (i), (iii)
(C) (ii), (iii) (D) (iii), (iv)

13. The language, L_1 and L_2 are defined as $L_1 = \{a^n b^n : n \geq 0$ and n is not a multiple of 5$\}$ and $L_2 = \{0^n \# 0^{2n} \# 0^{3n} | n \geq 0\}$. Which of following is true?
(A) L_1 and L_2 are context free
(B) Only L_1 is context free
(C) Only L_2 is context free
(D) Neither L_1 nor L_2 is context free

14. The language, L_1 and L_2 are defined as $\overline{L_1} = \{0^n 1^n)^m | m, n > 0\}$, $L_2 = \{0^n\ 1^n\ 0^n\ 1^n | n \geq 0\}$ which of following is true?
(A) L_1 and L_2 are context free
(B) Only $\overline{L_1}$ is context free
(C) Only L_2 is context free
(D) Neither L_1 nor L_2 is context free

15. The language L_1, L_2 are defined as $L_1 = \{0^i\ 1^i\ 0^j\ 1^i | i, j > 0\}$, $L_2 = \{1^k\ 0^i\ 1^i\ 0^j\ 1^j\ 0^k | i, j, k > 0\}$. Which of following is true?
(A) L_1 and L_2 are context free
(B) Only L_1 is context free
(C) Only L_2 is context free
(D) Neither L_1 nor L_2 is context free

Selected Questions

1. Match the following:

E. Checking that identifiers are declared before their use	P. $L = \{a^n b^m c^n d^m \| n \geq 1, m \geq 1\}$
F. Number of formal parameters in the declaration of a function agrees with the number of actual parameters in use of that function	Q. $X \to XbX \| XcX \| dX \| g$
G. Arithmetic expressions with matched pairs of parentheses	R. $L = \{wcw \| w \in (a\|b)^*\}$
H. Palindromes	S. $X \to bXb \| cXc \| \varepsilon$

(A) E − P, F − R, G − Q, H − S
(B) E − R, F − P, G − S, H − Q
(C) E − R, F − P, G − Q, H − S
(D) E − P, F − R, G − S, H − Q

2. Consider the languages L_1, L_2 and L_3 as given below.
$L_1 = \{0^p\ 1^q | p, q \in N\}$,
$L_2 = \{0^p\ 1^q | p, q \in N$ and $p = q\}$ and
$L_3 = \{0^p\ 1^q\ 0^r | p, q, r \in N$ and $p = q = r\}$. Which of the following statements is NOT TRUE?
(A) Push Down Automata (PDA) can be used to recognize L_1 and L_2.
(B) L_1 is a regular language.
(C) All the three languages are context free
(D) Turing machines can be used to recognize all the languages.

3. Which of the following problems are decidable?
(1) Does a given program ever produce an output?
(2) If L is a context free language, then, is $\overline{L}$ also context free?
(3) If L is a regular language, then, is $\overline{L}$ also regular?
(4) If L is recursive language, then, is $\overline{L}$ also recursive?
(A) 1, 2, 3, 4 (B) 1, 2
(C) 2, 3, 4 (D) 3, 4

4. Consider the following languages.
$L_1 = \{0^p\ 1^q\ 0^r | p, q, r \geq 0\}$
$L_2 = \{0^p\ 1^q\ 0^r | p, q, r \geq 0, p \neq r\}$
Which one of the following statements is **FALSE**?
(A) L_2 is context-free
(B) $L_1 \cap L_2$ is context-free
(C) Complement of L_2 is recursive
(D) Complement of L_1 is context-free but not regular

5. Which one of the following is **TRUE**?
(A) The language $L = \{a^n b^n | n \geq 0\}$ is regular
(B) The language $L = \{a^n | n$ is prime$\}$ is regular
(C) The language $L = \{w | w$ has $3k + 1b$'s for some k $\in N$ with $\Sigma = \{a, b\}\}$ is regular
(D) The language $L = \{ww | w \in \Sigma^*$ with $\Sigma = \{0, 1\}\}$ is regular.

6. Consider the following languages over the alphabet $\Sigma = \{0, 1, c\}$.
$L_1 = \{0^n 1^n | n \geq 0\}$
$L_2 = \{wcw^r | w \in \{0, 1\}^*\}$
$L_3 = \{ww^r | w \in \{0, 1\}^*\}$

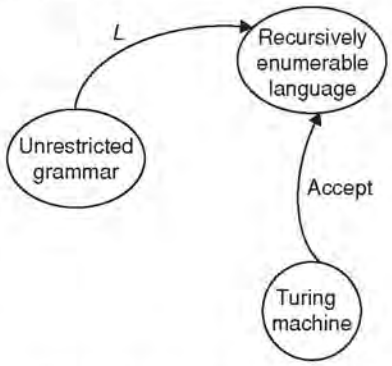
Chapter 3

Recursively Enumerable Sets and Turing Machines, Decidability

LEARNING OBJECTIVES

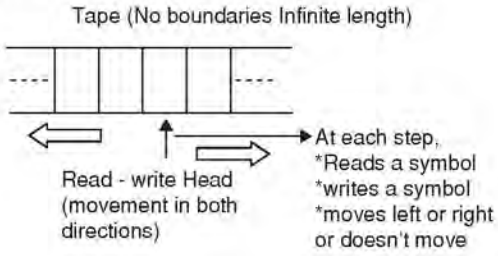
- *Turing machines*
- *Model of turing machine*
- *Types of turing machines*
- *Offline turing machine*
- *Universal turing machine*
- *Recursively enumerable languages*
- *Recursive language*
- *Undecidability*
- *Church's hypothesis*
- *Halting problem*
- *Post's correspondence problem*
- *7 P problems*
- *NP problems*
- *NP – complete problem*
- *NP – hard problem*
- *Closure properties of formal languages*

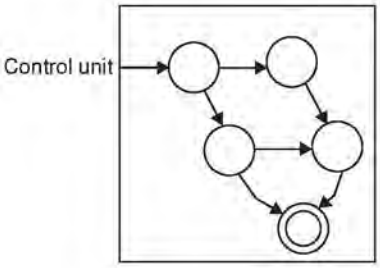
Turing Machines



A Turing machine is a kind of state machine, which is much more powerful in terms of languages it can recognize. At any time, the machine is in any one of the finite number of states. Instructions for a turing machine include the specification of conditions, under which the machine will make transitions from one state to other.

Model of Turing Machine





Compiler Design

UNIT 6

Chapter 1

Lexical Analysis and Parsing

LEARNING OBJECTIVES

- *Language processing system*
- *Lexical analysis*
- *Syntax analysis*
- *Context free grammars and ambiguity*
- *Types of parsing*
- *Top down parsing*
- *Bottom up parsing*
- *Conflicts*
- *Operator precedence grammar*
- *LR parser*
- *Canonical LR parser(CLR)*

Language Processing System

Language Processors

Interpreter

It is a computer program that executes instructions written in a programming language. It either executes the source code directly or translates source code into some effi cient intermediate representation and immediately executes this.

Example: Early versions of Lisp programming language, BASIC.

Translator

A software system that converts the source code from one form of the language to another form of language is called as translator. There are 2 types of translators namely (1) Compiler (2) Assembler.

Compiler converts source code of high level language into low level language.

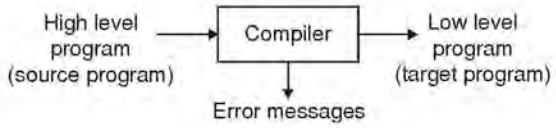
Assembler converts assembly language code into binary code.

Compilers

A compiler is a software that translates code written in high-level language (i.e., source language) into target language.

Example: source languages like C, Java, … etc. Compilers are user friendly.

The target language is like machine language, which is effic ient for hardware.



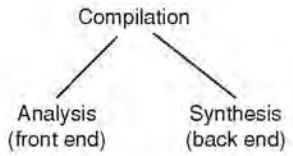
Passes

The number of iterations to scan the source code, till to get the executable code is called as a pass.

Compiler is two pass. Single pass requires more memory and multipass require less memory.

Analysis–synthesis model of compilation

There are two parts of compilation:



Analysis It breaks up the source program into pieces and creates an intermediate representation of the source program. This is more language specifi c.

Synthesis It constructs the desired target program from the inter-mediate representation. The target program will be more machine specifi c, dealing with registers and memory locations.

Chapter 2

Syntax Directed Translation

LEARNING OBJECTIVES

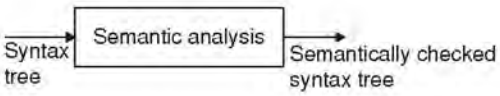
- *Syntax directed translation*
- *Syntax directed definition*
- *Dependency graph*
- *Constructing syntax trees for expressions*
- *Types of SDD's*
- *S-attributed definition*
- *L-attributed definitions*
- *Synthesized attributes on the parser*
- *Syntax directed translation schemes*
- *Bottom up evaluation of inherited attributes*

Syntax Directed Translation

To translate a programming language construct, a compiler may need to know the type of construct, the location of the first instruction, and the number of instructions generated…etc. So, we have to use the term 'attributes' associated with constructs.

An attribute may represent type, number of arguments, memory location, compatibility of variables used in a statement which cannot be represented by CFG alone.

So, we need to have one more phase to do this, i.e., 'semantic analysis' phase.



In this phase, for each production CFG, we will give some semantic rule.

Syntax directed translation scheme

A CFG in which a program fragment called output action (semantic action or semantic rule) is associated with each production is known as Syntax Directed Translation Scheme.

These semantic rules are used to

1. Generate intermediate code.
2. Put information into symbol table.
3. Perform type checking.
4. Issues error messages.

Notes:

1. Grammar symbols are associated with attributes.
2. Values of the attributes are evaluated by the semantic rules associated with production rules.

Notations for Associating Semantic Rules

There are two techniques to associate semantic rules:

Syntax directed definition (SDD) It is high level specification for translation. They hide the implementation details, i.e., the order in which translation takes place.

Attributes + CFG + Semantic rules = Syntax directed definition (SDD).

Translation schemes These schemes indicate the order in which semantic rules are to be evaluated. This is an input and output mapping.

Syntax Directed Definitions

A SDD is a generalization of a CFG in which each grammar symbol is associated with a set of attributes.

There are two types of set of attributes for a grammar symbol.

1. Synthesized attributes
2. Inherited attributes

Each production rule is associated with a set of semantic rules.

Chapter 3

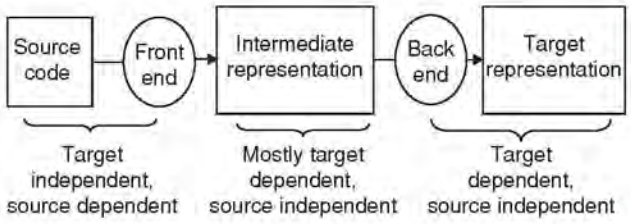
Intermediate Code Generation

LEARNING OBJECTIVES

- *Introduction*
- *Directed Acyclic Graphs (DAG)*
- *Three address code*
- *Symbol table operations*
- *Assignment statements*
- *Boolean expression*
- *Flow control of statements*
- *Procedure calls*
- *Code generation*
- *Next use information*
- *Run-time storage management*
- *DAG representations of basic blocks*
- *Peephole optimization*

Introduction

In the analysis–synthesis model, the front end translates a source program into an intermediate representation (IR). From IR the back end generates target code.



There are different types of intermediate representations:

- High level IR, i.e., AST (Abstract Syntax Tree)
- Medium level IR, i.e., Three address code
- Low level IR, i.e., DAG (Directed Acyclic Graph)
- Postfix Notation (Reverse Polish Notation, RPN).

In the previous sections already we have discussed about AST and RPN.

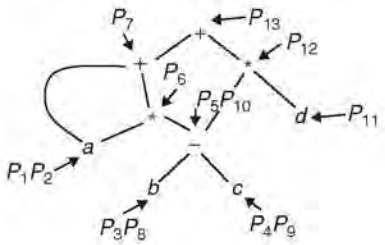
Benefits of Intermediate code generation: The benefits of ICG are

1. We can obtain an optimized code.
2. Compilers can be created for the different machines by attaching different backend to existing front end of each machine.
3. Compilers can be created for the different source languages.

Directed acyclic graphs for expression: (DAG)

- A DAG for an expression identifies the common sub expressions in the given expression.
- A node N in a DAG has more than one parent if N represents a common sub expression.
- DAG gives the compiler, important clues regarding the generation of efficient code to evaluate the expressions.

Example 1: DAG for $a + a*(b - c) + (b - c)*d$



P_1 = makeleaf (id, a)
P_2 = makeleaf (id, a) = P_1
P_3 = makeleaf (id, b)
P_4 = makeleaf (id, c)
P_5 = makenode (−, P_3, P_4)
P_6 = makenode (*, P_1, P_5)
P_7 = makenode (+, P_1, P_6)
P_8 = makeleaf (id, b) = P_3
P_9 = makeleaf (id, c) = P_4
P_{10} = makenode (−, P_8, P_9) = P_5

Operating System

UNIT 7

Chapter 1

Processes and Threads

LEARNING OBJECTIVES

- *Basics of operating systems*
- *Services of OS*
- *Evolution of operating systems*
- *Processes*
- *Processes and process control blocks*
- *Process states*
- *Process creation*
- *Suspended processes*
- *OS control structures*
- *Process attributes*
- *Modes of execution*
- *Threads*
- *Thread functionality*
- *Thread synchronization*

Basics of Operating System

An operating system (OS) is a program that controls the execution of application programs and acts as an interface between applications and the computer hardware. The three objectives of an OS are as follows:

1. *Convenience*: An OS makes a computer more convenient to use.
2. *Efficiency*: An OS allows the computer system resources to be used in an efficient manner.
3. *Ability to evolve*: An OS should be constructed in such a way as to permit the effective development, testing and introduction of new system functions without interfering with service.

OS as a User–Computer Interface

Consider the below fi gure (Figure 1) which shows the hardware and software used in providing applications to a user in a layered fashion.

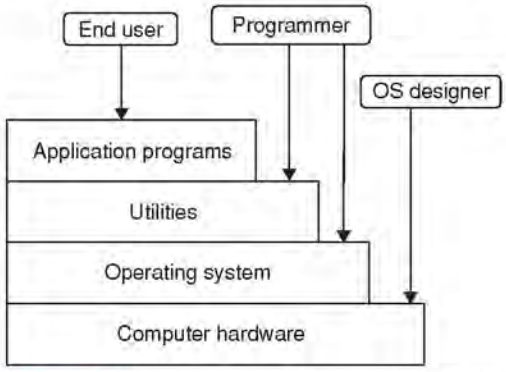


Figure 1 Layers and views of a computer system.

The end user of the application is not concerned with the details of computer hardware. Utilities implement the frequently used functions that assist in program creation, the management of fi les and control of input/output (I/O) devices. The most important collection of system programs comprises the OS. The OS masks the details of the hardware from the programmer and provides the programmer with a convenient interface for using the system.

Services of an OS

1. *Program development*: An OS provides a variety of facilities and services as editors and debuggers to assist programmers in creating programs.
2. *Program execution*: An OS handles the scheduling duties of program execution for the user.
3. *Access to I/O devices*: An OS provides a uniform interface that hides the details of I/O devices so that the programmers can access the I/O devices using simple reads and writes.
4. *Controlled access to files*: In a system with multiple users, an OS provides protection mechanism to control access to the files.
5. *System access*: For shared or public systems, an OS controls access to the system as a whole and to specific system resources.
6. *Error detection and response*: An OS must provide a response that clears the error condition with the least impact on running applications.
7. *Accounting*: A good OS will collect usage statistics for various resources and monitor performance parameters (viz., response time).

Chapter 2

Interprocess Communication, Concurrency and Synchronization

LEARNING OBJECTIVES

- *Principles of concurrency*
- *Process interaction*
- *Mutual exclusion*
- *Semaphores*
- *Binary semaphore*
- *Mutual exclusion using semaphores*
- *Progress using semaphores*
- *Classical problems of synchronization*
- *Dining philosophers problem*
- *Monitors*
- *Message passing*
- *Indirect addressing*
- *Mutual exclusion using message passing*

Basic Concepts

Multiprogramming: It deals with the management of multiple processes within a uniprocessor system.

Multiprocessing: It deals with the management of multiple processes within a multiprocessor.

The fundamental operating system (OS) design is *concurrency*. Concurrency encompasses a host of design issues, including communication among processes, sharing of and competing for resources, synchronization of the activities of multiple processes and allocation of processor time to processes.

Principles of Concurrency

There are two examples for concurrent processing as follows:

1. In a single-processor multiprogramming system, processes are interleaved in time to yield the appearance of simultaneous execution.
2. In a multiprocessor system, it is possible not only to interleave the execution of multiple processes but also to overlap them.

There are two problems with these techniques:

1. Problem with sharing of global resources
2. Problem with allocation of resources optimally.
3. Problem with locating a programming error as results is not deterministic and reproducible.

Example:

```
void process( )
{
in = getchar( );
out = in;
putchar(out);
}
```

The procedure 'process' reads a character and prints it. Let us suppose that we have a uniprocessor system, with single user. Let the user running multiple applications and all applications use the procedure for reading and printing, that is, all the applications share common procedure for efficient and close interaction among them. But this sharing leads to problems. For example,

1. Let the process P_1 invokes 'process' and is interrupted immediately after 'getchar' returns its value and stores it in 'in'. Here the most recently entered character 'C' is stored in variable 'in'.
2. Now, suppose the process P_2 is activated and it invokes 'process', which runs to conclusion, inputting and then displaying a single character, D, on the screen.
3. The process P_1 is resumed. By this time, the value 'C' has been overwritten in 'in' and therefore lost. Instead 'in' contains 'D', which is transferred to 'out' and displayed.

Here the problem is with sharing a global variable. To avoid these types of problems, we impose some rules like, only one process

The initial value of n and S are $n = 0$, $S = 1$. (n is the number of items in the buffer).

Table 3 *Producer consumer bounded buffer problem*

Producer	Consumer
repeat	repeat
produce item v;	while(in == out)
while((in + 1) % n = = out)	no operation;
no operation;	$w = b$[out];
b[in] = v;	out = (out + 1) % n;
in = (in + 1) % n;	consume w;
forever;	forever;

The buffer size is enforced using another counting semaphore.

Table 4 *Producer consumer bounded buffer problem solution*

Producer	Consumers
Repeat	repeat
Produce item v;	Semwait(e);
Semwait(e);	Semwait(S);
Semwait(S);	$w = b$[out];
b[in] = v;	out = (out + 1) % n;
in = (in + 1) % n	Semsignal(S);
Semsignal(S);	Semsignal(e);
Semsignal(e);	consume w;
forever;	forever;

The initial value of buffer size, e is the size of the bounded buffer.

Observations on semaphores:

1. Semaphores are easy to use.
2. wait() and signal() are to be implemented as atomic operations.

Problems:

1. signal() and wait() may be exchanged by the programmer, this may result in deadlock or violation of mutual exclusion.

Readers/Writers Problem

1. A reader reads data.
2. A writer writes data.
3. Data is shared among a number of processes.
4. Multiple readers may read the data simultaneously, that is, concurrently.
5. Only one writer can write the data any time, that is, no reader should be present.
6. A reader and writer cannot access data simultaneously.
7. Locking table: Whether any two can be in the critical section simultaneously is shown in the table.

	Reader	Writer
Reader	OK	NO
Writer	NO	NO

Solution: Readers have priority; if a reader is in CS, any number of readers could enter irrespective of any writer waiting to enter critical section

Writer	Reader
`while(true)`	`while(true)`
`{`	`{`
`Semwait(S);`	`Semwait(x);`
`writeunit();`	`Num = Num + 1;`
`Semsignal(S);`	`if (Num = = 1)`
`}`	`Semwait(S);`
	`Semsignal(x);`
	`Readunit();`
	`Semwait(x);`
	`Num = Num - 1;`
	`if(Num == 0)`
	`Semsignal(S);`
	`Semsignal(x);`
	`}`

Semaphore 'S' is used to enforce mutual exclusion.

Semaphore 'x' is used to assure that 'Num' is updated properly.

Solution: If a writer wants critical section as soon as the critical section is available, writer enters it.

