Reference: Introduction to Algorithm By Coreman

Syllabus: (1) Analysis
(2) Divide and conquer.
(3) Greedy Technique
(4) Dynamic programming.
(5) Hashing & Tree and graph Traversal.

Definition: It is a combination of sequence of finite steps to solve a problem.

Example: Multiplication of Two Numbers

```
MTN() { 1. TAKE 2no's (a,b).
        2. Multiply a and b and store result in c.
        3. return c
        }.
```

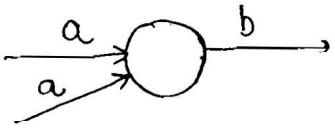
from which function we have come, we have to return there.

- finite steps - finite Time should be there (But it doesn't mean finite steps always leads to finite time)
- infine steps - Infinite Time
- All steps are compulsory, so combination is required, so finally it can solve the problem.

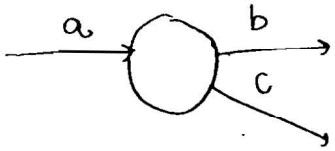
printf → c
cout → c++. } syntax

Properties of Algorithm

1. It should Terminate after finite Time.
2. It should produce "atleast" one output (Min^m 1 output).
3. It should take "0 or More input"
4. It should be "deterministic"
   (different behaviour - Non-deterministic)
   deterministic - always same answer.



deterministic (finite steps) also there.



2 O/Ps. Non deterministic.

No dependency → so we can swap the steps of Algo.
Non deterministic → special case.

Steps Required to Design Algorithm:

1. Problem definition (Knowing problem clearly).
2. Design Algorithm — Divide and conquer
   - Greedy Technique
   - Dynamic Prog.
   - Backtracking
   - Branch & Bough (BB).

$\begin{bmatrix} a-b \\ c-d \\ e-f \\ g-3 \end{bmatrix}$

Algorithm Design: After knowing the problem, Map the problem to the existing Algorithm.

3. Draw flow chart (Diagramatic Algorithm).
4. Testing and verification. (The Report we made (Test cases), our Prog should Run for those i/ps)
5. Coding or Implementation.
6. Analysis the Algorithm.

Run – MM (Go to Run)
Save – Hard disk
Running Time → MM (space complexity).
time complexity.

} operating system process state diagram.

Design and Analysis of Algorithm.

C

Analysis : chapter 1

If your problem having More than 1 solution, Best one will be decided by analysis based on 2 factors.

1. Time complexity (CPU Time)
2. Main Memory (Space complexity).

If your Problem having only 1 soln, go with that soln no no need of Analysis.

Time complexity:

Time required for Prog $T(P) = C(P) + R(P)$

- $C(P)$: compile Time of Prog.
  - ⇓ Based on compiler
  - ⇓ S/w
  - ⇓ Based on language of compiler
- $R(P)$: Running Time of Prog.
  - Based on processor
  - ⇓ H/w
  - ⇓ Type of Hardware.

Based on lang. of Program written compiler, is prog.

Types of Analysis

1. Apostiary Analysis.
2. Apriori Analysis

(postponing the things. (By asking a question, instead of giving answer, asking question to us).

Apostiary.

① It is based on (dependend) on language of compile & Type of H/w.

Adv ② Exact Answer (It will give exact answer bcoz we are considering Real things).

Dis ③ system to system different Answer (diff^nt Time) (constants differ sys to sys)

"it is Relative Analysis".

Here processor & compiler lang. is imp.

Apriori

① It is independent on lang – c. & type of H/w.

② Approximate Answer

Advantage ③ system to system same Answer (same ans with diff^nt sys.).

"Absolute Analysis".

if prog is Running faster prog. written in great logic.

NOTE: Everyone cannot buy supercomputer but everyone can write super ~~comput~~ algo. because lord is given same brain to all. But some people will use it some people will not use it.

software company uses – Apriory Analysis.

APriori Analysis:
we are finding strength of logic.

"It is a determination of order of Magnitude of a statement."
↓
while Running statement is Running how many times.

ex①

```
Main()
{
1. a = y+z;  ⟹ 1 (order of Magnitude).
}
```

(put Big Oh (O) before order of M.

O(1).

ex②

```
main()
{  x = y+3;  ——— 1
   for (i=1; i≤n; i++).
   {
      x=y+3;  ——— n.
   }
}.
```

initialization = 1
condition = n+1
statement = n.
i++ = n.

n+1 = O(n)

Exp③

```
main()
{
   x = y+3;  ——— ①
for (i=1; i≤n; i++)
      x = y+3;  ——— n
for (i=1; i≤n; i++)  ——— n.
{
 for (j=1; j ≤ n; j++)  ——— n
 { x = y+3;  ——— n.n
 }
}
}
```

when bracket is in 1 statement is their No bracket.

1 + n(n).
1 + n² = O(n²).

outer loop — add
inner loop = multiply.

Time complexity is finding Bigger loops.
(Where CPU spending More Time).

give this part to cache Memory, so CPU got to know that it is spending More Time, then. program is fast.

Locality of Reference — cache Memory; (which is More imp)

Example (4)

```
Main ()
{ while (i≤n)
  {
    i=i+1
    i=i+4
    i=i+5
  }
}
```

increamentation.

$i = i+10 \Rightarrow \frac{n}{10} \Rightarrow \frac{1}{10} . n.$

$\Rightarrow O(n)$.

How many times loop is executing $n/10$.

```
main ()
{ i=n
  while (i≥1).
  {
    i=i-1
    i=i-9
  }
}
```

decrementation.

$i = i-10 \Rightarrow \frac{n}{10} \Rightarrow O(n)$.

*

```
i=i-1  > -10
i=i-9  /
i=i+7  > 10
i=i+3.
```

$i = -10+10$

$i = 0$. — not incrementing NO decrementing infinite loop.

example:5

```
main()
{
 i=1;
while (i<=n).
  {
   i= 2*i
  }
}.
```

$1<64$ ✓
$2<64$ ✓
$4<64$ ✓
$8<64$ ✓
$16<64$ ✓
$32<64$ ✓
$64<64$ ✗

64 — 6 steps
32 — 5 steps
16 — 4 steps
$n = \log_2 n$.

Proof

1	1
2	3
2^2	3^2
2^4	:
:	$3^k = n$
:	$\log_3 3^k = \log_3 n$
	$k = \log_3 n$
$2^k = n$.	
$\log_2 2^k = \log_2 n$	
$k = \log_2 n$	

$i = 2*i$
$i = 3*i$

$i = 2*i*3$
$i = 6i$
$k = \log_6 n$

$i = 2*i$
$i = 3*i$
$i = 5*i$ $\Rightarrow$ $i = 30i$
$O(\log_{30} n)$.