Dining Philosophers Problem

N philosophers are sitting around a dining table. There are N plates placed on the table such that each plate is in front of a philosopher and N forks placed between the plates. There is a bowl of Noodles placed at the centre of the table. Whenever a philosopher feels hungry, he tries to pick two forks which are shared with his nearest neighbour. If any of his neighbours happens to be eating at the time, the philosopher has to wait. Whenever a hungry philosopher gets two forks, he pours noodles into his plate. After he finishes, he places the chopsticks back onto the table and starts thinking. Now forks are available for neighbours.

Solution:

```
# define N 5 /* Number of philosophers*/
void philosopher(int i) /* philosopher number,
from 0 to 4*/
{
while (true)
{
think( ); /* philosopher is thinking*/
take_fork(i) ; /*take left fork*/
take_fork ((i+ 1)% N); /* take right fork; %
is modulo operator*/
eat( );
put _ fork ( ); /* put left back on the table*/
put _ fork ((i + 1) % N); /* put right fork
back on the table */
```

Notes:

1. This solution leads to deadlock.
2. Everyone picks the left fork and indefinitely wait for right fork causing starvation.

Chapter 3

Deadlock and CPU Scheduling

LEARNING OBJECTIVES

- *Deadlock*
- *System model*
- *Bridge crossing*
- *Resources*
- *Resource allocation graph*
- *Methods of handling deadlocks*
- *Deadlock prevention*
- *Resource allocation denial (OR) banker's algorithm*
- *Deadlock detection*
- *Dining philosophers problem*
- *Scheduling algorithms*
- *Scheduling policies*
- *Round Robin scheduling*
- *Shortest remaining time*
- *Highest response ratio next*
- *Multilevel feedback queue scheduling*

DEADLOCK

It is a situation where a process or set of processes is blocked, waiting for some resource that is held by other waiting processes.

System Model

Let the resource types be R_1, R_2 ... R_m (like CPU cycles, memory space, input/output (I/O) devices, etc.). Each resource type R_i has W_i instances; each process utilizes a resource as follows:

Request A process, needing a resource, will request the operating system (OS) for assignment of the needed resource. Then the process waits, till operating system assigns it an instance of the requested resource.

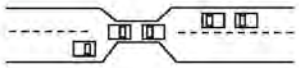
Assignment The OS will assign to the requesting process an instance of the requested resource, whenever, it is available. Then, the process comes out of its waiting state.

Use The process will use the assigned resource. In case, the resource is non-sharable, the process will have exclusive access to it.

Release After the process fi nished with the use assigned resource, it will return the resource to the system pool. The released resource can now be assigned to another waiting process.

Example:

Bridge crossing



If a deadlock occurs, it can be resolved if one car backs up (preempt resources and rollback). Several cars may have to be backed up if a deadlock occurs.

Problem of starvation (infinite wait) is possible.

Resources

Types of resources:

1. Reusable resources
2. Consumable resources

Reusable resources These resources can be safely used by only one process at a time, and are not depleted by that use.

Examples: Processors, I/O channels, main and secondary memory, devices and files, etc.

Consider two processes P and Q that compete for exclusive access to a disk file D and tape drive T. Let their implementation is as shown below:

Table 1 *Process P*

Step	Action
P_0	Request (*D*)
P_1	Lock (*D*)
P_2	Request (*T*)
P_3	Lock (*T*)
P_4	Perform function
P_5	Unlock (*D*)
P_6	Unlock (*T*)

Table 2 *Process Q*

Step	Action
Q_0	Request (*T*)
Q_1	Lock (*T*)
Q_2	Request (*D*)
Q_3	Lock (*D*)
Q_4	Perform Function
Q_5	Unlock (*T*)
Q_6	Unlock (*D*)

solution leads to deadlock, if all of the philosophers are hungry at the same time, they all sit down, they all pick up the fork on their left and they all reach out for the other fork, which is not there.

ExampleA, B two resources. Two processes (P1 and P2) Share these resources. When a Process request for a resources, if that resource is free then it will be allocated with that resources. If the resources are not free then the process will halt. Now the scenario is

a. P1 request A

b. P2 request B

c. P1 request B

d. P2 request A

1. Now A is allocated with which process?
2. B is allocated which process?
3. What is the name of the situation that happens in this scenario?

solution:
1. P1
2. P2
3. Deadlock

CPU Scheduling

1. The objective of multi programmed OS is to maximize CPU utilization by having some process running at all times.
2. The objective of time shared OS is to switch the CPU among processes so frequently that the users can interact with each program while it is executing.
3. When there are more than one process ready to execute with the processor, a selection decision needs to be made to pick a process for execution from among the ready processes. This activity is called *process scheduling*.

Scheduling queue: It maintains information of all ready processes for CPU devices. It is maintained as a linked list.

Types of Scheduling Queue

1. ***Job queue:*** It consists of all processes in the system.
2. ***Ready queue:*** It consists of all processes that are residing in the main memory and are ready but waiting to execute on CPU.
3. ***Device queue:*** It consists of processes waiting for a particular I/O device. Each device has its own queue.

Process CPU–I/O Burst Cycle

The execution of process consists of CPU burst and I/O burst. The execution of process starts with CPU burst and I/O burst, which are executed alternatively.

The alternating sequence of CPU and I/O burst are shown below:

Read *a*
Inc *a*
Read *x* } CPU Burst

I/O waiting } I/O Burst

Dec *x*
Store *x* } CPU Burst

I/O waiting } I/O Burst

.
.
.
.

There should be proper balance between CPU bound process and I/O bound process in a schedule.

Scheduler

A process migrates between various scheduling queues throughout its lifetime. The process of selecting processes from the queues is carried out by scheduler.

Types of Processor Scheduling

There are three types of processor scheduling:

1. Long-term scheduling
2. Medium-term scheduling
3. Short-term scheduling

The following figure relates the scheduling functions to the process state transition diagram:

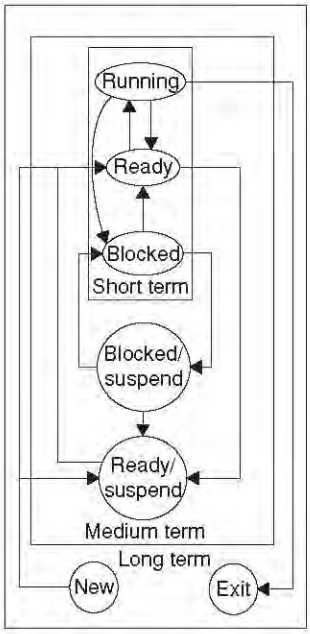


Figure 3 Levels of Scheduling

Chapter 4

Memory Management and Virtual Memory

LEARNING OBJECTIVES

- *Basic concepts*
- *Memory management requirements*
- *Relocation and memory mapping techniques*
- *Placement algorithm*
- *Dynamic partitioning*
- *Placement algorithm*
- *Buddy system*
- *Non-contiguous storage allocation methods*
- *Paging*
- *Segmentation*
- *Page table structure*
- *Hierarchical page table*
- *Inverted page table*
- *Address translation in a segmentation system*

Basic Concepts

Uniprogramming system Main memory is divided into two parts as follows:

1. Operating system (OS) part
2. Program part (which is currently being executed)

Multiprogramming system Here the user part of memory must be further subdivided to accommodate multiple processes.

The task of subdivision is carried out dynamically by the OS and is known as memory management.

Memory Hierarchy

The triangle in Figure 1 gives the hierarchy of memory. The memory hierarchy shows the performance issues.

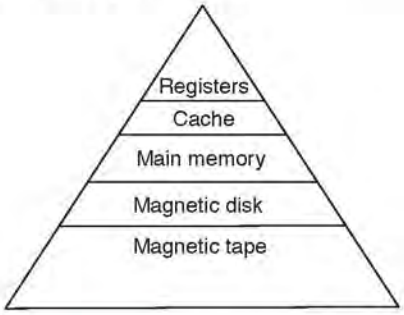


Figure 1 Memory hierarchy.

The memory hierarchy has different types of storage system in computers which are arranged in hierarchy, with respect to speed and cost.

If one moves down the hierarchy, access time increases, the cost per bit decreases, the memory capacity increases and memory access frequency by the processor decreases.

The registers, cache and main memory are volatile, whereas magnetic disc and magnetic tapes are non-volatile storage devices.

Memory Management Requirements

Memory management requirements are as follows:

1. Relocation
2. Protection
3. Sharing
4. Logical organization
5. Physical organization

Relocation

1. The role of relocation, the ability to execute processes independently from their physical location in memory, is central for memory management.
2. In a general purpose multiprogramming environment, a program cannot know in advance what processes will be running in memory when it is executed, nor how much memory the system has available for it, nor where it is located.
3. Hence program relocation is required such that a program must be compiled and linked in such a way that it can later be loaded starting from an unpredictable address in memory, an address that can even change during the execution of the process itself, if any swapping occurs.

Chapter 5

File Systems, I/O Systems, Protection and Security

LEARNING OBJECTIVES

- *File systems*
- *File management systems*
- *File system architecture*
- *Device drivers*
- *Basic input/output supervisor*
- *Logical input/output*
- *Access methods*

File Systems

The filesystemconsistsoftw o distinct parts:

1. Collection of files
2. Directory structure

File A file is a named collection of related information that is recorded on secondary storage. The files must have

1. Long-term existence
2. Sharable between processes
3. Structure

File attributes A typical file attributes are

1. Name
2. Identifier
3. Type
4. Location
5. Size
6. Protection
7. Time, date and user identification

File operations The operations that are applied on files are

1. Creation
2. Deletion
3. Closing
4. Reading
5. Writing

File types A common technique for implementing fi le types is to include the type as part of the fi le name. The name is split into two parts

1. A name
2. An extension (usually separated by a period character)

The type of a file may be

1. Executable (exe, com, bin)
2. Object (obj, o)
3. Source code (c, cc, java)
4. Batch (bat, sh)
5. Text (txt, doc)
6. Word processor (wp, text, doc)
7. Library (lib)
8. Print or view (ps, pdf, jpg)
9. Archive (zip, tar)
10. Multimedia (mpeg, mov, rm)

File structure The four common terms of file systems are

1. Field
   - Basic element of data
   - Contains a single value
   - Has a particular length and data type
2. Record
   - It is a collection of related fields
   - It is treated as a unit
3. File
   - It is a collection of similar records
   - Treated as a single entity
   - Has file names
   - Access to file may be restricted or unrestricted
4. Database
   - Collection of related data.
   - Relationship exists among elements.

1. Fixed blocking
2. Variable length spanned blocking
3. Variable length unspanned blocking.

Fixed blocking

1. Fixed-length records (Figure 5) are used and an integral number of records are stored in a block.
2. Possibility of internal fragmentation.
3. Used for sequential files.

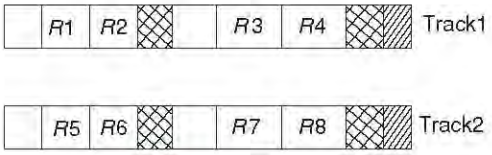


Figure 5 Fixed blocking.

Variable length spanned blocking Variable length records are used and are packed into blocks with no unused space. Some records can span two blocks. These do not limit the size of records.

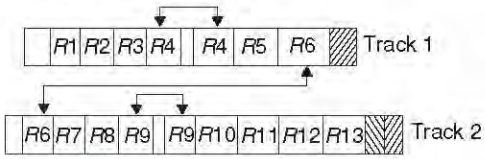


Figure 6 Variable blocking: spanned.

Variable length unspanned blocking Variable length records (Figure 7) are used, but spanning is not employed. There is wasted space in most blocks and limits record size.

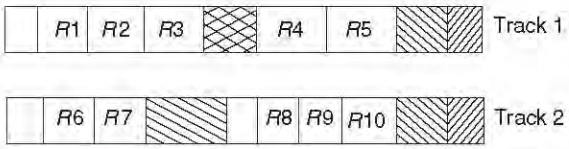
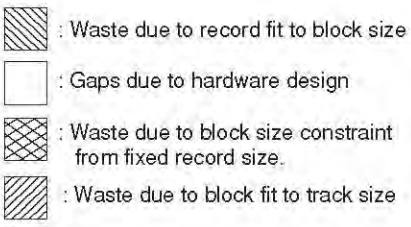


Figure 7 Variable blocking: unspanned.



Secondary Storage Management

1. Space (or blocks) must be allocated to files on disk.
2. Need to keep track of the space available (free blocks) for allocation to files.
3. Preallocation of blocks to files can be used to allocate space for files. For this, it needs to know the maximum size of the file at the time of creation.

File Allocation

Preallocation Versus Dynamic Allocation

1. A preallocation policy requires that the maximum size of a file be declared at the time of the file creation request. For many applications, it is difficult to estimate the file size.
   It is better to use dynamic allocation, which allocates space to a file in portions as needed.

Portion size The portion size which is allocated to a file may be

1. Variable, large contiguous portions.
2. Blocks

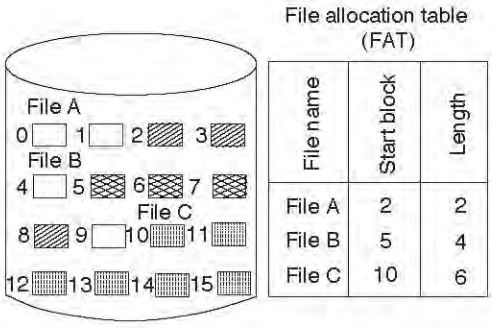
Some strategies for dealing with fragmentation of free space are as follows:

1. *First-fit*: Choose the first unused contiguous group of blocks of sufficient size.
2. *Best-fit*: Choose the smallest unused group that is of sufficient size.
3. *Nearest-fit*: Choose the unused group of sufficient size that is closest to the previous allocation.

File allocation methods It has three methods as follows:

1. Contiguous allocation
2. Chained allocation
3. Indexed allocation

Contiguous allocation Here, a single set of blocks is allo-cated to a file at the time of creation. Only a single entry in the file allocation table is created consisting of starting block and length of the file. It exhibits external fragmenta-tion and performs compaction.



Linked or chained allocation The allocation is done on basis of individual block. Each block contains a pointer to the next block in the chain. Only single entry is created in the file allocation table consisting of starting block and length of file. There occurs no external fragmentation and it is best for sequential files. There is no accommodation of principle of locality. If block size is n, then only $n - 1$ units of data are stored and 1 unit stores the link information.

Databases

UNIT 8

Chapter 1

ER Model and Relational Model

LEARNING OBJECTIVES

- *Data model*
- *Schemas*
- *Three-schema architecture*
- *ER model*
- *Types of attributes*
- *Mapping cardinality*
- *Complex attributes*
- *Entity types, entity sets and value sets*
- *Weak entity set*
- *Relational database*
- *NULL in tuples*
- *Inherent constraint*
- *Referential and entity integrity constraint*

INTRODUCTION

A database is a collection of related data. By data, we mean facts that can be recorded and that have implicit meaning.

Example: Consider the names, telephone numbers and addresses of the people. We can record this data in an indexed address book and store it as Excel fi le on a hard drive using a personal computer. This is a collection of related data with an implicit meaning and hence is a database.

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defi n ing, c onstructing, m anipulating a nd sharing databases among various users and applications.

1. Defining the database involves specifying the data types, structures, and constraints for the data to be stored in the database.
2. Constructing the database is the process of storing the data itself on some storage medium that is controlled by the DBMS
3. Manipulating a database includes such functions as querying the database to retrieve specific data, updating the database to reflect changes.
4. Sharing a database allows multiple users and programs to access the database concurrently.
5. Fundamental characteristics of the database approach is that it provides some level of data abstraction by hiding details of data storage that are not needed by users.

Data Model

A data model is a collection of concepts that can be used to describe the structure of a database. It provides the necessary means to achieve abstraction.

SCHEMAS

In any data model, it is important to distinguish between the description of the database and the database itself. The descrip-tion of a database is called the *database schema*, which is specifi ed during database design and is not expected to change frequently.

The actual data in a database may change frequently, for example the student database changes every time we add a student or enter a new grade for a student. The data in the database at a particular moment in time is called a *database state* or *snapshot*. It is also called the *current set of occurrences* or *instances in the database*.

The distinction between database schema and database state is very important. When we defi n e a new database, we specify its database schema only to the DBMS. At this point, the corresponding database state is the empty state with no data. We get the initial state of the database when the database is fi r st loaded with the initial data. The DBMS stores the description of the schema constructs and constraints, also called the *metadata* in the DBMS catalog so that DBMS software can refer to the schema whenever it needs. The schema is sometimes called the *intension*, and the database state an *extension* of the schema.

Chapter 2

Structured Query Language

LEARNING OBJECTIVES

- *Relational algebra*
- *Select operator*
- *Project operator*
- *Set operators*
- *Union compatible relations*
- *Union operation*
- *Aggregate operators*
- *Correlated nested queries*
- *Relational calculus*
- *Tuple relational calculus*
- *Tuple relational calculus*
- *DML*
- *Super key*
- *SQL commands*

Relational Algebra

1. A set of operators (unary or binary) that take relation instances as arguments and return new relations.
2. Gives a procedural method of specifying a retrieval query
3. Forms the core component of a relational query engine
4. *SQL* queries are internally translated into *RA* expressions
5. Provides a framework for query optimization

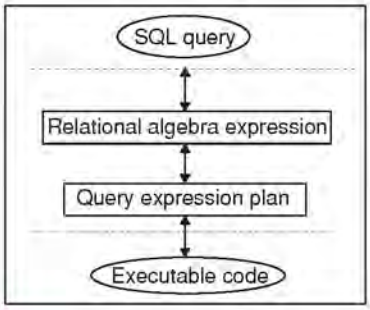
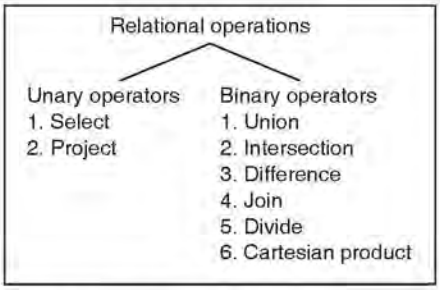


Figure 1 *Role of relational algebra in DBMS:*

Relational Operations

A collection of simple 'low-level' operations used to manipulate relations.

1. It provides a procedural way to query a database.
2. Input is one (or) more relations.
3. Output is one relation.



Select Operator (σ)

Select operator is an unary operator. It can be used to select those tuples of a relation that satisfy a given condition.

Notation: $\sigma_\theta(r)$
σ: Select operator(read as sigma)
θ: Selection condition
r: Relation name

Result is a relation with the same scheme as r consisting of the tuples in r that satisfy condition θ

Syntax: $\sigma_{\text{condition}}$ (relation)

Example:

Table 2.1 *Person*

Id	Name	Address	Hobby
112	John	12, SP Road	Stamp collection
113	John	12, SP Road	Coin collection
114	Mary	16, SP Road	Painting
115	Brat	18, GP Road	Stamp collection

Chapter 3

Normalization

LEARNING OBJECTIVES

- *Normalization*
- *Anomalies*
- *First normal form*
- *Functional dependency*
- *Inference rules*
- *Second normal form*
- *Third normal form*
- *Higher normal forms (Boyce-Codd normal form)*
- *Fifth normal form*
- *Courses*

NORMALIZATION

Database design theory includes design standards called *normal forms*. The process of making data and tables match these standards is called *normalizing data* or *data normalization*. By normalizing data, we eliminate redundant information and organize table to make it easier to manage the data and make future changes to the table and database structure. This process removes the insertion, deletion, and modification anomalies. In normalizing your data, we usually divide large tables into smaller, easier to maintain tables. We can then use the technique of adding foreign keys to enable connections between the tables.

Data normalization is part of the database design process and is neither specific nor unique to any particular RDBMS. These are in order, such as first, second, third, Boyce-Codd, fourth, and fifth normal forms. Each normal form represents an increasingly stringent set of rules; that is, each normal form assumes that the requirements of the preceding forms have been met. Many relational database designers feel that, if their tables are in third normal form, most common design problems have been addressed. However, the higher-level normal forms can be of use and are included here.

Database normalization is the process of removing redundant data from tables to improve storage efficiency, data integrity and scalability.

1. In the relational model, methods exists for quantifying how efficient a database is, these classifications are called q'.
2. Normalization generally involves splitting existing tables into multiple ones, which must be rejoined (or) linked each time a query is issued.
3. Edgar F. Codd originally established three normal forms: 1NF, 2NF, 3NF. There are others also, but 3NF is widely considered to be sufficient for most applications, most tables when reaching 3NF are also in BCNF (Boyce–Codd normal form).

Table 1

Title	Author 1	Author 2	I SBN	Subject	Pages	Publisher
Database system concepts	Abraham Silber schatz	Henry F. Korth	0072958863	My SQL, computers	1160	McGraw-Hill
OS concepts	Abraham Silberschatz	Henry F. Korth	0471694665	Computers	990	McGraw-Hill

Problems:

1. This table is not very efficient with storage.
2. This design doesn't protect data integrity.
3. This table doesn't scale well.

Anomalies

An anomaly is a variation that differs in some way from what is said to be normal, with respect to maintaining a database.

1. The basic operations performed on Databases are Record insertion, Record updation, Record deletion.

Chapter 4

Transaction and Concurrency

LEARNING OBJECTIVES

- *Transactions and concurrency control*
- *Transaction*
- *Transaction properties*
- *Uncommitted data*
- *Transaction processing systems*
- *Concurrency control with locking methods*
- *Two-phase locking to ensure serializability*
- *Concurrency control with time stamping methods*
- *Concurrency control with optimistic methods*
- *Recoverability*
- *Equivalence of schedules*
- *Testing for conflict serializability*

Introduction

A transaction is a logical unit of work. It begins, with the execution of a BEGIN TRANSACTION operation, and ends with the execution of a COMMIT or ROLLBACK operation. The logical unit of work that is, a transaction does not necessarily involve just a single database operation. Rather, it involves a sequence of several such operations as follows:

1. Database updates are kept in buffers in main memory and not physically written to disk until the transaction commits. That way, if the transaction terminates unsuccessfully, there will be no need to undo any disk updates.
2. Database updates are physically written to disk as part of the process of honouring the transaction's COMMIT request. That way if the system subsequently crashes, we can be sure that there will be no need to redo any disk updates.

Transactions and Concurrency Control

Database transactions reflect real-world transactions that are triggered by events, such as buying a product, registering for a course, or making a deposit in your checking account. Transactions are likely to contain many parts, for example, a sales transaction consists of at least two parts.

UPDATE inventory by subtracting number of units sold from the PRODUCT table's available quantity on hand and UPDATE the ACCOUNTS RECEIVABLE table in order to bill the CUSTOMER. All parts of a transaction must be completed to prevent data integrity problems. Therefore, executing and managing transactions are important database system activities.

Concurrency control is the management of concurrent transactions execution. When many users are able to access the database, the number of concurrent transactions tends to grow rapidly; as a result, concurrency control is especially important in multiuser database environments.

Transaction

A transaction is a logical unit of work that must be either entirely completed or aborted, no intermediate states are acceptable, that is, multicomponent transactions like the previously mentioned sale, must not be partially completed. If you read from and/or write to (update) the database, you create a transaction. Another example is using SELECT, to generate a list of table contents. Many real-world database transactions are formed by two or more database requests. A database request is the equivalent of a single SQL statement in an application program or transaction. Each database request generates several input/output operations. A transaction that changes the contents of a database must alter the database from one consistent state to another. A consistent database state is one in which all data integrity constraints are satisfied.

Example:

1. Checking an account balance:
   SELECT ACC_NUM, ACC_BALANCE
   FROM CHECKACC
   WHERE ACC_NUM = '0908110638';
   Even though we did not make any changes to the CHECKACC table, the SQL code represents a transaction, because we accessed the database.

Chapter 5

File Management

LEARNING OBJECTIVES

- *Files*
- *Memory hierarchies*
- *Description of disk devices*
- *File records*
- *Sorted files*
- *Hashing techniques*
- *Extendible hashing*
- *Index update*
- *Clustering index*
- *B-Trees*
- *B+Trees*
- *Over flow in internal node*

Files

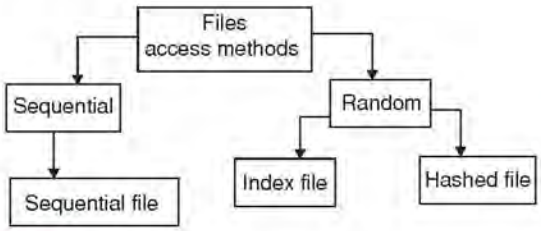
Databases are stored on magnetic disks as fi l es of records. Computer storage media form a storage hierarchy that includes two main categories.

Primary storage This category includes storage media that can be operated on, directly by CPU, such as the computer main memory and cache memory. Primary storage provides fast access but is of limited storage capacity.

Secondary storage This category includes magnetic disks, optical disks, and tapes. These devices usually have a larger capacity, less cost, and slower access to data. Data in secondary storage cannot be processed directly by the CPU, it must be copied into primary storage.

File Structure

Taxonomy of filestr ucture



Sequential file A sequential file is one in which records can only be accessed sequentially, one after another from beginning to end. Records are stored contiguously on the storage device.

Index files These files are used to access a record in the file. The entire index file is loaded into main memory data and indexes are stored in the same file. The term 'index file' is used as a synonym for the term 'database file'. The index file contains parameters that specify the name and location of file used to store DB.

Indexing Indexing mechanism is used to speed up access to desired data. An index file consists of records (called *index entries*) of the form.

Search-key	Pointer

Index filesaretypicall y much smaller than the original file.

Ordered indices In ordered index, index entries are stored, sorted on the search-key value.
Example: Author catalogue in library.

Memory Hierarchies

At the primary storage level, the memory hierarchy includes cache memory which is a static RAM.

The next level of primary storage is DRAM (dynamic RAM) which provides the main work area for the CPU for keeping programs and data and is called the *main memory*.

At the secondary storage level, the hierarchy includes magnetic disks, as well as mass storage in the form of CD-ROM (compact disk read-only memory) and tapes. Programs reside in DRAM and large permanent databases reside on secondary storage.

Another form of memory, *flash memory*, is non-volatile. Flash memories are high-density, high-performance memories using EEPROM (electrically erasable programmable read-only

Machine Learning

Unit 9

Machine Learning Basics

Learning Objectives

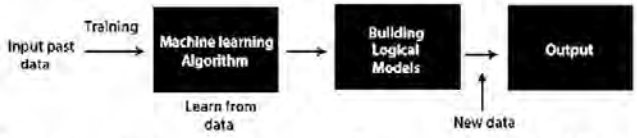
- Machine Learning Basic
- Deep Learning
- Life cycle of ML
- Supervised Learning
- Data Set
- Regression
- Types of Regression
- Unsupervised Learning
- Types of Clustering

Introduction

Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed.

How does Machine Learning work

A **Machine Learning system** learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. **The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.**



Importance of Machine Learning:

1. Rapid increment in the production of data
2. Solving complex problems, which are difficult for a human
3. Decision making in various sector including finance
4. Finding hidden patterns and extracting useful information from data.

Classification of Machine Learning

At a broad level, machine learning can be classified into three types:

1. **Supervised learning**
2. **Unsupervised learning**
3. **Reinforcement learning**

1) Supervised Learning

Supervised learning is a type of machine learning method in which we provide sample labeled data to the machine learning system in order to train it, and on that basis, it predicts the output.

Supervised learning can be grouped further in two categories of algorithms:

1. **Classification**
2. **Regression**

2) Unsupervised Learning

Unsupervised learning is a learning method in which a machine learns without any supervision.

The training is provided to the machine with the set of data that has not been labeled, classified, or categorized, and the algorithm needs to act on that data without any supervision. The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.

It can be further classifieds into two categories of algorithms:

1. **Clustering**
2. **Association**

3) Reinforcement Learning

Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with

Cyber Security
Cloud Computing
Data Center
Storage
Cryptography

Unit 10

Cyber Security

Learning Objectives

- **Basics on cyber Security**
- **Types of Security**
- **Different Types of Threats**
- **Types of Attack**
- **Man in the Middle Attack**
- **Firewalls**
- **Tools of Integrity**
- **Security Technologies**
- **Exercise Question**

What is Cyber Security?

The technique of protecting internet-connected systems such as computers, servers, mobile devices, electronic systems, networks, and data from malicious attacks is known as cybersecurity. We can divide cybersecurity into two parts one is cyber, and the other is security. Cyber refers to the technology that includes systems, networks, programs, and data. And security is concerned with the protection of systems, networks, applications, and information. In some cases, it is also called **electronic information security** or **information technology security**.

Types of Cyber Security

we can categorize cybersecurity in the following sub-domains:

- **Network Security**
- **Application Security**
- **Information or Data Security**
- **Identity management**.
- **Operational Security**
- **Mobile Security**
- **Cloud Security**
- **Disaster Recovery and Business Continuity Planning**
- **User Education**

Cyber Security Goals

Cyber Security's main **objective is to ensure data protection**. The security community provides a triangle of three related principles to protect the data from cyber-attacks. This principle is called the **CIA triad**. The CIA model is designed to guide policies for an organization's information security infrastructure. When any security breaches are found, one or more of these principles has been violated.

Confidentiality

Confidentiality is equivalent to privacy that avoids unauthorized access of information. It involves ensuring the data is accessible by those who are allowed to use it and blocking access to others. It prevents essential information from reaching the wrong people. **Data encryption** is an excellent example of ensuring confidentiality.

Integrity

This principle ensures that the data is authentic, accurate, and safeguarded from unauthorized modification by threat actors or accidental user modification. If any modifications occur, certain measures should be taken to protect the sensitive data from corruption or loss and speedily recover from such an event. In addition, it indicates to make the source of information genuine.

Availability

This principle makes the information to be available and useful for its authorized people always. It ensures that these accesses are not hindered by system malfunction or cyber-attacks.

Types of Cyber Security Threats

A threat in cybersecurity is a malicious activity by an individual or organization to corrupt or steal data, gain access to a network, or disrupts digital life in general. The cyber community defines the following threats available today:

File backups and snapshots economical and schedulable.	Block backups and mirrors require more storage.

What is NVMe?

NVMe, which stands for Non-Volatile Memory Express, is a computer storage interface that uses the Peripheral Component Interconnect Express (PCIe) bus to transfer data to and from solid-state drives (SSDs) at high speeds.

The NVMe specification is the industry standard for PCIe-based SSDs and is known for its lower latency, higher input/output per second (IOPS), lower power consumption, lower total cost of ownership, and improved scalability when compared to aging interfaces like SATA or SAS.

What is an NVMe SSD?

An NVMe solid-state drive, or SSD, is a high-speed storage device that implements the NVMe specification, which allows SSDs to use the PCIe bus to read and write data from and to the SSD. NVMe SSDs are much faster than SSDs using older but still widely used storage interfaces like SATA and SAS.

- The U.2 form factor is one of the most common SSD form factors and is typically found in high-performance workstations, servers, and storage systems.U.2 NVMe SSDs can usually be found connected to the PCIe bus via a miniSAS connector or an NVMe RAID controller.

- The M.2 form factor is usually reserved for internally mounted NVMe SSDs often found in small-form-factor computers, such as mini PCs and laptops. M.2 NVMe SSDs are often placed directly onto the motherboard, but they can also be installed using an NVMe RAID controller.

Cryptography

Learning Objectives

- **Encryption and Decryption**
- **Types of Agent**
- **Types**
- **Cryptographic Algorithm**
- **Cryptographic Attacks**

What is meant By Encryption?

Encryption is a process which transforms the original information into an unrecognizable form. This new form of the message is entirely different from the original message. That's why a hacker is not able to read the data as senders use an encryption algorithm. Encryption is usually done using key algorithms.

Data is encrypted to make it safe from stealing. However, many known companies also encrypt data to keep their trade secret from their competitors.

Encryption Process

What is meant by Decryption?

Decryption is a process of converting encoded/encrypted data in a form that is readable and understood by a human or a computer. This method is performed by un-encrypting the text manually or by using keys used to encrypt the original data.

Decryption process

Types of Keys

1) ***Symmetric Key:***
   Symmetric-key encryption are algorithms which use the same cryptographic keys for both encryption of plaintext and decryption of ciphertext.
2) ***Asymmetric Key:***
   Asymmetric encryption uses 2 pairs of key for encryption. Public key is available to anyone while the secret key is only made available to the receiver of the message. This boots security.
3) ***Public Key:***
   Public key cryptography is an encryption system which is based on two pairs of keys. Public keys are used to encrypt messages for a receiver.
4) ***Private Key:***
   Private key may be part of a public/ private asymmetric key pair. It can be used in asymmetric encryption as you can use the same key to encrypt and decrypt data.
5) ***Pre-Shared Key:***
   In cryptography, a pre-shared key (PSK) is a shared secret which was earlier shared between the two parties using a secure channel before it is used.

Discrete Mathematics

Unit II

Discrete Mathematics

Learning Objectives

- **Basic on Discrete Mathematics**
- **Probability**
- **Types of Probability**

Discrete Mathematics is a branch of mathematics involving discrete elements that uses algebra and arithmetic. It is increasingly being applied in the practical fields of mathematics and computer science. It is a very good tool for improving reasoning and problem-solving capabilities.

Topics in Discrete Mathematics

Though there cannot be a definite number of branches of Discrete Mathematics, the following topics are almost always covered in any study regarding this matter −

- Sets, Relations and Functions
- Mathematical Logic
- Group theory
- Counting Theory
- Probability
- Mathematical Induction and Recurrence Relations
- Graph Theory
- Trees
- Boolean Algebra

Probability Theory

How **likely** something is to happen. Many events can't be predicted with total certainty. The best we can say is how **likely** they are to happen, using the idea of probability.

Tossing a Coin

When a coin is tossed, there are two possible outcomes:

- heads (H) or
- tails (T)

We say that the probability of the coin landing H is ½
And the probability of the coin landing T is ½.

Throwing Dice

When a single die is thrown, there are six possible outcomes: **1, 2, 3, 4, 5, 6**. The probability of any one of them is *1*/**6**

Probability In general: Probability of an event happening =

$$\frac{\textit{Number of ways it can happen}}{\text{Total number of outcomes}}$$

Example-1: the chances of rolling a "4" with a die
Number of ways it can happen: 1 (there is only 1 face with a "4" on it)
Total number of outcomes: 6 (there are 6 faces altogether)
So the probability = *1*/**6**

Example-2: there are 5 marbles in a bag: 4 are blue, and 1 is red. What is the probability that a blue marble gets picked?
Number of ways it can happen: 4 (there are 4 blues)
Total number of outcomes: 5 (there are 5 marbles in total)
So the probability = *4*/**5** = 0.8

Probability Line

We can show probability on a Probability Line:

Probability is Just a Guide

Probability does not tell us exactly what will happen, it is just a guide

Example-3: toss a coin 100 times, how many Heads will come up?
Probability says that heads have a ½ chance, so we can **expect 50 Heads**. But when we actually try it we might get 48 heads, or 55 heads ... or anything really, but in most cases it will be a number near 50.

Words

Some words have special meaning in Probability:

Experiment: a repeatable procedure with a set of possible results.

Example-4: Throwing dice. We can throw the dice again and again, so it is repeatable.**The set of possible results from any single throw is {1, 2, 3, 4, 5, 6}**

Outcome: A possible result of an experiment.

(b) The proposition $(\exists n \in N)(n + 6 < 4)$ is false since $\{n \mid n + 6 < 4\} = \emptyset$.
(c) The symbol ∃ can be used to define the union of an indexed collection $\{Ai \mid i \in I\}$ of sets Ai as follows:
$\cup(Ai \mid i \in I) = \{x \mid \exists\, i \in I, x \mid \in Ai\}$

Negation of Quantified Propositions:

When we negate a quantified proposition, i.e., when a universally quantified proposition is negated, we obtain an existentially quantified proposition,and when an existentially quantified proposition is negated, we obtain a universally quantified proposition.

The two rules for negation of quantified proposition are as follows. These are also called DeMorgan's Law.

Example: Negate each of the following propositions:
1.∀x p(x)∧ ∃ y q(y)
Sol: ~.∀x p(x)∧ ∃ y q(y))
≅~∀ x p(x)∨~∃yq (y) (∴~(p∧q)=~p∨~q)
≅ ∃ x ~p(x)∨∀y~q(y)

2. (∃x∈U) (x+6=25)

Sol: ~(∃ x∈U) (x+6=25)
≅∀ x∈U~ (x+6)=25
≅(∀ x∈U) (x+6)≠25

3. ~(∃ x p(x)∨∀ y q(y)
Sol: ~(∃ x p(x)∨∀ y q(y))
≅~∃ x p(x)∧~∀ y q(y) (∴~(p∨q)= ~p∧~q)
≅ ∀ x ~ p(x)∧∃y~q(y))

Nested Quantifiers

If we use a quantifier that appears within the scope of another quantifier, it is called nested quantifier.

Example

- ∀ a∃bP(x,y)∀ a∃bP(x,y) where P(a,b)P(a,b) denotes a+b=0a+b=0
- ∀ a∀b∀cP(a,b,c)∀ a∀b∀cP(a,b,c) where P(a,b)P(a,b) denotes a+(b+c)=(a+b)+ca+(b+ c)=(a+b)+c

Note − ∀a∃bP(x,y)≠∃a∀bP(x,y)

Exercise

Question-1: Let $A = \{1, 2, 3, 4, 5\}$. Determine the truth value of each of the following statements:
(a) $(\exists x \in A)(x + 3 = 10)$ (c) $(\exists x \in A)(x + 3 < 5)$
(b) $(\forall x \in A)(x + 3 < 10)$ (d) $(\forall x \in A)(x + 3 \leq 7)$
(a) False. For no number in A is a solution to $x + 3 = 10$.
(b) True. For every number in A satisfies $x + 3 < 10$.
(c) True. For if $x0 = 1$, then $x0 + 3 < 5$, i.e., 1 is a solution.
(d) False. For if $x0 = 5$, then $x0 + 3$ is not less than or equal 7. In other words, 5 is not a solution to the given
condition.

Question-1: Determine the truth value of each of the following statements where U = {1, 2, 3} is the universal set:
(a) ∃x∀y, x2 < y + 1; (b) ∀x∃y, x2 + y2 < 12;
(c) ∀x∀y, x2 + y2 < 12.
(a) True. For if $x = 1$,
then 1, 2, and 3 are all solutions to $1 < y + 1$.
(b) True.
For each $x0$, let $y = 1$; then $x2$
$0 + 1 < 12$ is a true statement.
(c) False. For if $x_0 = 2$ and $y_0 = 3$, then $x_0^2 + y_0^2 < 12$ is not a true statement.

Question-1: Negate each of the following statements:
(a) $\exists x\ \forall y,\ p(x, y)$; *(b)* $\exists x\ \forall y,\ p(x, y)$; *(c)* $\exists y\ \exists x\ \forall z,\ p(x, y, z)$.
Use $\neg\, \forall x\ p(x) \equiv \exists x \neg p(x)$ and $\neg\, \exists x\ p(x) \equiv \forall x \neg p(x)$:
(a) $\neg\, (\exists x \forall y,\ p(x, y)) \equiv \forall x \exists y \neg p(x, y)$
(b) $\neg\, (\forall x \forall y,\ p(x, y)) \equiv \exists x \exists y \neg p(x, y)$
(c) $\neg\, (\exists y\ \exists x\ \forall z,\ p(x, y, z)) \equiv \forall y\ \forall x\ \exists z \neg p(x, y, z)$

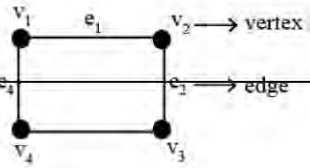
Graph Theory

Learning Objectives

- **Basic on Graph Theory**
- **Types of Graph**
- **Operation on Graph**

Introduction

A graph G = (V, E) consists of a set of objects V={v_1, v_2, v_3, … }called **vertices** (also called **points** or **nodes**) and other set E = {e_1, e_2, e_3,…… } whose elements are called **edges** (also



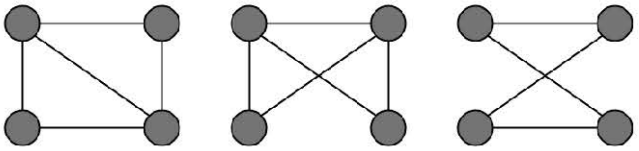
- Both the graphs G1 and G2 have different number of edges.
- So, Condition-02 violates.

Since Condition-02 violates, so given graphs can not be isomorphic.

∴ G1 and G2 are not isomorphic graphs.

Problem-12:

Which of the following graphs are isomorphic?



Sol^n-

Checking Necessary Conditions-

Condition-01:

- Number of vertices in graph G1 = 4
- Number of vertices in graph G2 = 4
- Number of vertices in graph G3 = 4

Here,

- All the graphs G1, G2 and G3 have same number of vertices.
- So, Condition-01 satisfies.