II
```
main()
{while (i=n
        i>=1)
  {
   i= i/2 ⇒O(log2n).
  }
}
```

n
$n/2$
$n/2^2$
$n/2^3$
:
$n/2^k = 1$
$n = 2^k$
$\log_2 n = k$

$i = i/2$
$i = i/3$
$i = i/4$ $\Rightarrow i/24$.
$\log_{24} n$.

```
main().
i = 1;
while (i ≤ n)
{
    i = 2 * i;
    i = 3 * i;
    i = i + 3;
}
```

if i = 10.

60 → 63 → 613

same

Neglect addition

Example: 6 proof

```
main()
{
    i = 2
    while (i ≤ n)
    {
        i = i^2
    }
}
```

```
main()
{ i = 2
while (i ≤ n)
{ i = i^12
}
}
```

2, 2^{12}, 2^{24}

2^{12^1}, 2^{12^2}, 2^{12^3}, 2^{12^4}, ..., 2^{12^K}.

$\log_2 2^{12K} = \log_2 n$

$12^K \log_{12} 12 = \log_{12} \log_2 n$

$\log_{12} 12^K = \log_{12} \log_2 n$.

$2 < 1000$.

$4 < 1000$

$16 < 1000$

$256 < 1000$

$(256)^2 < 1000$ ×

$2 = 2^{2^1}$

$2^2 = 2^{2^2}$

$(2^2)^2 = 2^{2^3}$

$(2^4)^2 = 2^{2^4}$.

$(28)^2 = 2^{16} \Rightarrow 2^{24}$.

K

$2^{2^k} = n$.

$2K \log_2 2 = \log_2 n$.

$K \log_2 2 \log_2 2 = \log_2 (\log_2 n)$

$\boxed{k = (\log_2 (\log_2 n))}$.

$\log_2 \log_2 1000$.

$2^{①} = 2^1 = 2$

$2^{②} = (2^1)^2 = 2^{2^1}$

$2^{④} = (2^{2^1})^2 = (2^{2^2})^2$

$2^{⑧} = (2^{2^2})^3 = (2^{2^3})^3$

$2^{2K} = n$.

$2^{12 \cdot K} = n$.

$12^K \log_2 2 = \log_2 n$.

$12^K = \log_2 n$

$K \log_{12} 12 = \log_{12} \log_2 n$

$i = i^{29}$

$i = i^2$

$i = i^{31}$

$\log_{12} 12^K = \log_{12} \log_2 n$.

ex i = 25. → inner case.

25^{29^K}

Ex

i = 25

$i = i^{29}$ ⇒ $O(\log_{29} \log_{25} n)$

// outer Base

$i = i^{29}$

$i = i^2$

$i = i^7$

$(i^{29})^2 = (i^{58})^7 = i^{406}$

$O(\log_{406} \log_{25} n)$

Example:7 for square Reverse is "Root" ⇒ Here decreasing.

```
main
{ i=n.                    termination
while (i>2).
  { i = i^1/2
  }
}
```

$n \longrightarrow n$

$n^{1/2} \longrightarrow$

$(n^{1/2})^{1/2}$

$n^{1/2^2}$

$n^{1/2^3}$

$n^{1/2^4}$

$n^{1/2^K} = 2$ ← termination

$\frac{1}{2^K} \log_2 n = \log_2 2$

$\log_2 n = 1 \times 2^K$

$\log_2 \log_2 n = K \log_2 2$.

$\boxed{\log_2 \log_2 n = K}$

36
3
108

$i = i^{1/36}$ → $O(\log_{36} \log_{29} n)$

$i = i^{1/3}$ → $O(\log_{108} \log_{29} n)$ ← termination

$i = i^2$

$O(\log_{.54} \log_{29} n)$

Computer Organization

10 marks

syllabus:

Module 1: computer architecture

Module 2: computer organization.

Ref Books: 1. computer architecture & organigization.
- Morris Mano. (Hardware design)

2. computer org^{n}.
- william stallings.

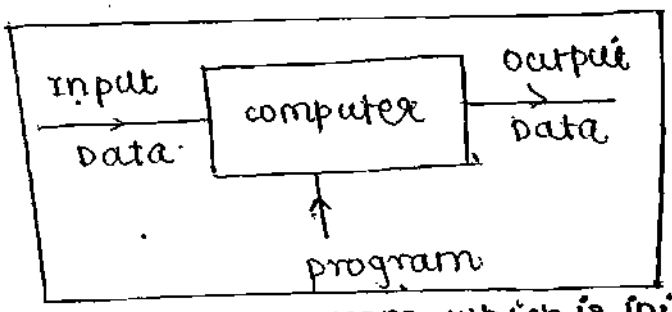
Faculty: Pingili sagar.

email: sagar262003@yahoo.co.in

keywords:

computer:

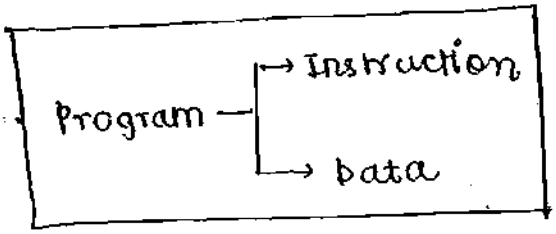
computer is a computational Machine used to process the data under the control of a application program. Therefore computer system functionality is program execution.



(program which is initiated by user)

program:

Program is a sequence of instructions along with the data.

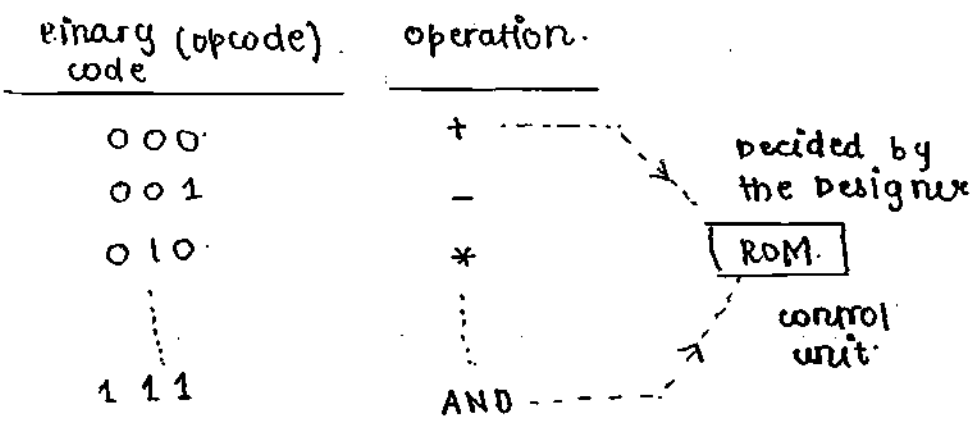


Instruction:

instruction is a Binary code which is designed inside the processor to confirm some task.

Binary code - Bind with - operation

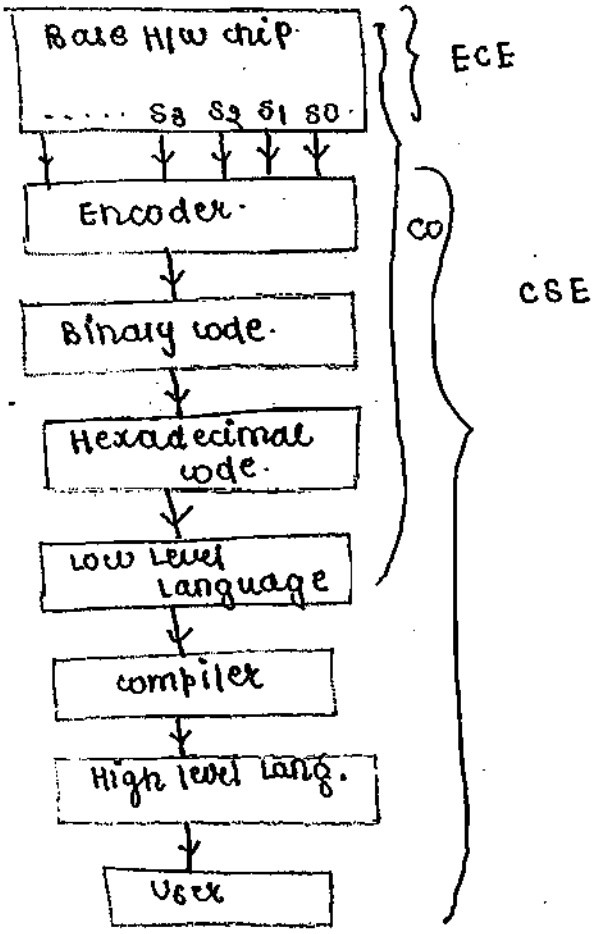
Eg: If CPU – 'x' supports 8 different operation
then opcode = $\log_2 8$ = 3bit



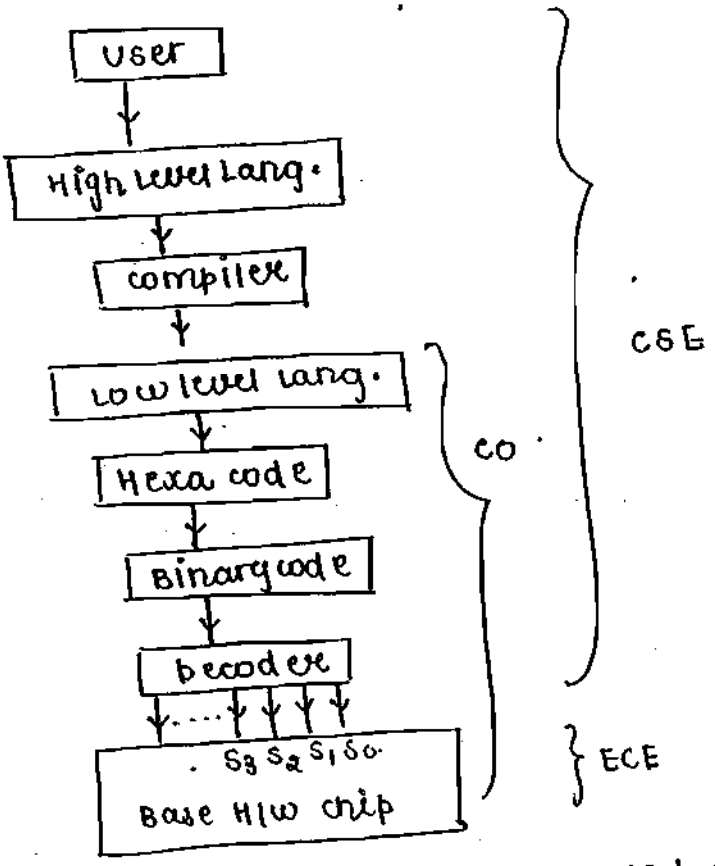
Encoding process: n signals given How many bits required to process signals $\log_2 n$.

Decoding process: n bits are given, How many operation can be performed by computer : '2^n' operation.

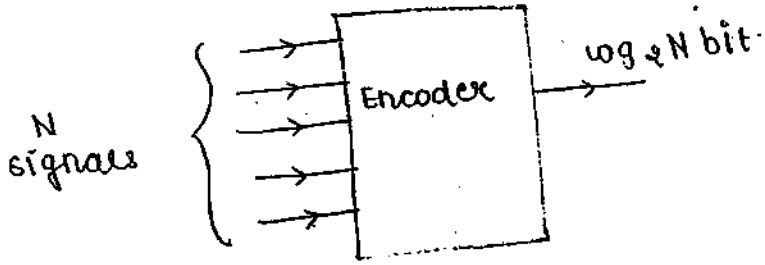
Designer view:



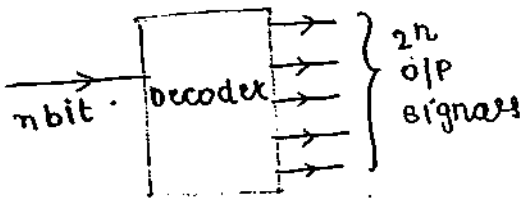
User view :



Encoding: In this process 'N' signals are represented using $\log_2 N$ bit format.



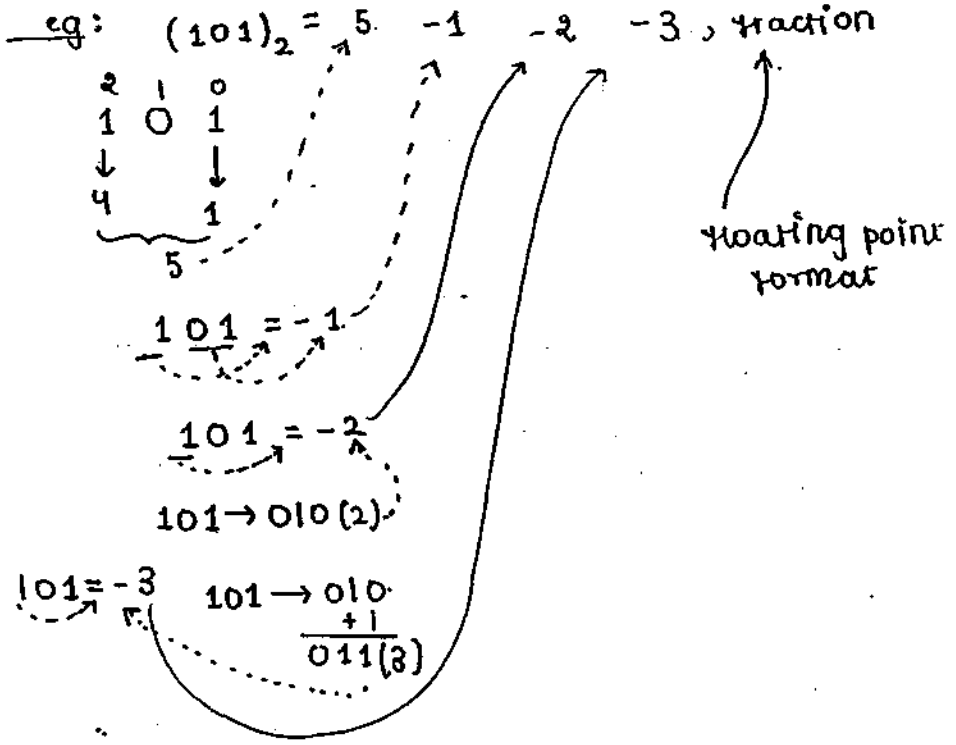
Decoding: In this process, n bit Decoder produces 2^n output signals.