Condition-02:

- Number of edges in graph G1 = 5
- Number of edges in graph G2 = 5
- Number of edges in graph G3 = 4

Here,

- The graphs G1 and G2 have same number of edges.
- So, Condition-02 satisfies for the graphs G1 and G2.
- However, the graphs (G1, G2) and G3 have different number of edges.
- So, Condition-02 violates for the graphs (G1, G2) and G3.

Since **Condition-02** violates for the graphs (G1, G2) and G3, so they can not be isomorphic.

∴ G3 is neither isomorphic to G1 nor G2.

Since **Condition-02** satisfies for the graphs G1 and G2, so they may be isomorphic.

∴ G1 may be isomorphic to G2.

Now, let us continue to check for the graphs G1 and G2.

Condition-03:

- Degree Sequence of graph G1={2 ,2 ,3, 3}
- Degree Sequence of graph G2 = {2,2,3,3 }

Here,

- Both the graphs G1 and G2 have same degree sequence.
- So, Condition-03 satisfies.

Condition-04:

- Both the graphs contain two cycles each of length 3 formed by the vertices having degrees { 2 , 3 , 3 }
- It means both the graphs G1 and G2 have same cycles in them.
- So, Condition-04 satisfies.

Thus,

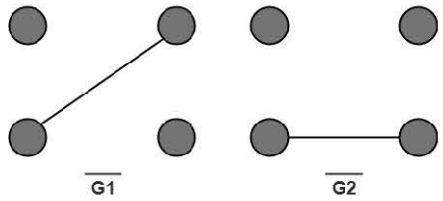
- All the 4 necessary conditions are satisfied.
- So, graphs G1 and G2 may be isomorphic.

Now, let us check the sufficient condition.

Checking Sufficient Condition-

We know that two graphs are surely isomorphic if and only if their complement graphs are isomorphic.

So, let us draw the complement graphs of G1 and G2. The complement graphs of G1 and G2 are-

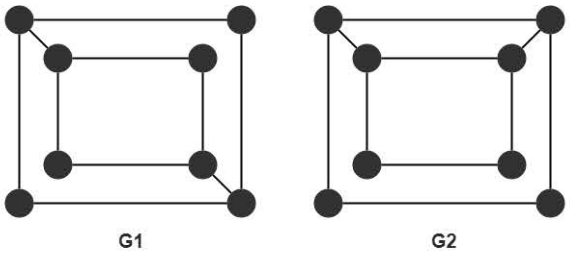


Clearly, complement graphs of G1 and G2 are isomorphic.

∴ Graphs G1 and G2 are isomorphic graphs.

Problem-13

Are the following two graphs isomorphic?



Sol^n-

Checking Necessary Conditions-

Condition-01:

- Number of vertices in graph G1 = 8
- Number of vertices in graph G2 = 8

Here,

- Both the graphs G1 and G2 have same number of vertices.
- So, Condition-01 satisfies.

Condition-02:

- Number of edges in graph G1 = 10

Object Oriented Programming With Java

Unit 12

```
7.      Test(int a, float b)
8.      {
9.          System.out.println("a = "+a+" b = "
+b);
10.     }
11.     public static void main (String args[])

12.     {
13.         byte a = 10;
14.         byte b = 15;
15.         Test test = new Test(a,b);
16.     }
17. }
```

The output of the following program is:
a = 10 b = 15
Here, the data type of the variables a and b, i.e., byte gets promoted to int, and the first parameterized constructor with the two integer parameters is called.

Question-45) What is the output of the following Java program?

Ans:

```
1.  class Test
2.  {
3.      int i;
4.  }
5.  public class Main
6.  {
7.      public static void main (String args[])

8.      {
9.          Test test = new Test();
10.         System.out.println(test.i);
11.     }
12. }
```

The output of the program is 0 because the variable i is initialized to 0 internally. As we know that a default constructor is invoked implicitly if there is no constructor in the class, the variable i is initialized to 0 since there is no constructor in the class.

Question-46) What is the output of the following Java program?

Ans:

```
1.  class Test
2.  {
3.      int test_a, test_b;
4.      Test(int a, int b)
5.      {
6.      test  a = a;
7.      test_b = b;
8.      }
9.      public static void main (String args[])
10.     {
11.         Test test = new Test();
12.         System.out.println(test.test  a+" "+test.test
        b);
13.     }
14. }
```

There is a **compiler error** in the program because there is a call to the default constructor in the main method which is not present in the class. However, there is only one parameterized constructor in the class Test. Therefore, no default constructor is invoked by the constructor implicitly.

Chapter-2

Learning Objectives

- **Static Variable**
- **Static Method**
- **Abstract Class**
- **This Keyword**

Question-47) What is the static variable?

Ans: The static variable is used to refer to the common property of all objects (that is not unique for each object), e.g., The company name of employees, college name of students, etc. Static variable gets memory only once in the class area at the time of class loading. Using a static variable makes your program more memory efficient (it saves memory). Static variable belongs to the class rather than the object.

```
1. //Program of static variable
2.
3. class Student8{
4.   int rollno;
5.   String name;
6.   static String college ="ITS";
7.
8.   Student8(int r,String n){
9.   rollno = r;
10.          name = n;
11.          }
12.        void display (){System.out.println(rollno+" "+na
   me+" "+college);}
13.
14.        public static void main(String args[]){
15.        Student8 s1 = new Student8(111,"Karan");
16.        Student8 s2 = new Student8(222,"Aryan");
17.
18.        s1.display();
19.        s2.display();
20.        }
21.        }
```

Question-48) What is the static method?

Ans:

- A static method belongs to the class rather than the object.
- There is no need to create the object to call the static methods.
- A static method can access and change the value of the static variable.

Question-49) What are the restrictions that are applied to the Java static methods?

Ans: Two main restrictions are applied to the static methods.

- The static method can not use non-static data member or call the non-static method directly.

- **this** can be passed as an argument in the constructor call.
- **this** can be used to return the current class instance from the method.

Question-61) Can we assign the reference to this variable?

Ans: No, this cannot be assigned to any value because it always points to the current class object and this is the final reference in Java. However, if we try to do so, the compiler error will be shown. Consider the following example.

```
1. public class Test
2. {
3.     public Test()
4.     {
5.         this = null;
6.         System.out.println("Test class constructor ca
   lled");
7.     }
8.     public static void main (String args[])
9.     {
10.        Test t = new Test();
11.    }
12. }
```

Question-62) Can this keyword be used to refer static members?

Ans: Yes, It is possible to use this keyword to refer static members because this is just a reference variable which refers to the current class object. However, as we know that, it is unnecessary to access static variables through objects, therefore, it is not the best practice to use this to refer static members. Consider the following example.

```
1.  public class Test
2.  {
3.      static int i = 10;
4.      public Test ()
5.      {
6.          System.out.println(this.i);
7.      }
8.      public static void main (String args[])
9.      {
10.         Test t = new Test();
11.     }
12. }
```

Question-63) How can constructor chaining be done using this keyword?

Ans: Constructor chaining enables us to call one constructor from another constructor of the class with respect to the current class object. We can use this keyword to perform constructor chaining within the same class. Consider the following example which illustrates how can we use this keyword to achieve constructor chaining.

```
1.  public class Employee
2.  {
3.      int id,age;
4.      String name, address;
5.      public Employee (int age)
6.      {
7.          this.age = age;
8.      }
9.      public Employee(int id, int age)
10.     {
11.         this(age);
12.         this.id = id;
13.     }
14.     public Employee(int id, int age, String name,
    String address)
15.     {
16.         this(id, age);
17.         this.name = name;
18.         this.address = address;
19.     }
20.     public static void main (String args[])
21.     {
22.         Employee emp = new Employee(105, 22, "
    Vikas", "Delhi");
23.         System.out.println("ID: "+emp.id+" Name:
    "+emp.name+" age:"+emp.age+" address: "+em
    p.address);
24.     }
25.
26. }
```

Question-64) What are the advantages of passing this into a method instead of the current class object itself?

Ans: As we know, that this refers to the current class object, therefore, it must be similar to the current class object. However, there can be two main advantages of passing this into a method instead of the current class object.

- this is a final variable. Therefore, this cannot be assigned to any new value whereas the current class object might not be final and can be changed.
- this can be used in the synchronized block.

Chapter-3

Learning Objectives

- **Inheritance**
- **Types of Inheritance**
- **Abstract Class, Method**
- **This and Super keyword**
- **Overriding**

Question-65) What is the Inheritance?

Ans: *When one object acquires all the properties and behaviors of a parent object*, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

Important terminology:

- **Super Class:** The class whose features are inherited is known as superclass(or a base class or a parent class).

```java
import java.io.Console;
class ReadStringTest{
public static void main(String args[]){
Console c=System.console();
System.out.println("Enter your name: ");
String n=c.readLine();
System.out.println("Welcome "+n);
}
}
```

Question-208) What is Socket?

Ans: A socket is simply an endpoint for communications between the machines. It provides the connection mechanism to connect the two computers using TCP. The Socket class can be used to create a socket.

Question-209) What are the steps that are followed when two computers connect through TCP?

Ans: There are the following steps that are performed when two computers connect through TCP.

- The ServerSocket object is instantiated by the server which denotes the port number to which, the connection will be made.
- After instantiating the ServerSocket object, the server invokes accept() method of ServerSocket class which makes server wait until the client attempts to connect to the server on the given port.
- Meanwhile, the server is waiting, a socket is created by the client by instantiating Socket class. The socket class constructor accepts the server port number and server name.
- The Socket class constructor attempts to connect with the server on the specified name. If the connection is established, the client will have a socket object that can communicate with the server.
- The accept() method invoked by the server returns a reference to the new socket on the server that is connected with the server.

Question-210) Write a program in Java to establish a connection between client and server?

Ans: Consider the following program where the connection between the client and server is established.

File: MyServer.java

```java
import java.io.*;
import java.net.*;
public class MyServer {
public static void main(String[] args){
try{
ServerSocket ss=new ServerSocket(6666
);
Socket s=ss.accept();//establishes connec
tion
DataInputStream dis=new DataInputStre
am(s.getInputStream());
String str=(String)dis.readUTF();
System.out.println("message= "+str);
ss.close();
}catch(Exception e){System.out.println(
e);}
}
}
```

File: MyClient.java

```java
import java.io.*;
import java.net.*;
public class MyClient {
public static void main(String[] args) {
try{
Socket s=new Socket("localhost",6666);
DataOutputStream dout=new DataOutputStrea
m(s.getOutputStream());
dout.writeUTF("Hello Server");
dout.flush();
dout.close();
s.close();
}catch(Exception e){System.out.println(e);}
}
}
```

Question-211) How do I convert a numeric IP address like 192.18.97.39 into a hostname like java.sun.com?

Ans: By InetAddress.getByName("192.18.97.39").getHostName() where 192.18.97.39 is the IP address. Consider the following example.

```java
import java.io.*;
import java.net.*;
public class InetDemo{
public static void main(String[] args){
try{
InetAddress ip=InetAddress.getByName("195.
201.10.8");

System.out.println("Host Name: "+ip.getHost
Name());
}catch(Exception e){System.out.println(e);}

}
}
```

Chapter-6

Learning Objectives

- **Wrapper Class, Singleton Class**
- **Applet**
- **Searching and Sorting Problem**
- **Threading**
- **Java Collections**

Question-212) What are wrapper classes?

Ans: Wrapper classes are classes that allow primitive types to be accessed as objects. In other words, we can say that wrapper classes are built-in java classes which allow the conversion of objects to primitives and primitives to

Step 2 –
Value of x is initialized as 0 by calling it using object 1.
Step 3 –
Function increament() of class access is used by object 1. After this the value of static variable x is increased to 1.
Step 4 –
Function increament() of class access is used by object 2. After this the value of static variable x is increased to 2.
Step 5 –
Print function is used to print the sum of variable x accessed by object 1 and object 2.
As we know that variable x is a static variable hence its value for both object 1 and object 2 is same. Hence the values of x for both object 1 and object 2 is 2 so the output is 2+2=4.

Question-9 **Find the output of following java program.**

```
class MainClass
{
        public static void main (String arg[])
        {
                System.out.print('h' + 'i');
                System.out.println();
        }
}
```

Output
209
Explanation
In this program the output is sum of ASCII values of h and i. The ASCII value of h is **104** and ASCII value of i is **105**. The addition of **104** and **105** is **209**. Hence the **output is 209**. This happens because in print function if we use arithmetic operations between the characters then print function performs the arithmetic operations between the ASCII values of given characters.

Question-10 **Find the output of following java program.**

```
class MainClass
{
        public static void main (String arg[])
        {
                int i;
                for( i=1; 1; i++)
                {
                        System.out.print(i);
                        break;
                }
                System.out.println();
        }
}
```

Output
The output of this code will not generated due to compilation error.
Explanation
Like C and C++, we cannot use 0 and 1 for checking condition in case of java. In java we have to use bool variable **'true'** and **'false'** to check conditions of the conditional operators.

The correct code should be:

```
class MainClass
{
        public static void main (String arg[])
        {
                int i;
                for( i=1; true; i++)
                {
                        System.out.print(i);
                        break;
                }
                System.out.println();
        }
}
```

Question-11 **Find the output of following java program.**

```
class MainClass
{
        public static void main (String arg[])
        {
                System.out.print( func() );
                System.out.println();
        }
        int func()
        {
                int test = 100;
                return test;
        }
}
```

Output
The output of this code will not generate due to compilation error.
Explanation
Like C and C++, we cannot call non static methods in a static method in java. To remove compilation error in this program we have to put static keyword before the function name. If we make the function static then this function can be call by main method in java.
The correct code should be:

```
class MainClass
{
        public static void main (String arg[])
        {
                System.out.print( func() );
                System.out.println();
        }
        static int func()
        {
                int test = 100;
                return test;
        }
}
```

Question-12 What is the Output of the following Java Program.

```
// filename Main.java
class Test {
   protected int x, y;
}
class Main {
   public static void main(String args[]) {
      Test t = new Test();
      System.out.println(t.x + " " + t.y);
   }
```

Microprocessor 8085

LEARNING OBJECTIVES

After reading this chapter, you will be able to understand:

- Organization of microprocessor based system
- Microprocessor 8085
- Instructions classifications
- Instruction word size
- Operations in micro-processor
- 8085 signals
- Peripheral – mapped I/O
- Memory mapped I/O
- Additional instructions
- Stack
- Subroutine
- Restart (RST) instructions

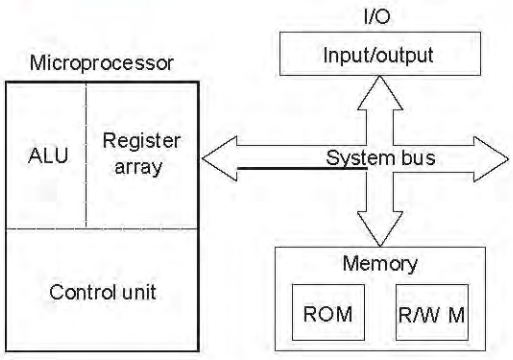
Introduction

The microprocessor is a programmable integrated device that has computing and decision-making capability similar to that of the central processing unit (CPU) of computer.

The microprocessor communicates and operates in the binary numbers 0 and 1 called bits.

Each microprocessor has a fixed set of instruction in the form of binary patterns called machine language to make it easier to understand the binary instructions that are given abbreviated names, called mnemonics, which form the assembly language for a given microprocessor.

Organization of MicroProcessor-based System



Microprocessor-based systems include three components: microprocessor, input/output (I/O) and memory

These components are organized around a common communication path called bus.

Microprocessor

The microprocessor is a clock driven semiconductor device consisting of electric logic circuits manufactured by using either LSI, or VLSI technique.

Arithmetic Logic Unit

The ALU unit performs arithmetic operations as addition, subtraction, and logic operations like AND, OR, exclusive OR.

Register Array

Microprocessor consists of various registers identified by *B*, *C*, *D*, *E*, *H* and *L*. These registers are primarily used to store data temporarily during the execution of a program.

Control Unit

The control unit provides the necessary timing and control signals to all the operations in the microcomputer.

MicroProcessor 8085

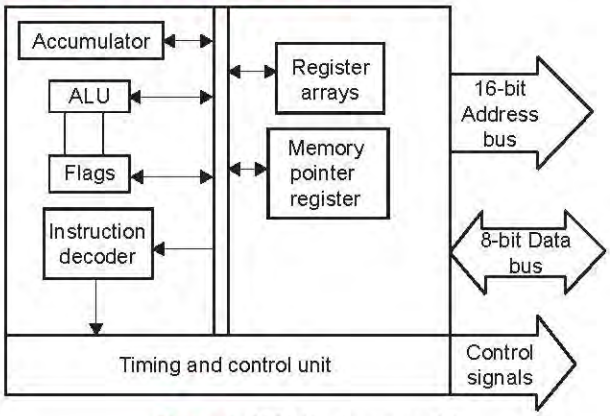


Figure 1 8085 Hardware model

Tele-Communication

Unit 14

TeleCommunication Basics

Learning Objectives

- Telecommunication Basic
- Mobile Communication
- GSM, GPRS, UMTS, LTE
- MANET
- Satellite System
- Digital Transmission
- Analog Transmission
- Wireless Communication
- 5G Technology

Telecommunication

Telecommunication is the transmission of information by various types of technologies over wire, radio, optical or other electromagnetic systems.

Question-1: Advantages of Telecommunication:

- Quick and accessible communication
- Lack of time period
- Saves time
- Saves gasoline (do not need to drive distance)
- More than two people can communicate with at least one another at an equivalent time
- Next "best thing" to being there
- Easy to exchange ideas and knowledge via phone and/or fax
- Worldwide access
- Easy access to the people you would like to contact.
- Less effort in using transportation just to satisfy a private personally.
- You can just occupy your home and use a telephone or a cellphone if you would like to speak to someone.
- Enable end-users to speak electronically and share hardware, software, and data resources.
- This makes corporation to do the transaction at the point only and in a very fast way from many remote locations, exchange business documents electronically with customers and suppliers, or remotely monitor and control production processes.
- Interconnect the pc systems of a business so their computing power is often shared by end-users throughout an enterprise.
- Make the organization work with collaboration and communication among the staff inside and out of doors a corporation.
- Speed
- Develops new products and inventions

Question-2: Disadvantages of Telecommunication:

- Cultural Barrier
- Misunderstanding
- Prank calls
- Sometimes expensive
- High electric bills
- Remote areas don't have access
- Remote areas might not be ready to afford the necessary equipment
- Cannot see whom you're speaking with
- Cannot see facial expressions, therefore results in misunderstandings
- Cultural barriers
- Poor connections or downed power lines during/after storms

Mobile technology

Mobile technology is technology that goes where the user goes. It consists of portable two-way communications devices, computing devices and the networking technology that connects them.

Question-3: Tracing the Journey of Communications Technology

(I) 1G: Voice Only

1. Cell phones began with 1G technology in the 1980s. 1G is the first generation of wireless cellular technology. 1G supports voice only calls.
2. 1G is analog technology, and the phones using it had poor battery life and voice quality, little security, and were prone to dropped calls.
3. The maximum speed of 1G technology is 2.4 Kbps.

(II) 2G: SMS and MMS

1. Cell phones received their first major upgrade when their technology went from 1G to 2G. This leap took place in Finland in 1991 on GSM networks and effectively took cell phones from analog to digital communications.
2. The 2G telephone technology introduced call and text encryption, along with data services such as SMS, picture messages, and MMS.
3. Although 2G replaced 1G and is superseded by later technology versions, it's still used around the world.

Unguided media is divided into three types of wave:

1. **Radio Wave**
2. **Micro Wave**
3. **Infrared Wave**

Radio Wave

The frequencies of electromagnetic wave lies between 3 kHz and 1 GHz are known as the Radio wave. Radio waves are mostly omnidirectional. When an antenna sends the radio waves, they are transmitted those waves in all directions. This means that it can send and receive those waves which do not required the alliance. Radio waves are mostly used in multicast communication, such as television, FM radio, cordless phones, and paging systems. Shown in below omnidirectional.