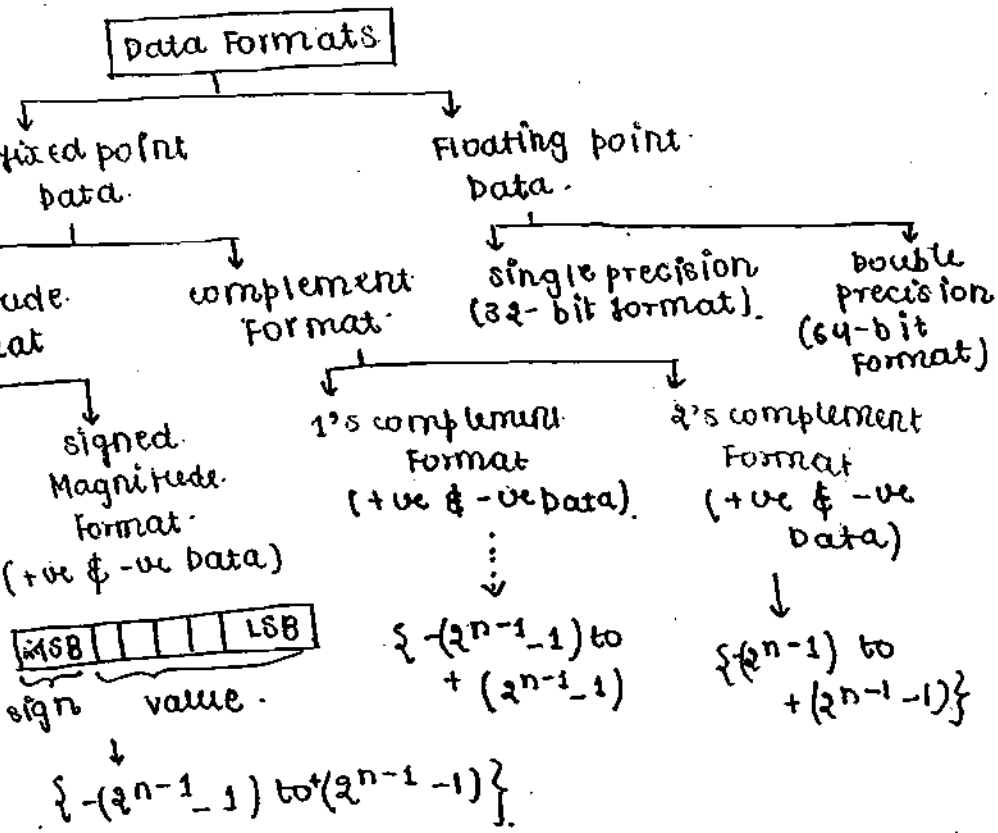
④ Data: It is a Binary code which is associated with a value based on the data format.

Binary code -- Bind with -- value

eg: $(101)_2$ = 5 -1 -2 -3, fraction



Data Representation:



Fixed point Data

4 Bit Binary.	Unsigned Data.	Sign Magnitude.	1's complement	2's complement
0000	0	+0	+0	+0
0001	1	+1	+1	+1
0010	2	+2	+2	+2
0011	3	+3	+3	+3
0100	4	+4	+4	+4
0101	5	+5	+5	+5
0110	6	+6	+6	+6
0111	7	+7	+7	+7
1000		-0	-7	-8
1001	8	-1	-6	-7
1010	9	-2	-5	-6
1011	10	-3	-4	-5
1100	11	-4	-3	-4
1101	12	-5	-2	-3
1110	13	-6	-1	-2
1111	14 15	-7	-0	-1

Data Redundancy Problem. "NOT in USE"

"NOT in USE"

1's complement

1000 = -7

111 (7)

1001 = -6

110 (6)

2's complement

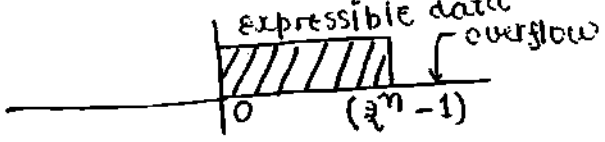
1000 = -8

000
111
+ 1
1000 (8)

1001 = -7

110
+1
111 (7)

unsigned data



eg: 4 bit data {0 to 15}

	①	1	1	1
15	:	1	1	1	1
+15	:	1	1	1	1
30		1	1	1	0

↓ overflow.

Test with 5 bit Data : {0 to 31}

NOTE:

(n-bit) + (n bit) = (n+1) bit.

↓

1 bit storage space required.

↓

1 Flip flop

↓

Flag.

↓

carry flag.

↓

condition : "Is there an extra bit out of MSB" → T = set = 1 = C
(or) → F = reset = 0 = NC

"Is Borrow required into the MSB"

eg:

	×	11
6	:	0110
⊕ 7		0111
13		1101
CY : 0		CY : 0

Justify PSW (program status word)

↓

ACC → 1101 ---- CY = 0 ← Flag Register

Ans : 1101 (13)

Eg:

8	1000
9	1001
17	0001
cy = 1	cy = 1

Justify:

PSW

- Acc → 0001
- Flag Reg → cy = 1

Ans: 10001 (17)

Multiplication:

- Multiplication process is controlled by a Multiplier.
- Two actions are present in the Multiplication.
  (1) generation of partial product
  (2) summation of partial product
- partial product is generated based on the Multiplier bits. i.e when the multiplier bit is '1' partial product is Multiplicant otherwise partial product is zero. (0).
- After the generation of a partial product, provide the solution to produce the final product.