Micro Wave

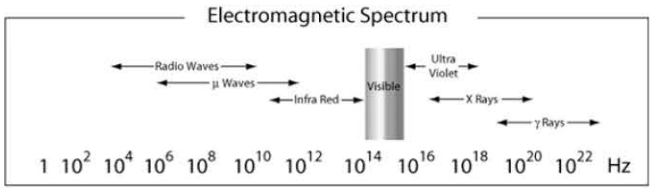
Micro-waves are those electromagnetic waves whose frequency range lies from 1 GHz to 300 GHz. It is unidirectional. These waves move in the unidirectional from the sender to the receiver. This means that it can send and receive those waves which have required the alliance. Micro-waves are mostly used in unicast communication (one to one communication), such as microwave (oven), cellular telephone, and satellite network.

Infrared Wave

Infrared waves are those electromagnetic waves whose frequency range lies from 300 GHz to 400 THz. Infrared waves are mostly used for short-range communication, such as television remote, FM remote, AC remote, CCTV, and wireless keyboard or mouse.

Infrared Radiation - Electromagnetic Waves

The visible light that we see every day is really a small portion of the electromagnetic spectrum. The electromagnetic spectrum includes all types of radiation ranging from the X-rays used at the hospitals, to radio waves used for communication.



Radiation in the electromagnetic spectrum is categorised by wavelength. Short wavelength radiation such as Gamma, X-rays and ultraviolet is of high energy and can be very dangerous. Longer wavelength radiation such as radio, microwaves and infrared are less harmful. In this article, we will be discussing infrared radiation and its characteristics in detail. Although infrared radiation is not visible, humans can sense it as heat. To experience infrared radiation "first-hand, put your hand next to a hot oven!

What is Infrared Radiation?

Infrared radiation (IR), sometimes known as infrared light, is electromagnetic radiation (EMR) with wavelengths longer than those of visible light. Hence, it is undetectable by the human eye, although IR of wavelengths up to 1050 nanometers (nm)s from specially pulsed lasers can be seen by humans under certain conditions. Infrared light extends from the suggested red edge of the visible spectrum at 700 nanometers to 1 millimetre. Most of the thermal radiation emitted by objects near room temperature is infrared. As with all **EMR**, IR carries radiant energy and behaves both like a wave and like its quantum particle, the photon. Depending on the wavelength and frequency, infrared is commonly divided into five categories as ***near-wavelength, short-wavelength, mid-wavelength, long-wavelength and far-infrared***.

The wavelength of Infrared Radiation

We already know that the wavelength of the infrared radiation is between 700 nm to 1 mm, which is between the red limit of the visible spectrum. But following is the classification of bands based on the spectral range 1μm and 50μm:

- 1μm to 3μm which is known as the Band I or Short Wave Infrared
- 3μm to 5μm which is known as the Band II or Middle Wave Infrared
- 8μm to 14μm which is known as the Band III or Long Wave Infrared

Characteristics of Regions of Infrared

Near-infrared	The wavelength ranges from 0.75 to 1.4 micrometres. This is used in material science, fibre optic communication, and in the medical field.
Short wavelength infrared	The wavelength ranges from 1.4 to 3 micrometres. This is used in telecommunications and for military purposes.

International Multimedia Teleconferencing Consortium. Industry forum for promoting interoperability trials.

IMUX
Inverse Multiplexer. Device that bonds two or more BRI lines to form a higher rate channel.

In-band signaling
Signaling made up of defined bits which pass within the data transmission stream.

Instant Messaging (IM)
A communications service that enables you to create a private chat room with another individual in order to communicate in real time over the Internet.

Incoming Call Routing
See default extension, DID, MSN, sub-addressing, TCS4.

IP
Internet Protocol. Packet-based protocol for delivering data across networks.

IP Address
The unique address of a computer attached to a TCP/IP network. IP addresses are 32 bits long. Each octet is represented in decimal and is separated by dots.

IP Multicast
A means of simultaneous transmission of data from a server to a group of selected users on a TCP/IP network, (internal, intranet or Internet). IP multicast is used for streaming audio and video
over the network.

IP Network
A network t hat uses the TCP/IP protocol.

IP Telephony
A set of technologies that enables voice, data and video collaboration over existing IP-based LANS, WANs, and the Internet. IP technology uses open IETF and ITU standards to move multimedia traffic over any network that uses IP.

IRQ
INFORMATION REQUEST Message. - A RAS message in which the Gatekeeper asks the endpoint for its current status.

IRR
INFORMATION REQUEST RESPONSE Message - A RAS message that the Gatekeeper sends to the calling endpoint, rejecting the IRQ.

ISDN
Integrated Services Digital Network. A set of standards that provide a common architecture for the development and deployment of digitally integrated communications services. A set of standardized customer interfaces and signaling protocols for delivering digital circuit-switched voice / data / video and packet-switched data services.

ISDN Rollover
In RADVISION implementations, Gateway support for the ISDN Rollover feature ensures that a call is completed even when call volume is high. ISDN Rollover requires support by the PSTN.

ISO
International Standardization Organization. International standards body concerned with non-telecommunications issues.

ITU
International Telecommunications Union. Organization composed of the telecommunications administrations of the participating nations. Focus is the maintenance and extension of international cooperation for improving telecommunications development and applications.

IXC
Inter Exchange Carrier. Common carrier providing communications channels between local companies (LECs, or Local Exchange Carriers). Also known as long distance carriers such as AT&T, LCI, LDDS, MCI, US Sprint, WilTel, etc.

Jitter
The result of a change in latency or the tendency towards lack of synchronization caused by mechanical or electrical changes. Technically, jitter is the phase shift of digital pulses over a transmission medium.

Jitter Buffer
A portion of memory specifically allocated to storing IP packets awaiting transmission, or to storing received IP packets. The buffer facilitates flow control by capturing IP packets and then transmitting packets as "playback" using speeds and rates of delay that the destination device can handle without causing packet loss through overloading.

Jitter Buffer Management
Jitter buffer management represents the trade-off between a larger buffer and increased rates of jitter.

Latency
A measure of accumulated waiting time or delay, representing the length of time required for information to pass through a network

LDAP
Lightweight Directory Access Protocol - A protocol for accessing online directory services. LDAP is both an information model and a protocol for querying and manipulating the model.

LED
Light Emitting Diode. A display technology that uses a semiconductor diode that emits light when charged. LEDs usually indicate both correct and problematic operation.

Mbps
Megabits per second. A unit of measure of data of 1,000,000

Electromagnetics Spectrum

Unit 15

Electromagnetic Spectrum

Learning Objectives

- Spectrum Basic
- Frequency Band
- Radio Waves
- Spectrum Radiation

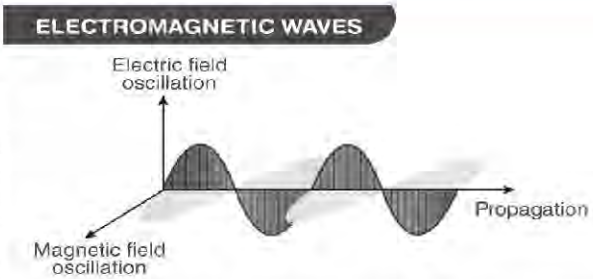
Question-1: What are Electromagnetic Waves?

Electromagnetic waves are also known as EM waves that are produced when an electric field comes in contact with the magnetic field. It can also be said that electromagnetic waves are the composition of oscillating electric and magnetic fields. Electromagnetic waves are solutions of Maxwell's equations, which are the fundamental equations of electrodynamics.

Question-2: How are Electromagnetic waves formed?

- Generally, an electric field is produced by a charged particle. A force is exerted by this electric field on other charged particles. Positive charges accelerate in the direction of the field and negative charges accelerate in a direction opposite to the direction of the field.
- The Magnetic field is produced by a moving charged particle. A force is exerted by this magnetic field on other moving particles. The force on these charges is always perpendicular to the direction of their velocity and therefore only changes the direction of the velocity, not the speed.
- So, the electromagnetic field is produced by an accelerating charged particle. Electromagnetic waves are nothing but electric and magnetic fields travelling through free space with the speed of light c. An accelerating charged particle is when the charged particle oscillates about an equilibrium position. If the frequency of oscillation of the charged particle is f, then it produces an electromagnetic wave with frequency f. The wavelength λ of this wave is given by λ = c/f. Electromagnetic waves transfer energy through space.

Graphical Representation of Electromagnetic Waves



Electromagnetic waves are shown by a sinusoidal graph. It consists of time-varying electric and magnetic fields which are perpendicular to each other and are also perpendicular to the direction of propagation of waves. Electromagnetic waves are transverse in nature. The highest point of the wave is known as crest while the lowest point is known as a trough. In vacuum, the waves travel at a constant velocity of 3×10^8 m.s^{-1}.

***Question-3:* Mathematical Representation of Electromagnetic Wave**

A plane Electromagnetic wave travelling in the x-direction is of the form

$E(x,t)=E_{max}\cos(kx-\omega t+\Phi)$
$B(x,t)=B_{max}\cos(kx-\omega t+\Phi)$

In the electromagnetic wave, E is the electric field vector and B is the magnetic field vector.

Maxwell gave the basic idea of electromagnetic waves, while Hertz experimentally confirmed the existence of an electromagnetic wave.

The direction of propagation of the electromagnetic wave is given by vector cross product of the electric field and magnetic field. It is given as:

$$\vec{E} \times \vec{B}$$

Electromagnetic Spectrum

Linux, UML Case and Class Diagram

Unit 16

Part A Linux

Linux Overview

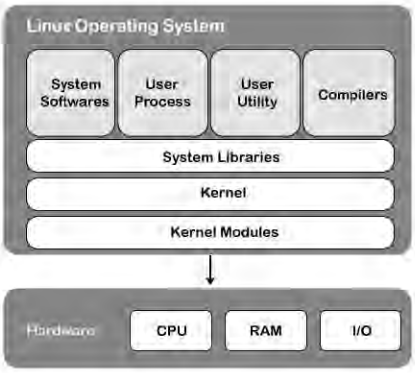
LEARNING OBJECTIVES

- *OverView On Linux OS*
- *Linux Command*
- *Shell Script*
- *Program Using Shell script*

Linux OS

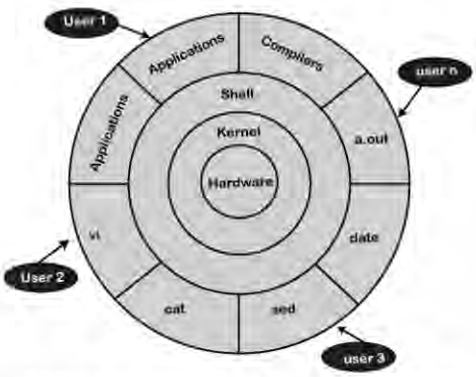
Linux is an open-source operating system like other operating systems such as Microsoft Windows, Apple Mac OS, iOS, Google android, etc. An operating system is a software that enables the communication between computer hardware and software.

Linux OS has following components:



1) Kernel

Linux kernel is the core part of the operating system. It establishes communication between devices and software. Moreover, it manages system resources. It has four responsibilities:



- **device management:** A system has many devices connected to it like CPU, a memory device, sound cards, graphic cards, etc. A kernel stores all the data related to all the devices in the device driver (without this kernel won't be able to control the devices). Thus kernel knows what a device can do and how to manipulate it to bring out the best performance. It also manages communication between all the devices. The kernel has certain rules that have to be followed by all the devices.
- **Memory management:** Another function that kernel has to manage is the memory management. The kernel keeps track of used and unused memory and makes sure that processes shouldn't manipulate data of each other using virtual memory addresses.
- **Process management:** In the process, management kernel assigns enough time and gives priorities to processes before handling CPU to other processes. It also deals with security and ownership information.
- **Handling system calls:** Handling system calls means a programmer can write a query or ask the kernel to perform a task.

2) System Libraries

System libraries are special programs that help in accessing the kernel's features. A kernel has to be triggered to perform a task, and this triggering is done by the applications. But applications must know how to place a system call because each kernel has a different set of system calls.

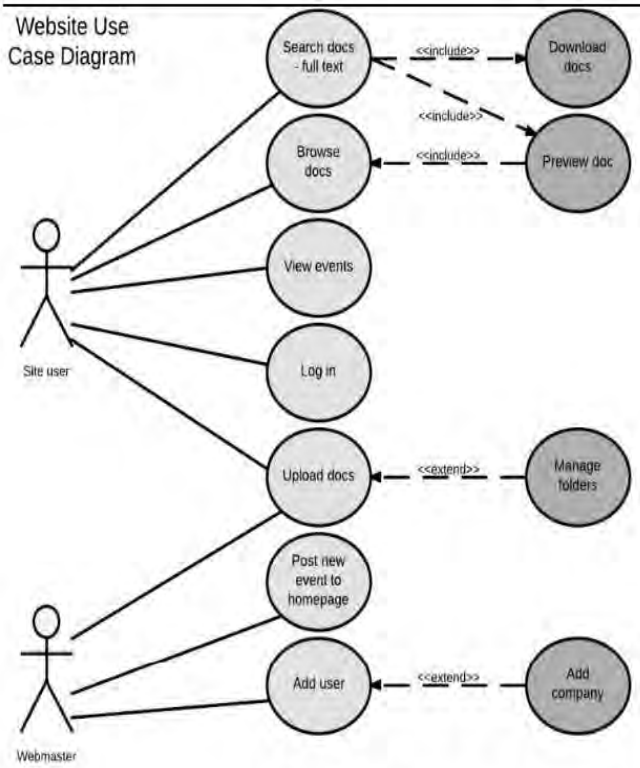
3) System Tools

Linux OS has a set of utility tools, which are usually simple commands. It is a software which GNU project has written and publish under their open source license so that software is freely available to everyone.

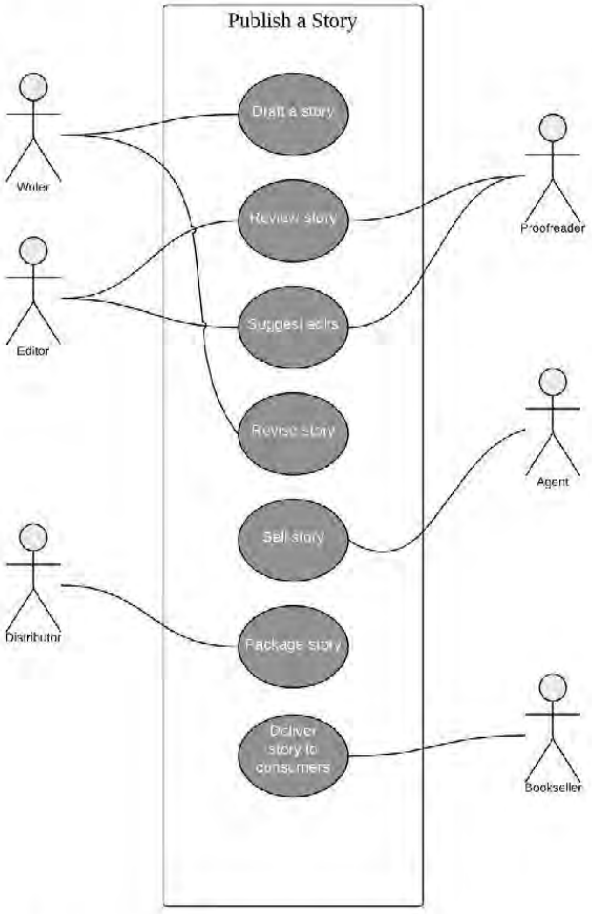
4) Development Tools

With the above three components, your OS is running and working. But to update your system, you have additional tools and libraries. These additional tools and libraries are written by the programmers and are called toolchain.

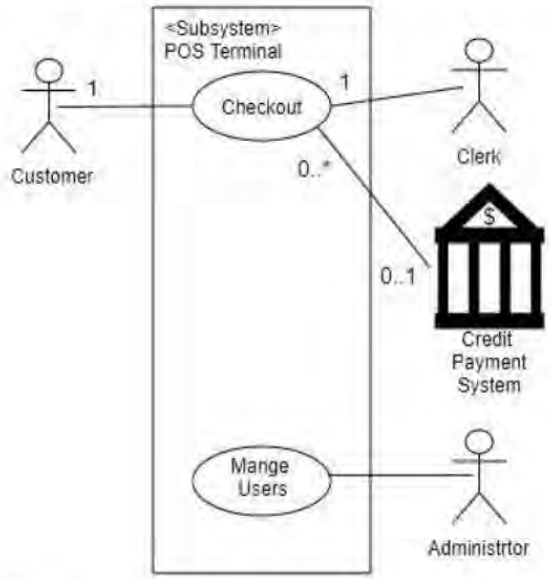
5) End User Tools



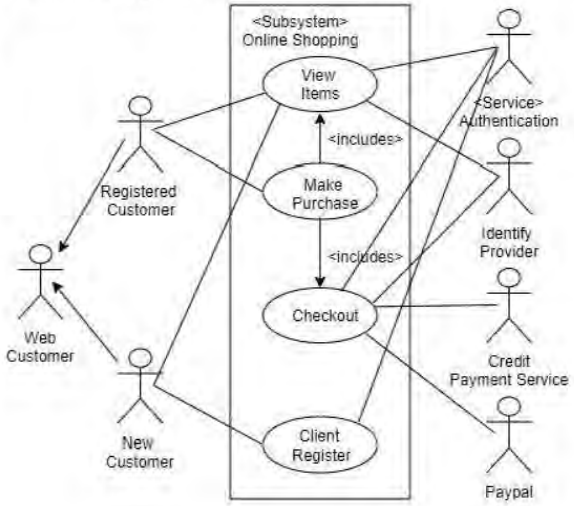
Book publishing use case diagram example



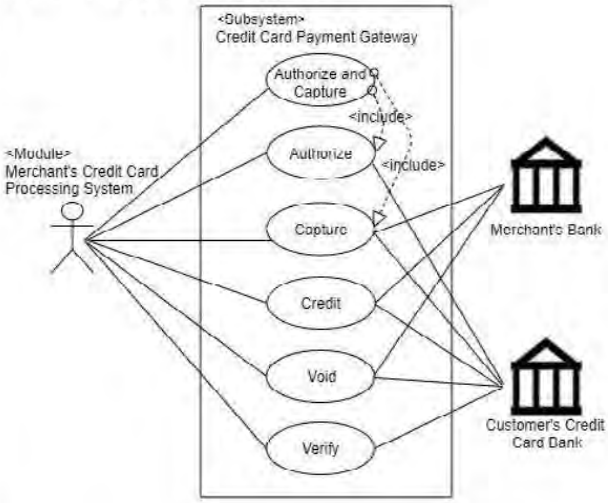
Point of Sales (POS) terminal



Online shopping use case diagrams



Credit card processing system



The **Document Management System (DMS)** use case diagram example below shows the actors and use cases of

Difference Hub

Hardware
Software
Internet
Database
Networking
Programming
Operating System

Unit 17

Difference

Learning Objectives

- **Basic Idea on Different Terms**
- **Clarify Idea**
- **Exam Preparation**
- **Presentation Idea on Exam Papers.**
- **Difference on Different Terms Used in ICT.**

Question-1: Difference between Intranet and Internet:

Internet	Intranet
It is a medium such as optical fiber cable that connects billions of computers with each other to establish a worldwide network.	It is a small, private network as it belongs to a specific organization.
It has billions of users as it is a public network with a worldwide presence.	It has limited users.
It is not as safe as an intranet.	It is a safer network than the internet.
It can be assessed or used by anyone using an internet-enable device, such as laptop, mobile phone, etc.	Only authorized persons can use this network.
It offers a wide range of information, such as news, blogs, websites, etc.	It offers limited information related to its organization's work, policies, updates, etc.
It is not owned by a single person or an organization.	It can be owned by a person or an organization.

Question-2: Difference between Static Website and Dynamic Website:

Static Website	Dynamic Website
As the name suggests, Its webpages do not change in terms of design, content, etc. The information or content remains the same.	As the name suggests, the webpages keep changing as per users' requirements such as Facebook profile pages and an E-commerce site. So, the content does not remain the same.
It mainly uses HTML and CSS and does not require server-side scripting, application server, and database.	It requires server-side scripting, application server, and database to create and send dynamic webpages to the client.
It has a limited number of pages.	It may contain thousands of pages in the database.
Its hosting cost is low, as HTML files need less space on the server.	Its hosting cost is higher as dynamic pages need more space on the server.
It requires low maintenance.	It requires high maintenance.
It loads quickly as it involves the use of mark-up languages to create a webpage.	It takes more time to load due to the more processing time.
It lacks the Content management feature.	It makes use of the Content Management Feature.
The content of the webpage cannot be changed during runtime.	The webpage content can be changed during runtime.
It does not require interaction with a database.	Interaction with the database occurs.
It is more secure or fewer chances of it getting hacked as it doesn't use plugins.	It is less secure and may get hacked easily as it uses many plugins and content sources.
It is more reliable, e.g., whenever the server is down, it is redirected to other nearby nodes.	It is less reliable, as it may go down for hours if the server breaks down.

Question-3: Key Differences Between Bandwidth and Frequency

1. Bandwidth measures the amount of data that a connection can transmit in a per unit time whereas, Frequency is a number of data packets arrived in per unit time.
2. Bandwidth is measured in bits/sec whereas, frequency is measured in hertz.

Question-4: Key Differences Between Broadcast and Multicast

1. The key difference between broadcast and multicast is that in the broadcast the packet is delivered to **all the**

Analog and Digital Electronics

UNIT 18

Chapter 1

Diode Circuits

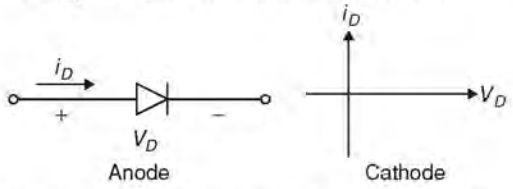
LEARNING OBJECTIVES

After reading this chapter, you will be able to understand:

- Small signal equivalent circuit of diode
- Practical PN junction diode
- Large signal diode-models
- Open circuit test
- Short circuit test
- Rectifiers
- Half wave rectifier analysis
- Full wave rectifier
- Peak inverse voltage
- Bridge rectifier
- Clipping circuits
- Clamper circuits
- Voltage multipliers
- Power supplies
- Voltage regulators

Small Signal Equivalent Circuit of Diode

An ideal diode is a two-element device which has the circuit symbol and Volt-ampere characteristic as shown below:



A diode is a two-terminal unipolar device which provides minimum resistance in the forward direction and maximum resistance in the reverse direction.

Ideal diode is an unilateral circuit element, as the current in the device is in one direction only. This behavior is important in switching as it provides an ON-OFF characteristic.

Practical PN Junction Diode

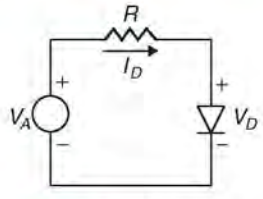


Figure 1 Diode circuit

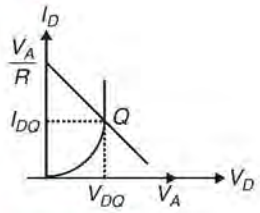


Figure 2 Diode characteristic and load line

From the circuit, $V_A = I_D R + V_D$ (KVL) $I_D = \frac{-1}{R} V_D + \frac{V_A}{R}$

This equation defines a straight line, called load line. The load line and diode characteristic must be satisfied simultaneously, at their point of intersection Q, called Quiescent or operating point. The values of current in and voltage across the diode are denoted by I_{DQ}, and V_{DQ} respectively.

Large Signal Diode-Models

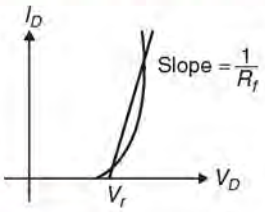


Figure 3 Piecewise linear diode forward characteristic

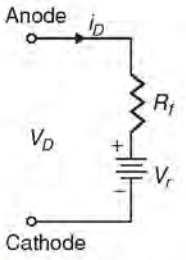


Figure 4 Diode model for forward bias

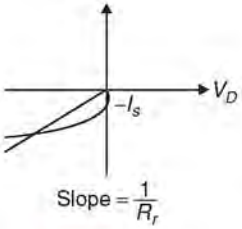


Figure 5 Piecewise linear reverse based characteristic diode

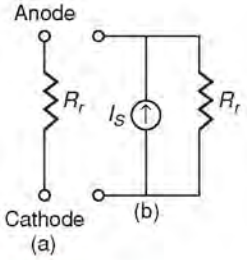


Figure 6 Diode model on piecewise of linear representation (b) Model to include surface leakage

Chapter 2

Bipolar Junction Transistors

LEARNING OBJECTIVES

After reading this chapter, you will be able to understand:

- Transistor construction
- Transistor symbols
- Transistor current Components
- Transistor configurations
- Common-base configuration
- Operation modes of transistor
- Common emitter configuration
- Common collector configuration
- Thermal run away
- Power rating of transistor

INTRODUCTION

When a third doped element is added to a diode in such a way that two pn junctions are formed, the resulting device is known as a transistor. Transistors are smaller than vacuum tubes. Invented in 1948 by J. Barden and W.H. Brattain of Bell Laboratories, USA.

Transistor Construction

A transistor consists of two-pn-junctions formed by sandwiching either p-type or n-type semi-conductor between a pair of opposite types. Accordingly there are two types of transistors,

Namely, (i) *n-p-n* transistor;

(ii) *p-n-p* transistor;

An *n-p-n* transistor is composed of two *n*-type semiconductor separated by a thin section of *p*-type.

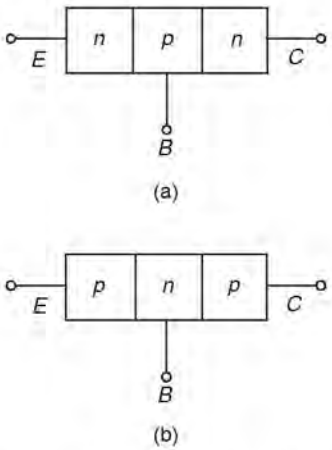


Figure 1 Types of transistors (a) *n-p-n*, and (b) *p-n-p*

- The emitter is heavily doped, the base is lightly doped, and the collector is moderately doped. According to area emitter is moderate, base is very thin and collector is large to dissipate heat.
- A transistor has two pn-junction, one junction is forward biased and the other junction is reverse biased. The forward junction has a low resistance path, whereas a reverse biased junction has a high resistance path. The weak signal is introduced at the low resistance circuit and output is taken from the high resistance circuit. Therefore, a transistor transfers a signal form low resistance to high resistance.

That is Transistor → Transfer + Resistor

A transistor has three sections of doped semiconductors. The section on one side is the emitter and the section on the opposite side is the collector. The middle section is called the base. It forms two junctions between the emitter and collector.

1. **Emitter:** The section on one side that supplies charge carriers (electrons or holes) is called the emitter. The emitter is always forward biased with refer to the base, so that it can supply a large number of majority carriers.
2. **Base:** The middle section which forms two pn-junctions between the emitter and collector is called the base. The base-emitter junction is forward biased andallowing low resistance for the emitter circuit. The base-collector junction is reverse biased. So it provides high resistance in the collector circuit.
3. **Collector:** The section on the other side that collects the charges is called the collector. The collector is always reverse biased.

Chapter 3

Field Effect Transistors

LEARNING OBJECTIVES

After reading this chapter, you will be able to understand:

- JFET
- Parameters of FET
- Relation between BJT and JFET
- Comparisons of JFET and BJT
- Introduction to MOSFET
- D–MOSFET
- E–MOSFET
- The body effect
- N–MOS transistor
- PMOS transistors
- Conditions for saturation region

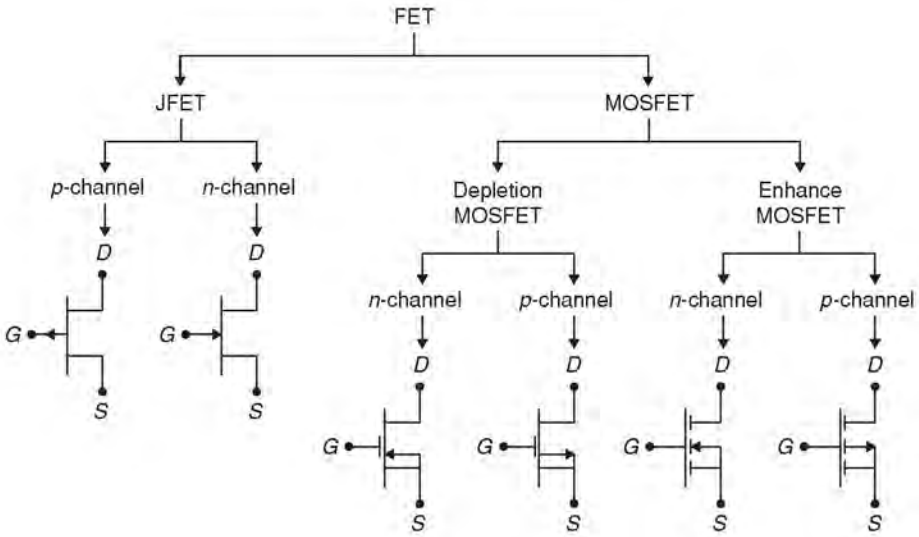
INTRODUCTION

The bipolar junction transistor relies on two types of charge carriers i.e., free electrons and holes. Another kind of transistor is called the field effect transistor (FET). This type of device is unipolar because its operation depends on only one type of charge, either free electrons or holes. In other words, an FET has majority carriers but not minority carriers.

The FET is generally much less noisy than the BJT. BJT is a current controlled device, and FET is a voltage controlled current device.

These are two types:

1. Junction field effect transistors (JFET).
2. Metal oxide semiconductor field effect transistor (MOSFET).



JFET

A junction field effect transistor (see Figure 1) is a three terminal semiconductor device in which current conduction is by majority types of carriers i.e., electrons or holes. JFET has high input impedan ce and bw noise level.

Chapter 5

Differential and Feedback Amplifiers

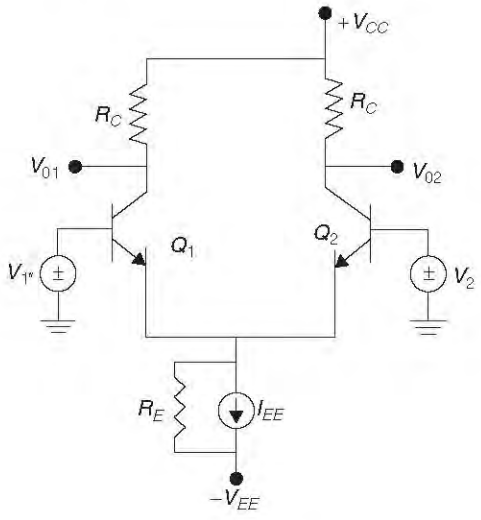
LEARNING OBJECTIVES

After reading this chapter, you will be able to understand:

- The differential amplifier
- FET differential amplifiers
- Feed back amplifiers
- Power amplifiers
- Amplifier efficiency
- Maximum theoretical efficiency
- Dissipation
- Class 'AB' amplifier
- Class 'C' amplifier
- Resonant frequency
- 555 timer
- Schmitt trigger
- Voltage controlled oscillator

THE DIFFERENTIAL AMPLIFIER

The emitter coupled differential amplifier is an essential building block in modern IC amplifiers.



Differential mode voltage gain $A_{DM} = -g_m R_C$

Common mode voltage gain $A_{CM} = \dfrac{-R_C}{2R_E}$

Common mode rejection ratio

$$\text{CMRR} = \frac{A_{DM}}{A_{CM}} = 1 + 2g_m R_E$$

Output for arbitrary input signals: If V_1 and V_2 are inputs applied to transistors Q_1 and Q_2

$$V_{DM} = \frac{V_1 - V_2}{2}, V_{CM} = \frac{V_1 + V_2}{2}$$

$$V_{01} = A_{DM} V_{DM} + A_{CM} V_{CM}$$

$$= A_{DM} = \left(V_{DM} + \frac{V_{CM}}{\text{CMRR}} \right)$$

$$V_{02} = -A_{DM} V_{DM} + A_{CM} V_{CM}$$

$$= -A_{DM} \left(V_{DM} - \frac{V_{CM}}{\text{CMRR}} \right)$$

Input and output resistances: Differential mode output resistance

$$R'_{O(DM)} = R_c \| r_o \cong R_C$$

Differential mode input resistance $R_{i(DM)} = 2r_e$

FET differential Amplifiers: The source-coupled pair differential amplifier with MOSFETS is shown in the figure

Chapter 6

Operational Amplifiers

LEARNING OBJECTIVES

After reading this chapter, you will be able to understand:

- Operational Amplifiers
- Single–ended operation
- CMRR (common mode rejection ratio)
- Level–shifter
- 2 electrical parameters of op-Am
- Open loop op-Amp configurations
- The non-inverting amplifier
- Closed loop op-Amp configurations
- Filters
- Oscillators
- Theory of sinusoidal oscillators
- Waveform generators and wave shaping circuits
- Square wave generation from sinusoid
- Monostable multivibrator
- Power supplies

OPERATIONAL AMPLIFIERS

An operational amplifier is a direct, coupled, high gain amplifier consisting of one or more differential amplifiers, usually followed by a level translator and an output stage. The output stage is generally a push-pull or push-pull complementary-symmetry pair

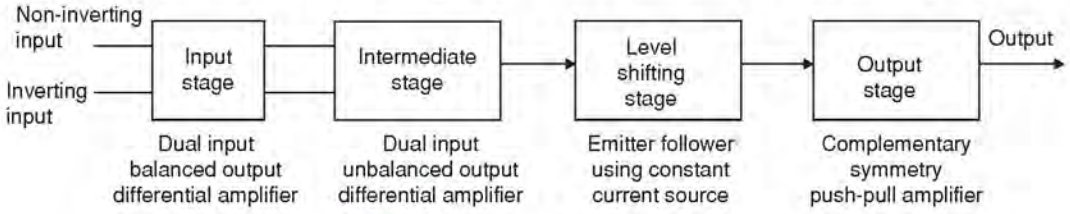
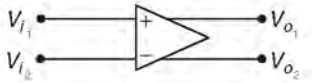


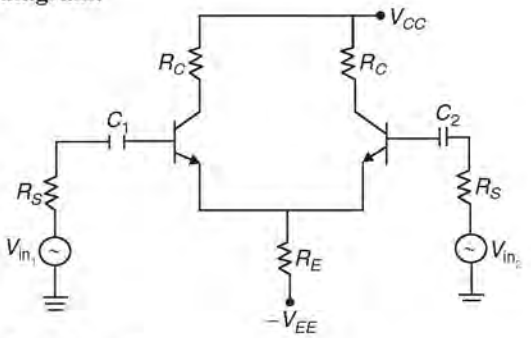
Figure 1 Block diagram of a typical op-amp

Differential Amplifier

Symbol:



Circuit diagram:



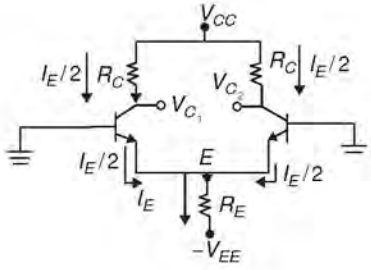
Modes of operation:

(i) Single-ended: Either of the input is grounded
(ii) Double-ended: Opposite polarity signals are applied at two inputs.
(iii) Common mode: Two similar input signals are applied at both inputs.

DC analysis:

(i) Make the AC sources ground
(ii) Replace the coupling capacitors open

Equivalent Circuit:



Electric Circuits and Fields

UNIT 19

Chapter 1

Network Elements and Basic Laws

LEARNING OBJECTIVES

After reading this chapter, you will be able to understand:

- Basic concepts
- Basic quantities
- Classification of network elements
- Independent sources
- Network terminology
- DC network
- AC network
- Kirchhoff's laws
- Circuit elements are connected in parallel
- Nodal analysis
- Mesh analysis

Basic Concepts

The most basic quantity used in the analysis of electrical circuits is the electric charge.

Basic Quantities

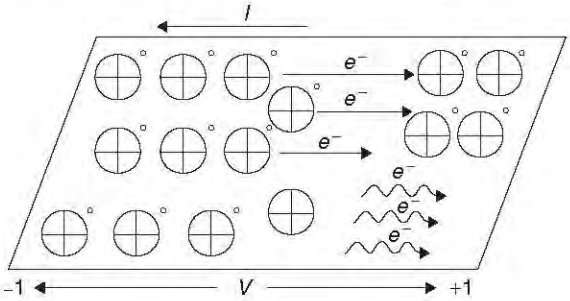
Electron

Electron is a mobile charge carrier. The electron (e^-) is measured in coulombs (C). $1\ e^- = 1.6 \times 10^{-19}$ C

- Multiples of electrons constitute charge (q)
- The movement of charge (q) over time causes current.

Current

There are free electrons available in all semi-conductive and conductive materials. These free e^-'s move at random in all directions with in the structure in the absence of external pressure or voltage. If a certain amount of voltage is applied across the material, all the free e^-'s move in one direction depending on the polarity of the applied voltage.



The voltage

According to the structure of an atom, there are two types of charges; positive and negative charge. A force of attraction exists between these charges. A certain amount of energy is required to overcome the force and move the charges through a specific distance. All opposite charges possess a certain amount of potential energy because of the separation between them. The difference in potential energy of the charges is called the potential difference.

The potential difference in electrical terminology is known as voltage and is denoted by V. It is expressed in terms of energy (W) per unit charges (Q)

$$\therefore \quad V = \frac{W}{Q} \text{ or } \upsilon = \frac{dW}{dQ}$$

The voltage is defined as the work (or) energy required to move a unit charge through an element.

The time rate of change of charge produces an electrical current

$$i(t) = \frac{dq(t)}{dt}$$

The electric current is measured in Ampere (A).

$$1\,\text{A} = \frac{1\,\text{C}}{1\,\text{sec}}$$

Power and energy

Energy is the capacity for doing work, i.e., energy is nothing but stored work.

Chapter 3

Transient Analysis (AC and DC)

LEARNING OBJECTIVES

After reading this chapter, you will be able to understand:

- Classification of transients
- Singularity functions
- Step response of an *RC* circuit
- Transient response
- Steady-state response
- Higher-order circuits
- AC transients
- Sinusoidal steady-state analysis
- Phasor
- Inductor
- Sinusoidal steady state analysis of *RLC* circuits
- AC transients
- Sinusoidal steady-state analysis
- Sinusoidal steady state analysis of *RLC* circuits

INTRODUCTION

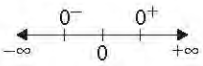
Whenever a circuit is switched from one condition to another, either by a change in the applied source or a change in the circuit elements, there is a transitional period during which the branch currents and element voltages change from their initial values to new ones. This period is called the transient period. After the transient period has passed, the circuit is said to be in steady state.

Transient in the system is because of the presence of energy storing elements (i.e., L and C).

Since the energy stored in a memory element cannot change instantaneously, i.e., within zero time.

The network consists of only resistances, no transients in the system at the time of switching. Since the resistor can accommodate any amount of voltage and currents.

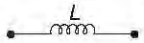
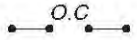
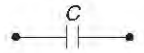
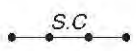
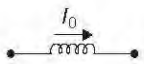
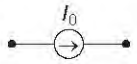
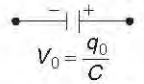
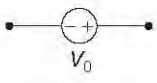
The equivalent form of the elements in terms of the initial condition of the elements.



1. $Z_L = sL\ \Omega$
2. $Z_C = \dfrac{1}{sC}\ \Omega$

At $t = 0^+ \Rightarrow f = \infty \Rightarrow Z_L = \infty \Rightarrow L \Rightarrow 0.C$

$Z_C = \dfrac{1}{2\pi fc} = 0\ \Omega \Rightarrow C \Rightarrow$ Short circuit

S. No	Element (Initial Condition)	Equivalent Circuit at $t = 0^+$
1.		
2.		
3.		
4.		
5.	 $V_0 = \dfrac{q_0}{C}$	 V_0

A long time after the switching action ($t \to \infty$) is the steady state. In *S.S*, the inductor behaviours is a short circuit and capacitor behaviours is an open circuit.

$t \to \infty \Rightarrow f = 0.$

$Z_L = sL\ \Omega \Rightarrow Z_L = 0\ \Omega \Rightarrow S.C$

$Z_C = \dfrac{1}{sC}\ \Omega \Rightarrow Z_C = \infty \Rightarrow$ Open circuit.

* The equivalent form of the elements in terms of the final condition of the element.