Multiplicant * Multiplier

1 1 1 1 * 1 1 1 1 ← LSB

```
    1 1
    3 1 1 1 1
  3 1 1 1 1 x
2 1 1 1 1 x x
1 1 1 1 x x x
-------------
1 1 1 0 0 0 0 1
```

partial products.

final product

2 → 1 0 → carry (1).

4 → 1 0 0 → carry (2).

6 = 1 1 0 → carry (3)

5 = 1 0 1 → carry (2).

NOTE:

(nbit) * (nbit) = 2n bit

↓

Register pair is used to report the result

Ques: consider the following Multiplication.

$(10w1z)_2 * (15)_{10} = (Y01011001)_2$.

what are the value of w, Y, & z variables?

$(15)_{10} = (1111)_2$.

```
      1 0 w 1 z   *  1 1 1 1.
     ---------------------------
        2 2 1
      2 1 0 w 1 z.
    2 1 0 w 1 z x
  1 1 0 w 1 z x x
  1 0 w 1 z x x x
 ------------------------------
 Y 0 1 0 1 1 0 0 1  ⇒ (Y01011001)2.
 ------------------------------
```

y = 1

z = 1

1 + z ⇒ 1 + 1 = 0

1 carry.

Now replace z with 1.

Now; if (w = 0). for 1 + w + 1 + 1 = 0

1 + 0 + 1 + 1 = 3 (11) — Not matching

if (w = 1)

1 + 1 + 1 + 1 = 4 (100) — matching.

↳ carry (2)

w = 1

NOTE: In a Manual Multiplication process, 2 limitations present

(1) Requires More Registers to Hold the partial product

(2) summation process become complex in the H/w Therefore optimization Required that is Accumulated addition.

Described in Flow chart

COMPILER

Grammar $G = (V, T, P, S)$ → Start Symbol
Variable | Production
Terminals

Example → $S \longrightarrow ABC$
$AB \longrightarrow CD$
$C \longrightarrow a$
$D \longrightarrow b$

Variables : {S, A, B, ...} — given by grammar (check), otherwise if not given, consider Capital letters as variables.

Types of Grammar (according to Noam Chomsky)

1. Type–0 (unrestricted Grammar) — By default, every grammar is Type-0.
2. Type–1 (context Sensitive Grammar)
3. Type–2 (context Free Grammar)
4. Type–3 (Regular Grammar)

Type–0 → $\alpha \longrightarrow \beta$ — unrestricted because no restrictions
where $\alpha, \beta \in (V+T)^*$

Type–1 → ① Type–0 ($A \to \epsilon$, production not allowed)
② $|\alpha| \leq |\beta|$

Type–2 → ① $A \longrightarrow \beta$ (single variable on left side) no restriction on right side.
② $A \in V$
$\beta \in (V+T)^*$

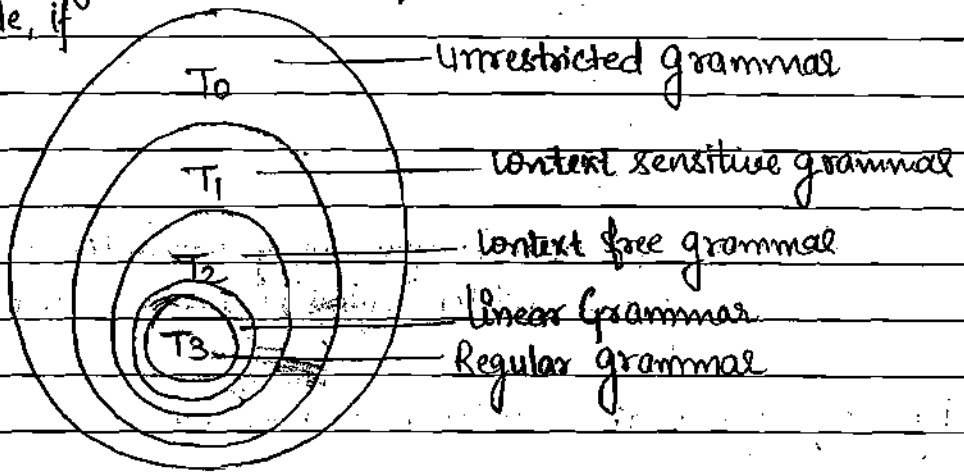
Type–3 → ① $A \longrightarrow \beta T^* / T^*$ (Left linear Grammar)
or
$A \longrightarrow T^* \beta / T^*$ (Right linear Grammar)

Page. No.
Date : / /

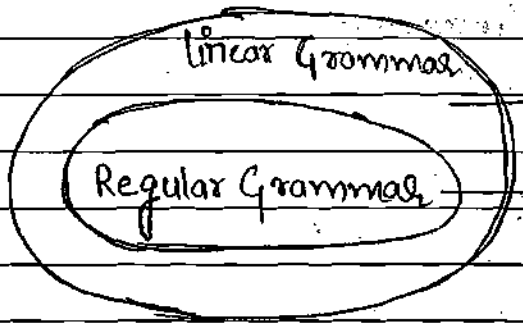
Ex → S → ABC
A → ab
B → CD

T_2, T_3 (Right side)
CFG

① First check left side, single variable, T_2 confirmed
② Then check Right side, if doesn't follow left and Right linear, then not T_3(X)



V → T*VT* | T* → Linear Grammar (In between Type2 and Type3)



Context Free Grammar (CFL)

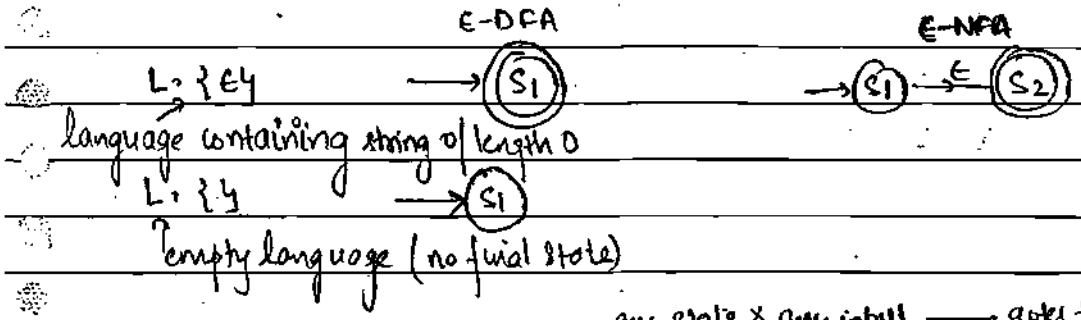
① Write a context free Grammar for a language
$L = \{a^m b^n \mid m, n \geq 0\}$

S → AB
A → aA | ϵ
B → bB | ϵ

Page. No.
Date : / /

ϵ ⟶ string of length 0 (empty string)

$L = \{\}$ ⟶ Empty language



language containing string of length 0

$L = \{\}$

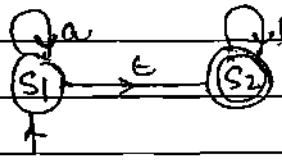
Empty language (no final state)

Transition function ⟩ DFA: $Q \times \Sigma \longrightarrow Q$ (any state × any input ⟶ goes to one of state)

NFA: $Q \times \Sigma \longrightarrow 2^Q$

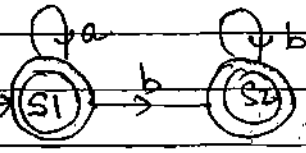
ε-NFA: $Q \times \Sigma \cup \{\epsilon\} \longrightarrow 2^Q$ (includes ε) ⟶ can go to any number of states

a^*b^* ⟶ (Void-Y + ε)



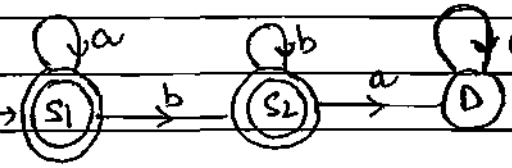
(ε-NFA)

(Void-Y)



(NFA)

(Void-Y / Void-N)



(DFA)

Dead or Trap state → Permanent Non-final states.
Non-final states → Temporary Non-final states

→ Can DFA have more than one final state?

→ DFA can have multiple final states and dfa doesn't accept the NULL move (ε → X dfa)

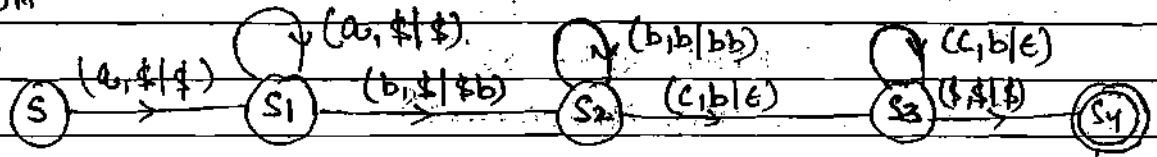
Page. No.
Date : / /

② Give context free Grammar for language
$L = \{ a^m b^n c^n \mid m, n \geq 1 \}$
(A: a^m, B: $b^n c^n$)

$S \longrightarrow AB$
$A \longrightarrow aA \mid a$
$B \longrightarrow bBC \mid bc$

PDA = Finite Automata + 1 stack

DPDA

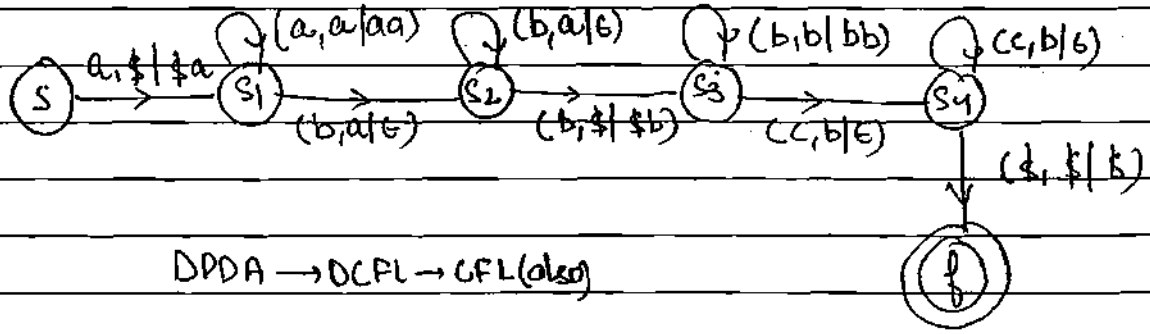


PDA → acceptance by final state
→ acceptance by empty stack

acceptance by final state

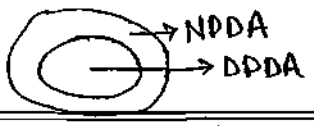
③ Give CFG for the language
$L = \{ a^i b^{i+j} c^j \mid i, j \geq 1 \}$
$a^i b^i b^j c^j$

$S \longrightarrow AB$
$A \longrightarrow aAb \mid ab$
$B \longrightarrow bBC \mid bc$



DPDA → DCFL → CFL (also)

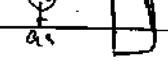
→ NFA is equivalent to DFA
→ NPDA has more power than DPDA.



aabbcc

Page. No.
Date : / /

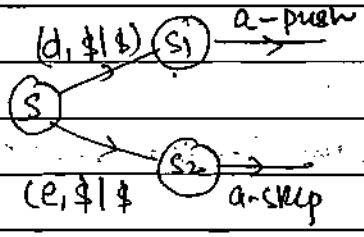
(3) $L = \{ a^m b^m c^n \cup a^m b^n c^n \mid m, n \geq 1 \}$



$S \longrightarrow S_1 \mid S_2$
$S_1 \longrightarrow AB$
$A \longrightarrow aAb \mid ab$
$B \longrightarrow cB \mid c$
$S_2 \longrightarrow CD$
$C \longrightarrow aC \mid a$
$D \longrightarrow bDc \mid bc$

NPDA → CFL language

2) $L = \{ d a^m b^m c^n \cup e a^m b^n c^n \mid m, n \geq 1 \}$



→ no ambiguity
→ DPDA

(5) CFL for language, $L = \{ a^n b^n c^n \mid n \geq 1 \}$
↳ not CFL → It is CSL

LBA = Finite Automata + 2 stacks

(1) → Union of two DCFL's need not be DCFL.
Ex → (4) above. $L = \{ a^m b^m c^n \cup a^m b^n c^n \mid m, n \geq 1 \}$
DCFL CCFL
Union
→ CFL (here in this example)
But can be possible with some examples not.

Page. No.
Date : / /

⊙ Intersection of two DCFL's need not be DCFL.

⊙ Intersection of two CFL's need not be CFL.

⊙ CFL are not closed under complementation

Ex → $a^n b^n c^n$ ← CSL — complement → Other than $a^n b^n c^n$ comb — CFL

(Compiler part)

⑥ Given CFG for language L = set of all Arithmetic expression over the α, β, id.

$E \longrightarrow id \mid E+E \mid E*E \mid E-E \mid E/E \mid (E)$

$id \rightarrow +/-/*/\,/$

$\alpha \rightarrow (id+id)\alpha$

$A\alpha \longrightarrow id\alpha / id\beta A$

$\alpha \longrightarrow +/-/*/$

⑦ Give CFG for language
L = set of all Boolean expressions, over the alphabet 0 and 1.

$B \longrightarrow 0/1/B \text{ or } B/B \text{ and } B/\text{not } B/(B)$

⑧ Give CFG for language L = set of all Regular Expressions over the alphabet a, b ($\Sigma = \{a, b\}$).