Electrical Machines

UNIT 20

Chapter 1

Transformers

LEARNING OBJECTIVES

After reading this chapter, you will be able to understand:

- Single-phase transformer
- Windings
- Bushings
- Breather
- Voltage transformation ratio
- Ideal two-winding transformer
- Phasor diagram of transformer under load
- Equivalent circuit
- Transformer tests
- Sumpner's test
- Losses and efficiency
- Voltage regulation of a transformer
- Auto-transformer
- Parallel operation of single-phase transformers

INTRODUCTION

Single-Phase Transformer

The transformer is a device that

1. Transfers electrical energy from one electrical circuit to another
2. Does so without a change in the frequency
3. Works on the principle of mutual induction
4. Has electric circuits that are linked by a common magnetic circuit

The energy transfer usually takes place with a change of voltage. If the energy transfer occurs at the same voltage, the purpose of transformer is merely to isolate the two electric circuits

- Transformer is a static device
- No rotating or moving parts
- Single-phase transformer has two circuits which are electrically isolated but magnetically coupled
- Transformer is an electromagnetic energy conversion device, since the energy received by the primary is first converted to magnetic energy and it is then reconverted to useful electrical energy in the secondary winding circuit
- Voltage and current levels change depending on transformation ratio
- Due to the absence of moving parts, transformers require very little maintenance and it has the highest possible efficiency out of all electrical machines
- Transformer is the main reason for the widespread popularity of AC systems over DC systems

Constructional Details

- Core: Sheet steel or silicon steel with 4% silicon is used
- The core is laminated and the laminations are insulated from each other by a coat of varnish or by an oxide layer on the surface to minimize eddy current losses
- The thickness of laminations varies from 0.35 mm for a frequency of 50 Hz to 0.5 mm for a frequency of 25 Hz
- The core provides a continuous magnetic path of low reluctance. The relative permeability of the core material is of the order of 1000
- Core stepping gives high space factor and also results in reduced length of mean turn and the consequent I^2R loss

Types of Transformers

1. **Core type:** In this type of transformers, the windings surround a considerable part of the core
2. **Shell type:** In this type of transformers, the core surrounds a considerable portion of the windings
3. **Spiral-core or wound-core (trade name—Spirakore):** In this type of transformers, the core is assembled of a continuous strip or ribbon of transformer steel wound in the form of a circular or elliptical cylinder

Chapter 3

DC Machines

LEARNING OBJECTIVES

After reading this chapter, you will be able to understand:

- Armature windings
- Types of generators
- E.M.F. equation of a DC generator
- Performance equations of DC generators
- Losses in a DC generator
- Armature reaction
- Demagnetizing ampere-turns per pole
- Compensating windings
- Commutation
- Equalizing connections
- Characteristics of DC generator
- Parallel operation of DC generators
- DC motors
- Motor characteristics

Fundamentals of DC Machines

Energy can neither be created nor be destroyed. We can change its forms, using appropriate energy -conversion processes. Change of the form of energy requires an energy conversion device.

- DC generator and DC motor are electromechanical energy conversion devices,
- A DC generator converts mechanical energy into electrical energy. It requires a prime mover such as a turbine diesel engine, etc.
- In a DC generator, the energy conversion is based on the principle of the production of dynamically induced e.m.f. This process can be explained by Faraday's laws of electromagnetic induction.
- The two basic essential parts of an electrical generator are
  1. A magnetic field, and
  2. A conductor or conductors which can so move as to cut the flux.
- Whenever there is relative motion between the conductor and the magnetic field, an emf is induced in the conductors which causes a current to flow if the conductor circuit is closed.
- In a DC generator, relative motion is produced by rotation of the armature. The armature winding is on the rotor and field winding is on the stator.
- In normal DC machines stator core is not laminated, armature core is laminated to reduce eddy current losses.

Constructional Details

The major parts of a DC machine are

1. Field system
2. Armature
3. Commutator
4. Brush and brush gear.

Constructionally there is no difference between a DC generator and DC motor.

1. The Field System

The purpose of the field system is to provide a uniform magnetic field within which armature rotates. The magnetic field is generated by electromagnets rather than permanent magnets on account of their greater magnetic strength and field strength regulation (i.e., field flux can be varied according to the requirement). The field system consists of four parts:

1. **Yoke:** This provides mechanical support for the poles and acts as a protecting cover for the whole machine and it carries the magnetic flux produced by the pole.

Chapter 4

Induction Motors

LEARNING OBJECTIVES

After reading this chapter, you will be able to understand:

- Principle of operation
- Rotor current and power factor
- Power stages in an induction motor
- Synchronous watt
- Rotor torque and breakdown torque
- Equivalent circuit of the rotor
- Starting of induction motors
- Double squirrel cage motor
- Speed control of induction motor
- Revolving field theory of single phase induction motor
- Capacitor split phase motors
- Shaded pole single-phase motor
- Repulsion start induction motor

INTRODUCTION

Three-phase induction motors (IMs) are the widely used AC motors due to their low cost, simple and rugged construction, high reliability, high efficiency. IMs have good speed regulation, less maintenance, simple starting arrangement, reasonable overload capacity. Conversion of electrical power into mechanical power takes place in the rotating part of an electric motor. In DC motors, the electric power is conducted directly to the armature (i.e., rotating part) through brushes and commutator. Hence in this sense, a DC motor can be called a conduction motor. However, in AC motors, the rotor does not receive electric power by conduction but by induction. This is why such motors are known as IMs. An IM can be treated as a rotating transformer. Unlike a transformer due to the presence of air gap, IM has more magnetizing currents, almost lagging 90° the applied voltage and therefore p.f. is low: The no-load current of IM is about 30% to 50% of full-load current, whereas in transformer no-load current is only 2% to 6% of full-load current.

Constructional Details

The IMs essentially consists of two parts:

(a) A stationary part called stator
(b) The revolving part, called rotor

Stator

The stator of an IM is made up of a number of stampings, which are slotted on the inner periphery to receive the windings. The stator carries a 3-phase winding which may be star or delta connected and is fed from a 3-phase supply. It is wound for a definite number of poles, the exact number of poles being determined by the requirements of speed.

The number poles P, produced in the rotating field is $P = 2n$, where n is the number of stator slots/pole/phase.

Rotor

Rotor is made up of thin laminations of the same material as stator. These laminations are slotted on the outer periphery. There are two types of rotors.

1. Squirrel-cage rotor: Motors employing this type of rotor are known as squirrel-cage induction motors (SCIMs)
2. Phase wound (or) wound rotor: The motors employing this type of rotor are variously known as 'phase-wound' motors or 'wound' motors or 'slip-ring' motors

Slip-ring Rotor

- Has slotted armature
- Provided with 3-phase double-layer distributed winding housed in the rotor slots

SQL Query WorkBook

Unit 4

```
    return revNum;
}

int main()
{
    unsigned int num = 0x4;
    printf("\n%u", revBits(num));
    return 0;
}
```

```
8912
```

Program-57: Counting number of 1's using C program

```
#include <stdio.h>
int count1s(unsigned int num)
{
    unsigned char i;
    int count=0;
    unsigned char totalBits=sizeof(num)*8;
    for(i=0;i< totalBits;i++)
    {
        if( num & (1<< i) )
            count++;
    }
    return count;
}
int main()
{
    unsigned int data=0x58;
    printf("\nTotal number of 1's are :
%d\n",count1s(data));
    return 0;
}
```

Output

```
Total number of 1's are : 3
```

Program-60: Swap two numbers using Bitwise XOR Operator in C

```
#include <stdio.h>
void swap(int* a, int* b); //function declaration

int main()
{
    int a, b;

    printf("Enter first number: ");
    scanf("%d", &a);
    printf("Enter second number: ");
    scanf("%d", &b);

    printf("Before swapping: a=%d, b=%d\n", a,
b);
    swap(&a, &b);
    printf("After swapping:  a=%d, b=%d\n", a,
b);

    return 0;
}

//function definition
void swap(int* a, int* b)
{
    *a = *a ^ *b;
    *b = *a ^ *b;
    *a = *a ^ *b;
}
```

Output:

```
Enter first number: 10
Enter second number: 20
Before swapping: a=10, b=20
After swapping:  a=20, b=10
```

Program-61: Number of Trailing Zero in Binar Reprentation

```
#include <stdio.h>

int main()
{
    unsigned int n;
    printf("enter the integer\n");
    scanf("%d",&n);
    int count=0;
    while(n!=0){
            if(n & 1 == 1) //if current bit is 1
            break; //no more trailing zero
            n=n>>1; //right shift
            count++; //if trailing zero, increase
count
    }

    printf("no of trailing zero ");
    //print no of trailing zero
printf("in its binary representation: %d
\n",count);

    return 0;
}
```

Output (first run)

```
enter the integer
12
no of trailing zero in its binary representation: 2
```

Program-62: C program to find the Highest Bit Set for any given Integer

```
#include <stdio.h>

int main()
{
    unsigned int n;
    printf("enter the integer\n");
    scanf("%d",&n);
    int count=0,store=-1;
    while(n!=0){
    if(n & 1 == 1) //if current bit is set
            store=count; //update store
            n=n>>1; //right shift
            count++; //increase count
    }
```

```
    if(store==-1){//if store not updated
  printf("No bit is set\n"); //no set bit present at all
            return 0;
    }
    printf("Highest bit set ");
    //printing highest set bit
  printf("in its binary representation: %d \n",store);

    return 0;
  }
```

Output

```
enter the integer
12
Highest bit set in its binary representation: 3
```

Program-63: C program to check if all the bits of a given integer is one (1)

C implementation

```
#include <stdio.h>

int main()
{
    unsigned int n;
    printf("enter the integer\n");
    scanf("%d",&n);

    while(n>0){
            int temp=n&1;
            if(temp == 0){ //if any bit not set
            printf("all bits are not set\n");
                    return 0;
            }
            n=n>>1; //right shift operator
    }

    printf("all bits are set ");
    printf("in its binary representation\n");

    return 0;
}
```

Output

```
First run:
enter the integer
12
all bits are not set
```

Program-64: C program to count number of bits set to 1 in an Integer

```
#include <stdio.h>

int main()
{
    unsigned int n;
    printf("enter the integer\n");
    scanf("%d",&n);

    int count=0;

    while(n!=0){
    if(n & 1 == 1){ //if current bit 1
            count++;//increase count
            }
            n=n>>1;//right shift
    }

    printf("no of bits those are 1 ");
  printf("in its binary representation: %d\n",count);

    return 0;
}
```

Output

```
First run:
enter the integer
7
no of bits those are 1 in its binary representation: 3
```

Program-65: C program to check whether a given number is palindrome or not using Bitwise Operator

C Implementation

```
#include <stdio.h>
#define SIZE 8

int main()
{
    unsigned int n;
    printf("enter the no ( max range 255)\n");
    scanf("%d",&n);
    int c[SIZE]={0};
    int i=SIZE-1;
    printf("binary representation is: ");
    while(n!=0){
      c[i--]=n&1;
      n=n>>1;

    }
    for(int j=0;j<SIZE;j++)
    printf("%d",c[j]);
    printf("\n");

    for(int j=0,k=SIZE-1;j<k;j++,k--){
      if(c[j]!=c[k]){
        printf("Not palindrome\n");
        return 0;
      }
    }

    printf("it's palindrome\n");

    return 0;
}
```

Output

```
First run:
enter the no ( max range 255)
153
binary representation is: 10011001
it's palindrome
```

```
int sumdig(int n)
{
  int s, d;
  if(n!=0)
  {
    d = n%10;
    n = n/10;
    s = d+sumdig(n);
  }
  else
    return 0;
  return s;
}
```

A. 4, 4
B. 3, 3
C. 6, 6
D. 12, 12
Answer: Option **C**

42. What will be the output of the program?

```
#include<stdio.h>

int main()
{
  void fun(char*);
  char a[100];
  a[0] = 'A'; a[1] = 'B';
  a[2] = 'C'; a[3] = 'D';
  fun(&a[0]);
  return 0;
}
void fun(char *a)
{
  a++;
  printf("%c", *a);
  a++;
  printf("%c", *a);
}
```

A. AB
B. BC
C. CD
D. No output
Answer: Option **B**

43. What will be the output of the program?

```
#include<stdio.h>

int main()
{
  int fun(int);
  int i = fun(10);
  printf("%d\n", --i);
  return 0;
}
int fun(int i)
{
  return (i++);
}
```

A. 9
B. 10
C. 11
D. 8
Answer: Option **A**
Explanation:
Step 1: int fun(int); Here we declare the prototype of the function fun().
Step 2: int i = fun(10); The variable i is declared as an integer type and the result of the fun(10) will be stored in the variable i.
Step 3: int fun(int i){ return (i++); } Inside the fun() we are returning a value return(i++). It returns 10. because i++ is the post-increement operator.
Step 4: Then the control back to the main function and the value 10 is assigned to variable i.
Step 5: printf("%d\n", --i); Here --i denoted pre-increement. Hence it prints the value 9.

44. What will be the output of the program?

```
#include<stdio.h>
int check (int, int);

int main()
{
  int c;
  c = check(10, 20);
  printf("c=%d\n", c);
  return 0;
}
int check(int i, int j)
{
  int *p, *q;
  p=&i;
  q=&j;
  i>=45 ? return(*p): return(*q);
}
```

A. Print 10
B. Print 20
C. Print 1
D. Compile error
Answer: Option **D**
Explanation:
There is an error in this line i>=45 ? return(*p): return(*q);. We cannot use return keyword in the terenary operators.

45. What will be the output of the program?

```
#include<stdio.h>
int fun(int, int);
typedef int (*pf) (int, int);
int proc(pf, int, int);

int main()
{
  printf("%d\n", proc(fun, 6, 6));
  return 0;
}
int fun(int a, int b)
{
  return (a==b);
}
int proc(pf p, int a, int b)
{
  return ((*p)(a, b));
}
```

A. 6
B. 1
C. 0
D. -1
Answer: Option **B**
Explanation:
No answer description available for this question.

46. What will be the output of the program?

```
#include<stdio.h>

int main()
```

```
{
  int i=1;
  if(!i)
    printf("IndiaBIX,");
  else
  {
    i=0;
    printf("C-Program");
    main();
  }
  return 0;
}
```

A. prints "IndiaBIX, C-Program" infinitely
B. prints "C-Program" infinetly
C. prints "C-Program, IndiaBIX" infinitely
D. Error: main() should not inside else statement

Answer: Option **B**

Explanation:

Step 1: int i=1; The variable i is declared as an integer type and initialized to 1(one).
Step 2: if(!i) Here the !(NOT) operator reverts the i value 1 to 0. Hence the if(0) condition fails. So it goes to else part.
Step 3: else { i=0; In the else part variable i is assigned to value 0(zero).
Step 4: printf("C-Program"); It prints the "C-program".
Step 5: main(); Here we are calling the main() function. After calling the function, the program repeats from **step 1** to **step 5** infinitely.

Hence it prints "C-Program" infinitely.

47. What will be the output of the program?

```
#include<stdio.h>

int addmult(int ii, int jj)
{
  int kk, ll;
  kk = ii + jj;
  ll = ii * jj;
  return (kk, ll);
}

int main()
{
  int i=3, j=4, k, l;
  k = addmult(i, j);
  l = addmult(i, j);
  printf("%d %d\n", k, l);
  return 0;
}
```

A. 12 12
B. No error, No output
C. Error: Compile error
D. None of above

Answer: Option **A**

48. What will be the output of the program?

```
#include<stdio.h>
int i;
int fun1(int);
int fun2(int);

int main()
{
  extern int j;
  int i=3;
  fun1(i);
  printf("%d,", i);
  fun2(i);
  printf("%d", i);
  return 0;
}
int fun1(int j)
{
  printf("%d,", ++j);
  return 0;
}
int fun2(int i)
{
  printf("%d,", ++i);
  return 0;
}
int j=1;
```

A. 3, 4, 4, 3
B. 4, 3, 4, 3
C. 3, 3, 4, 4
D. 3, 4, 3, 4

Answer: Option **B**

Explanation:

Step 1: int i; The variable i is declared as an global and integer type.
Step 2: int fun1(int); This prototype tells the compiler that the fun1() accepts the one integer parameter and returns the integer value.
Step 3: int fun2(int); This prototype tells the compiler that the fun2() accepts the one integer parameter and returns the integer value.
Step 4: extern int j; Inside the main function, the extern variable j is declared and defined in another source file.
Step 5: int i=3; The local variable i is defines as an integer type and initialized to 3.
Step 6: fun1(i); The fun1(i) increements the given value of variable i prints it. Here fun1(i) becomes fun1(3) hence it prints '4' then the control is given back to the main function.
Step 7: printf("%d,", i); It prints the value of local variable i. So, it prints '3'.
Step 8: fun2(i); The fun2(i) increements the given value of variable i prints it. Here fun2(i) becomes fun2(3) hence it prints '4' then the control is given back to the main function.
Step 9: printf("%d,", i); It prints the value of local variable i. So, it prints '3'.

Hence the output is "4 3 4 3".

49. What will be the output of the program?

```
#include<stdio.h>
int func1(int);

int main()
{
  int k=35;
  k = func1(k=func1(k=func1(k)));
  printf("k=%d\n", k);
  return 0;
}
int func1(int k)
{
  k++;
  return k;
}
```

A. k=35
B. k=36
C. k=37
D. k=38

Answer: Option **D**

Explanation:

Step 1: int k=35; The variable k is declared as an integer type and initialized to 35.
Step 2: k = func1(k=func1(k=func1(k))); The func1(k) increement the value of k by 1 and return it. Here the func1(k) is called 3 times. Hence it increements value of k = 35 to 38. The result is stored in the variable k = 38.
Step 3: printf("k=%d\n", k); It prints the value of variable k "38".

50. What will be the output of the program?

```
#include<stdio.h>

int addmult(int ii, int jj)
{
   int kk, ll;
   kk = ii + jj;
   ll = ii * jj;
   return (kk, ll);
}

int main()
{
   int i=3, j=4, k, l;
   k = addmult(i, j);
   l = addmult(i, j);
   printf("%d, %d\n", k, l);
   return 0;
}
```

A. 12, 12
B. 7, 7
C. 7, 12
D. 12, 7

Answer: Option **A**
Explanation:
Step 1: int i=3, j=4, k, l; The variables i, j, k, l are declared as an integer type and variable i, j are initialized to 3, 4 respectively.

The function addmult(i, j); accept 2 integer parameters.

Step 2: k = addmult(i, j); becomes k = addmult(3, 4)
In the function addmult(). The variable kk, ll are declared as an integer type int kk, ll;
kk = ii + jj; becomes kk = 3 + 4 Now the kk value is '7'.
ll = ii * jj; becomes ll = 3 * 4 Now the ll value is '12'.
return (kk, ll); It returns the value of variable ll only.
The value 12 is stored in variable 'k'.
Step 3: l = addmult(i, j); becomes l = addmult(3, 4)
kk = ii + jj; becomes kk = 3 + 4 Now the kk value is '7'.
ll = ii * jj; becomes ll = 3 * 4 Now the ll value is '12'.
return (kk, ll); It returns the value of variable ll only.
The value 12 is stored in variable 'l'.
Step 4: printf("%d, %d\n", k, l); It prints the value of k and l

Hence the output is "12, 12".

51. What will be the output of the program?

```
#include<stdio.h>
int check(int);
int main()
{
   int i=45, c;
   c = check(i);
   printf("%d\n", c);
   return 0;
}
int check(int ch)
{
   if(ch >= 45)
      return 100;
   else
      return 10;
}
```

A. 100
B. 10
C. 1
D. 0

Answer: Option **A**
Explanation:
Step 1: int check(int); This prototype tells the compiler that the function check() accepts one integer parameter and returns an integer value.
Step 2: int l=45, c; The variable i and c are declared as an integer type and i is initialized to 45.
The function check(i) return 100 if the given value of variable i is >=(greater than or equal to) 45, else it will return 10.
Step 3: c = check(i); becomes c = check(45); The function check() return 100 and it get stored in the variable c.(c = 100)
Step 4: printf("%d\n", c); It prints the value of variable c.

Hence the output of the program is '100'.

52. If int is 2 bytes wide.What will be the output of the program?

```
#include <stdio.h>
void fun(char**);

int main()
{
   char *argv[] = {"ab", "cd", "ef", "gh"};
   fun(argv);
   return 0;
}
void fun(char **p)
{
   char *t;
   t = (p+= sizeof(int))[-1];
   printf("%s\n", t);
}
```

A. ab
B. cd
C. ef
D. gh

Answer: Option **B**
Explanation:
Since C is a machine dependent language sizeof(int) may return different values.
The output for the above program will be cd in Windows (Turbo C) and gh in Linux (GCC).

To understand it better, compile and execute the above program in Windows (with Turbo C compiler) and in Linux (GCC compiler).