$R \longrightarrow \epsilon/a/b/R^*/R+R/R*R/(R)$

$(a+b)^*$

$A \rightarrow a/b/A^*/$

$(a+$

ϵ $A+A)$

$a^*\ b^*$

$R*$

$(R+R)^*$

$(a+b)^*$

$a(a+b)^*$

$R\cdot(R+R)^*$

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

⑨ Consider the following Grammar

$S \longrightarrow aS/Sa/a$

i/p string : aaa

How many parse tree?
(Derivation tree or Syntax tree)

Left most derivation → always explore 'n' most variable first

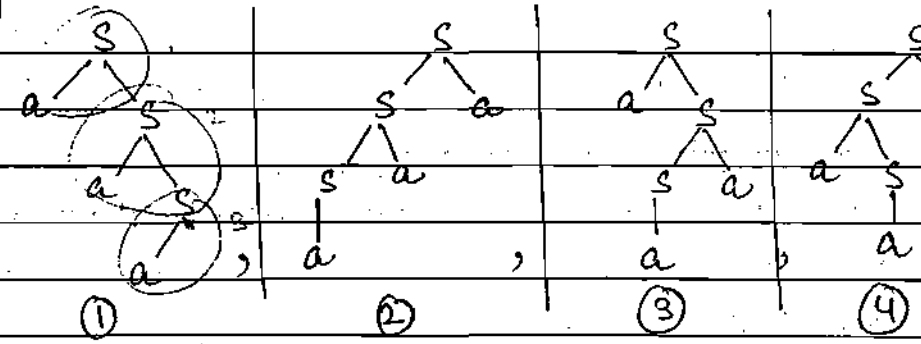
Right most derivation → always explore Right most derivation first

Network Layer — Transport Layer —

Page. No.

Date: / /

String → aaa



→ To know, what string is generated → read all leaf nodes from left to right.

In above, left most derivation tree → 4

Right most derivation tree → 4

① In above, example, only one variable, so LMDT and RMDT same (even numbering too)

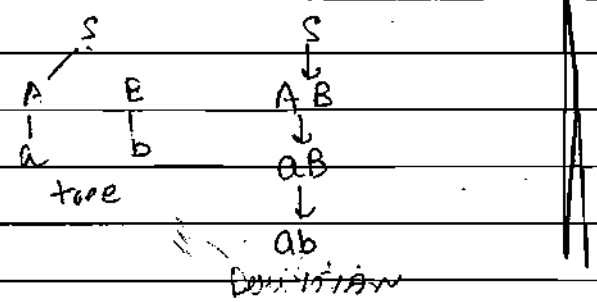
② If more than one variable, then check (can defer in numbering)

→ Always no. of left Most Derivation tree = no. of Right Most Derivation tree, only numbers changes (which may not even change sometimes)

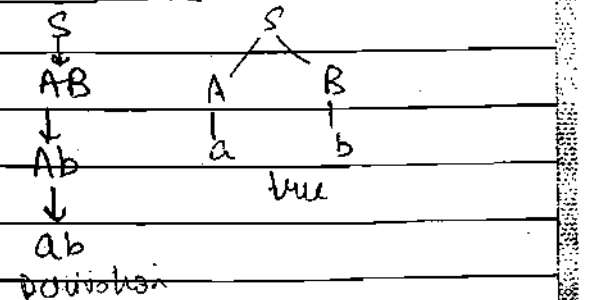
No. of Left Most Derivation tree = No. of Right Most Derivation tree = No. of Parse tree

→ Left most Derivation and Right most Derivation may differ, but their trees will always be same.

LMD's



RMD



→ So, Derivation are differing, but both have same parse tree.

→ Above Grammar is Ambigious Grammar, because more than one Parse tree possible.

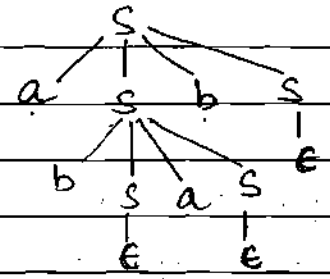
→ A Grammar G is said to be ambigious, if for atleast one string, more than one parse tree possible

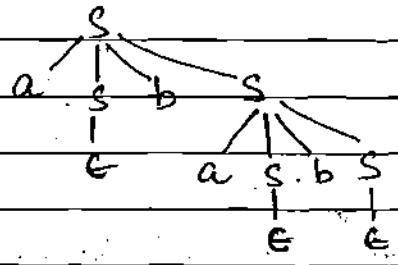
→ Checking given grammar is ~~ambigious grammar or not~~ undecidable, ~~because there is no algorithm ambigious or not~~, is undecidable, beoz there is no algorithm to check it.

Q) Consider the following grammar

$S \rightarrow aSbS \mid bSaS \mid \epsilon$

I/p string → abab$





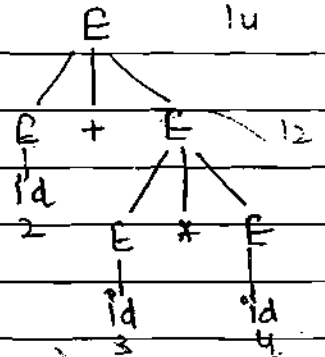
Above is Ambigious grammar.

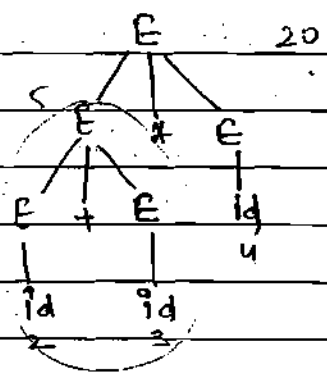
Q) Consider the following grammar

$E \rightarrow id \mid E+E \mid E*E$

Input string: id+id*id

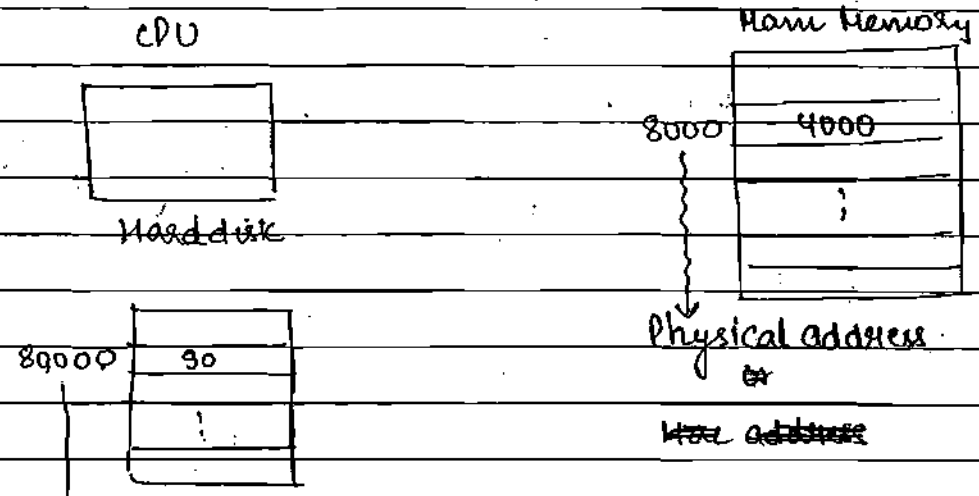
2 + 3 * 4





→ Ambigious Grammar.