53. What will be the output of the program?

```
#include<stdio.h>
int fun(int(*)());

int main()
{
   fun(main);
   printf("Hi\n");
   return 0;
}
```

Analysis

Simple Statement
This statement takes **O(1)** time.
int y= n + 25;
If Statement
The worst case O(n) if the if statement is in a loop that runs n times, best case **O(1)**

```
if( n> 100)
{
...
}else{
..
..
}
```

For / While Loops
The **for** loop takes n time to complete and and so it is **O(n)**.

```
for(int i=0;i<n;i++)
{
..
..
}
```

The **while** loop takes n time as well to complete and so it is **O(n)**.

```
int i=0;
while( i<n)
{
..
i++;
}
```

If the **for** loop takes n time and i increases or decreases by a constant, the cost is **O(n)**

```
for(int i = 0; i < n; i+=5)
  sum++;

for(int i = n; i > 0; i-=5)
  sum++;
```

If the **for** loop takes n time and i increases or decreases by a multiple, the cost is **O(log(n))**

```
for(int i = 1; i < =n; i*=2)
  sum++;

for(int i = n; i > 0; i/=2)
  sum++;
```

Nested loops
If the nested loops contain sizes n and m, the cost is **O(nm)**

```
for(int i=0;i<n;i++)
{
   for(int i=0;i<m;i++){
   ..
   ..
   }
}
```

If the first loop runs N times and the inner loop runs log(n) times or (vice versa), the cost is **O(n*log(n))**

```
for(int i=0;i<n;i++)
{
   for(int j=1;i<=n;j*=4){
   ..
   ..
   }
}
```

If the first loop runs n^2 times and the inner loop runs n times or (vice versa), the cost is **O(n^3)**

```
for(int j=0;j<n*n;j++)
{
   for(int i=0;i<n;i++){
   ..
   ..
   }
}
```

If the first loop runs n times and the inner second loop runs n^2 times and the third loop runs n^2, then **O(n^5)**

```
for(int i = 0; i < n; i++)
   for( int j = 0; j < n * n; j++)
      for(int k = 0; k < j; k++)
         sum++;
```

Analyze the following sorting algorithm:

```
for (int i = 0; i < list.length - 1; i++) {
  if (list[i] > list[i + 1]) {
    swap list[i] with list[i + 1];
    i = -1;
  }
}
```

This is similar to bubble sort. Whenever a swap is made, it goes back to the beginning of the loop. In the worst case, there will be O(n^2) of swaps. For each swap, O(n) number of comparisons may be made in the worst case. So, the total is O(n^3) in the worst case.

Use the Big O notation to estimate the time complexity of the following methods:
(a)

```
public static void mA(int n) {
  for (int i = 0; i < n; i++) {
    System.out.print(Math.random());
  }
}
```

(b)

```
public static void mB(int n) {
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++)
      System.out.print(Math.random());
  }
}
```

(c)
```
public static void mC(int[] m) {
  for (int i = 0; i < m.length; i++) {
    System.out.print(m[i]);
  }

  for (int i = m.length - 1; i >= 0; )
  {
    System.out.print(m[i]);
    i--;
  }
}
```

(d)
```
public static void mD(int[] m) {
  for (int i = 0; i < m.length; i++) {
    for (int j = 0; j < i; j++)
      System.out.print(m[i] * m[j]);
  }
}
```
(a): O(n)
(b): O(n^2)
(c): O(n)
(d): O(n^2)

How many stars are displayed in the following code if n is 10? How many if n is 20? Use the Big O notation to estimate the time complexity.
(a)
```
for (int i = 0; i < n; i++) {
  System.out.print('*');
}
```

(b)
```
for (int i = 0; i < n; i++) {
  for (int j = 0; j < n; j++) {
    System.out.print('*');
  }
}
```

(c)
```
for (int k = 0; k < n; k++) {
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      System.out.print('*');
    }
  }
}
```

(d)
```
for (int k = 0; k < 10; k++) {
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      System.out.print('*');
    }
  }
}
```
if n is 10: (a) 10 (b) 10^2 (c) 10^3 (d) 10*10^2
if n is 20: (a) 20 (b) 20^2 (c) 20^3 (d) 20*20^2
Using Big-O notation: O(n), O(n^2), O(n^3), O(n^2)

Count the number of iterations in the following loops.
(a)
```
int count = 1;
while (count < 30) {
  count = count * 2;
}
```

(b)
```
int count = 15;
while (count < 30) {
  count = count * 3;
}
```

(c)
```
int count = 1;
while (count < n) {
  count = count * 2;
}
```

(d)
```
int count = 15;
while (count < n) {
  count = count * 3;
}
```
(A) 5
(B) 1
(C) The ceiling of log2n times
(D) The ceiling of log3(n/15) times

Question 1

```
total = 0;
for (int i = 0; i < n; i++)
        total += i;
```
1. O(1) 3. O(n)
2. O(log n) 4. O(n log n)

Answer: 3

Question 2
```
for (int i = 0; i < n; i++)    for (int j = 0; j < n; j++)
count++;
```
1. O(1) 3. O(n2)
2. O(n) 4. O(n log n)

Answer: 3

Question 3
```
for (int i = 0; i < n; i+=2)    for (int j = 0; j <n; j++)
sum = i + j;
```
1. O(n) 3. O(n3)
2. O(n2) 4. O(n log n)

Answer: 2

Question 4
```
for (int i = 0; i < 6n; i+=2)        sum++;
```
1. O(n) 3. O(3n) 2. O(2n) 4. O(6n)

Answer: 1

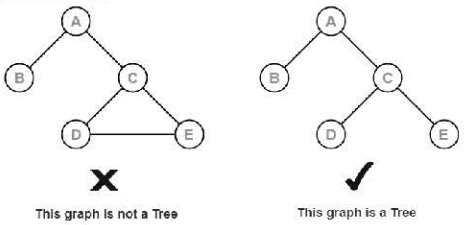
Question 5

Tree
Data Structure
WorkBook

Unit 2

Final Exercise on Tree

Example-



Properties-

The important properties of tree data structure are-

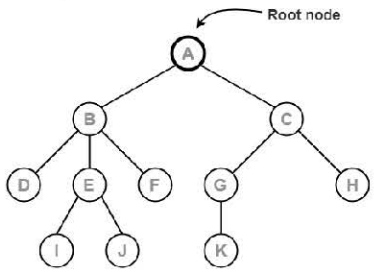
- There is one and only one path between every pair of vertices in a tree.
- A tree with n vertices has exactly (n-1) edges.
- A graph is a tree if and only if it is minimally connected.
- Any connected graph with n vertices and (n-1) edges is a tree.

Tree Terminology-

1. Root-

- The first node from where the tree originates is called as a **root node**.
- In any tree, there must be only one root node.
- We can never have multiple root nodes in a tree data structure.

Example-

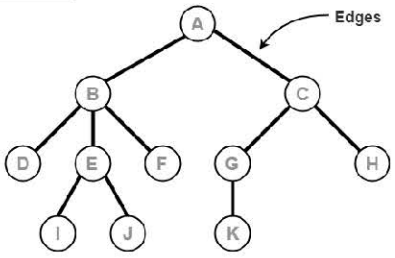


Here, node A is the only root node.

2. Edge-

- The connecting link between any two nodes is called as an **edge**.
- In a tree with n number of nodes, there are exactly (n-1) number of edges.

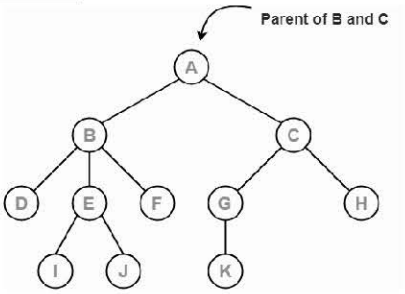
Example-



3. Parent-

- The node which has a branch from it to any other node is called as a **parent node**.
- In other words, the node which has one or more children is called as a parent node.
- In a tree, a parent node can have any number of child nodes.

Example-



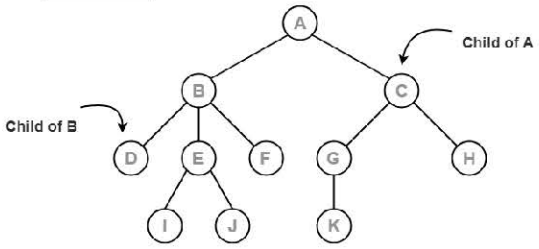
Here,

- Node A is the parent of nodes B and C
- Node B is the parent of nodes D, E and F
- Node C is the parent of nodes G and H
- Node E is the parent of nodes I and J
- Node G is the parent of node K

4. Child-

- The node which is a descendant of some node is called as a **child node**.
- All the nodes except root node are child nodes.

Example-



Here,

- Nodes B and C are the children of node A
- Nodes D, E and F are the children of node B
- Nodes G and H are the children of node C
- Nodes I and J are the children of node E
- Node K is the child of node G

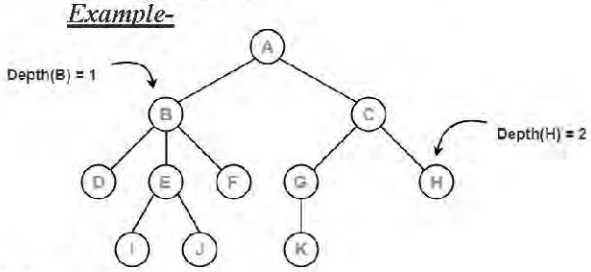
5. Siblings-

- Nodes which belong to the same parent are called as **siblings**.
- In other words, nodes with the same parent are sibling nodes.

Example-

- Total number of edges from root node to a particular node is called as **depth of that node**.
- **Depth of a tree** is the total number of edges from root node to a leaf node in the longest path.
- Depth of the root node = 0
- The terms "level" and "depth" are used interchangeably.

Example-



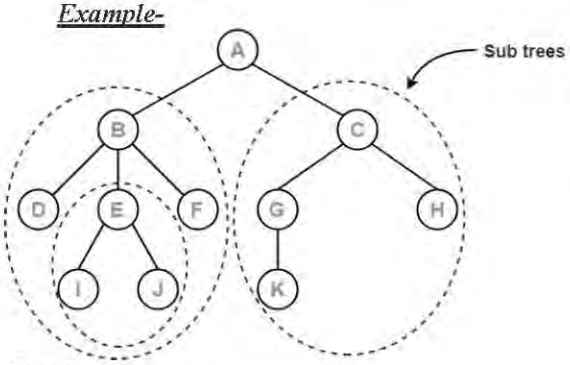
Here,

- Depth of node A = 0
- Depth of node B = 1
- Depth of node C = 1
- Depth of node D = 2
- Depth of node E = 2
- Depth of node F = 2
- Depth of node G = 2
- Depth of node H = 2
- Depth of node I = 3
- Depth of node J = 3
- Depth of node K = 3

12. Subtree-

- In a tree, each child from a node forms a **subtree** recursively.
- Every child node forms a subtree on its parent node.

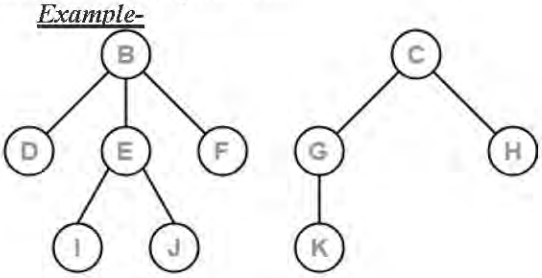
Example-



13. Forest-

A forest is a set of disjoint trees.

Example-



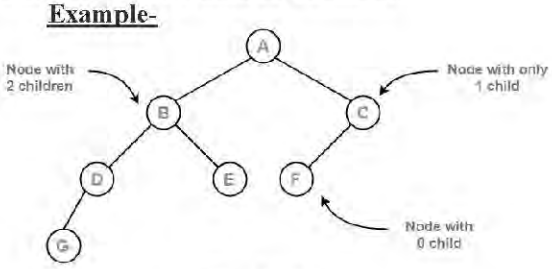
Forest

To gain better understanding about Tree Terminology,

Binary Tree-

Binary tree is a special tree data structure in which each node can have at most 2 children. Thus, in a binary tree, Each node has either 0 child or 1 child or 2 children.

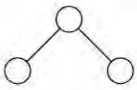
Example-



Binary Tree Example

Unlabeled Binary Tree-

A binary tree is unlabeled if its nodes are not assigned any label.



Unlabeled Binary Tree

Number of different Binary Trees possible with 'n' unlabeled nodes $= \frac{^{2n}C_n}{n+1}$

Example-

Consider we want to draw all the binary trees possible with 3 unlabeled nodes.

Using the above formula, we have-

Number of binary trees possible with 3 unlabeled nodes

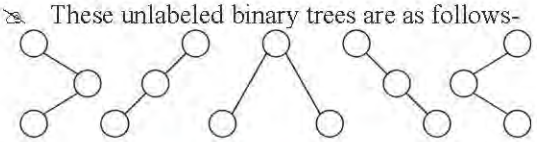
$= {}^{2 \times 3}C_3 / (3 + 1)$

$= {}^{6}C_3 / 4$

$= 5$

Thus,

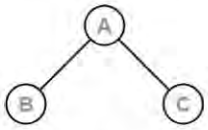
- With 3 unlabeled nodes, 5 unlabeled binary trees are possible.
- These unlabeled binary trees are as follows-



Binary Trees Possible With 3 Unlabeled Nodes

Labeled Binary Tree-

A binary tree is labeled if all its nodes are assigned a label.



Labeled Binary Tree

Number of different Binary Trees possible with 'n' labeled nodes $= \frac{^{2n}C_n}{n+1} \times n!$

Example-

Consider we want to draw all the binary trees possible with 3 labeled nodes.

Using the above formula, we have-

Number of binary trees possible with 3 labeled nodes

$= \{ {}^{2 \times 3}C_3 / (3 + 1) \} \times 3!$

$= \{ {}^{6}C_3 / 4 \} \times 6$

$= 5 \times 6$

$= 30$

Thus,

7. Assume a TDMA Communication system having 8 transmitter-Receiver pairs. Each is sampled at 8khz that generated 16 bit per sample. If two synchronous bits are used in each frame. Calculate the data rate of the TDMA Link.

Solution:

Data Rate = (nM+a)fs
= (16X8+2)X8
=1040kbps

Section 5.9 :

Ex. 5.9.4 : A PCM system uses a uniform quantizer followed by a 7 bit encoder. The system bit rate is 50 Mbits/sec. Calculate the maximum bandwidth of the message signal for which this system operates satisfactorily.

Soln. :

It has been given that : Bit rate r = 50 Mbits/sec and N = 7

We know that bit rate $r = N f_s$

$\therefore \quad f_s = \quad = \quad = 7.14$ MHz

$\therefore$ Maximum signal bandwidth BW $= f_s / 2 =$

$\therefore \quad BW = 3.57$ MHz **...Ans.**

Ex. 5.9.5 : The bandwidth of a video signal is 4.5 MHz. This signal is to be transmitted using PCM with the number of quantization levels Q = 1024. The sampling rate should be 20% higher than the Nyquist rate. Calculate the system bit rate.

Soln. :

Bandwidth $W = 4.5$ MHz

$\therefore$ As per Nyquist rate $f_s = 2W = 9$ MHz

But f_s should be 20% higher than Nyquist rate

$\therefore \quad f_s = 1.2 \times 9 \text{ MHz} = 10.8 \text{ MHz}$ ...(1)

We know that, $Q = 2^N, \quad \therefore \quad 1024 = 2^N$

$\therefore \quad N = 10$ ...(2)

$\therefore$ System bit rate $r = N f_s = 10 \times 10.8$ MHz

Ex. 5.9.7 : Plot the characteristics of a μ-law compressor.

Soln. :

To plot the characteristics of μ-law compressor :

The expression for the normalized output of a μ-law compressor is given by,

$$Z(x) = \pm \quad \ldots\ldots x \leq 1$$

Let $\mu = 255$.

$\therefore \quad \delta_{min} = = = 0.3436$ Volt **...Ans.**

2. Granular noise power :

$$N_q = \times = \times = 2.15 \times 10^{-3} \text{ W}$$ **...Ans.**

3. Signal power S_o and SNR_o :

As the signal is sinusoidal, the normalized output signal power

$$S_o = [A/]^2 = A/2 = 1/2 \text{ Watt.}$$ **...Ans.**

$$\therefore \quad SNR_o = = = 232.3 \text{ or } 23.66 \text{ dB.}$$

4. Signal power for uniformly distributed signal :

The signal PDF for a uniformly distributed signal is as shown in Fig. P. 5.15.9.

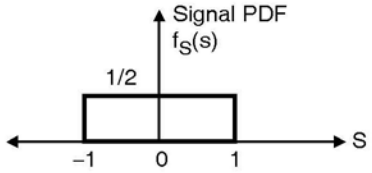


Fig. P. 5.15.9

$\therefore$ Mean square value of the signal, $E[S^2] = S^2 \cdot f_s(S)\, dS = S^2\, dS = = 1/3$

Assuming R = 1.

Normalized signal power S_o = Mean square value = 1/3 W **...Ans.**

Signal to noise ratio = SNR = = 155.03 or 21.9 dB **...Ans.**

Ex. 5.15.10 : The information in an analog signal voltage waveform is to be transmitted over a PCM system with an accuracy of ± 0.1 % full scale accuracy. The analog voltage waveform has a bandwidth of 100 Hz and an amplitude range of – 10 to + 10 Volts.

(a) Determine the minimum sampling rate required.

(b) Determine the number of bits in each PCM word.

(c) Determine the minimum bit rate required in the PCM system.

(d) Determine the minimum absolute channel bandwidth required for the transmission of the PCM signal.

Soln. :

It has been given that,

1. Accuracy of ± 0.1 % of full scale is expected.
2. W = 100 Hz and amplitude range is – 10 to + 10 V

(a) Sampling rate f_s :

By sampling theorem the minimum sampling rate is

$$f_{s\,(min)} = 2W = 200 \text{ Hz}$$ **...Ans.**

(b) Number of bits per word (N) :

As accuracy is expected to be ± 0.1 % of full scale, the maximum quantization error should be ± 0.1 % of full scale.

Bangladesh Rural Electrification Board (BREB)
Assistant General Manager (IT)
Preliminary Exam: 2016

		Answer
1.	**What is the full menaing of SQL?**	
	A Search and Query Language B Simulation of query Language	D
	C Standard query Language D Structured Query Language	
2.	**Maximum Speed of voice band is**	**C**
	A 6900 bps B 6900 kbps	
	C 9600 bps D 96000 kbps	
3.	**Which is not application Software?**	**B**
	A Bing B Red hat Linux	
	C MS Office D Adobe	
4.	**Which one is faster memory**	**D**
	A RAM B Secondary Memory	
	C DRAM D Cache	
5.	**Which is Logical Operator?**	**C**
	A + B >=	
	C AND D <<	
6.	**Which of the following will not increase the value of variable c by 1?**	**C**
	A C++ B C= c+1	
	C C+1>=c D C+=1	
7.	**What is pipilika form www.pipilika.com?**	**None but Related: C**
	A A bangla Font B A Bangladeshi Graphic software	
	C A Bangladeshi browser D A Bangladeshi Game App.	
8.	**Which one is government base procurement website in Bangladesh?**	**D**
	A Cgp.gov.bd B Eprocurement.gov.bd	
	C Procurement.gov.bd D Eprocure.gov.bd	
9.	**Which memory is called Primary memory.**	**D**
	A Hard Disk B Pen Drive	
	C ROM D RAM	
10	**The escape sequence "/b" in c Programming is ……**	**A**
	A Backspace B Next Line	
	C Tab D None of this	
11	**Which one is not operating system software**	**D**
	A DOS B Linux	
	C Windows D Oracle	
12	**Microsoft .NET is**	**A**
	A Open source B Close source	
	C Browser D None of this	
13	**Which is not the kind of data type**	**A**
	A Logical B Text	
	C Number D Currency	
14	**What smart phones are compatible of .apk file**	**D**
	A Microsoft B IOS	
	C Symbian D Android	
15	**Hungarian notation is used to ……..**	**C**
	A Design System Manual B Design User Manual	
	C Define name of the variable D All	
16	**OCR Stand for**	**B**
	A Optical CPU Recognition B Optical Character Recognition	
	C Optical Character Reading D Other character reading	
17	**Which is the universal gate**	**A**