Computer Network and Security



MAC address is local, so cannot be used as unique identification Number

Physical address
(or)
Mac address
(or)
Ethernet address ⇒ 48 bit address
(or)
LAN card address
(or)
NIC card address

→ Implicit address

① S.A, D.A ② S.MAC, D.MAC — Scope is local

logical address (classful addressing)
32 bit address (IPv4)

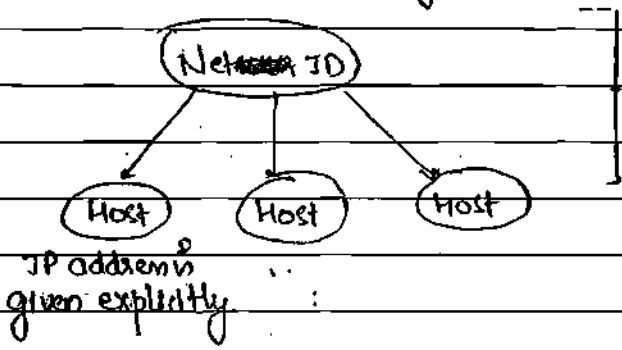
IANA → Internet Assigned Number Authority

Note> Using MAC address alone cannot be used as an identification unit in transmitting the data, because scope is local.

(IP) logical address → 32 bit address
address
↓ classes → A, B, C, D, E — Research
Unicasting
multicasting

Page. No.
Date : / /

⊙ classful supports two level hierarchy.



www.yahoo.com
hostname

→ Whenever an IP address is assigned to a computer, it is known as host.
→ Entire Network will be represented by a number known as the Net ID.

Notation

i) Binary notation [2]

Ex- 10101111 10101010 10101111 11110000
System friendly. 1st octal 2nd octal 3rd octal 4th octal

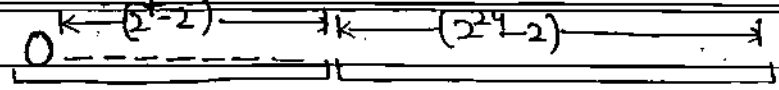
ii) Dotted Decimal Notation [10]

Ex- 143.89.99.126 ——→ User friendly.

——→ In Binary notation starting few bits will decided the type of class
——→ In dotted decimal notation, first octate will decided the type of class.

Page. No.
Date : / /

Class A



Net bits
8bit

Host bits
24 bits

0 0000000 → 0

1 1111111 → 127

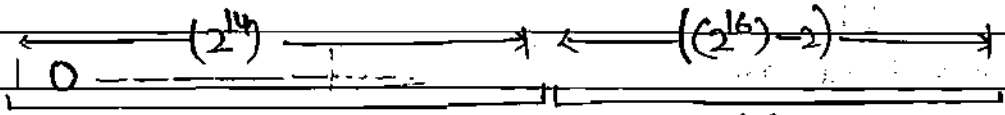
(0—127) but 0 and 127 not used

∴ (1—126) → Class A

0.0.0.0 → DHCP client

127.x.y.z → loop back address

Class B



Net bits
16 bits

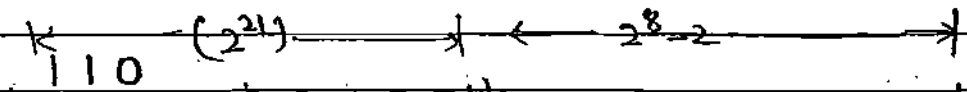
Host bits
16bits

1 0 000000 → 128

1 1 1111 → 191

Class B Range → (128—191)

Class C



Net bits
24 bits

Host bits
8bits

1 1 0 00000 → 192

1 1 0 11111 → 223

Class C Range → (192—223)

224
+5
239

Page. No.
Date: / /

Class D — 1110 — no division — multicasting

1110 0000 → 224
1110_1111 → 239

Class D Range → (224–239)

Class E — 1111 — Research

1111 0000 → 240
1111 1111 → 255

Class E Range → (240–255)

Q) IP: 201.44.89.99

Net Id =

Direct Broad cast address of network =

Network mask (Default mask or)

Class A: 11111111.00000000.00000000.00000000

maskA → 255.0.0.0
Class B: 255.255.0.0
Class C: 255.255.255.0

Mask → allowing and stopping

→ Network mask is a mathematical tool which is used for solving networking problems

	dotted decimal	binary
IP:	201.44.89.99	99: 01100011
mask:	255.255.255.0	01 00000000
	201.44.89.0 (And)	01 00000000

x	y	output and
0	0	0
0	1	0
1	0	0
1	1	1

89: 01011001
255: 11111111
89: 01011001

When want to send address to all the host present

Page No
Date: / /

201.44.89.0

Net Id

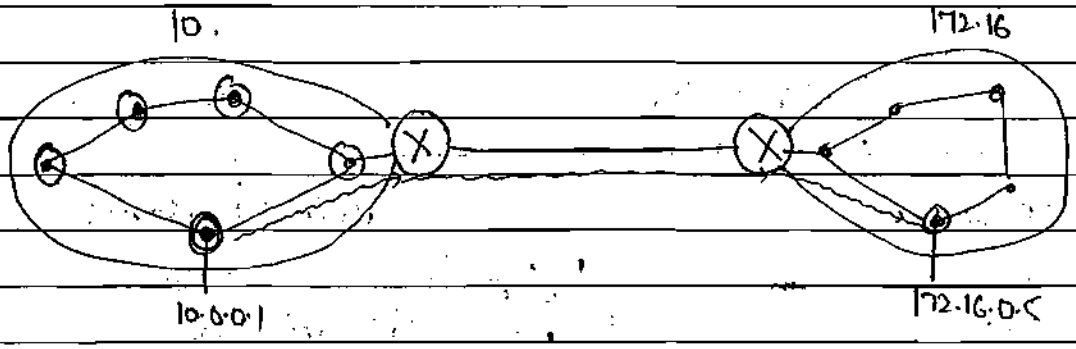
Direct broadcast address.

201.44.89.255

201.44.89.1 | 201.44.89.2 | 201.44.89.254

① We are subtracting 2 addresses in the no. of host, because one is used for net Id and other one is used for DBA of network

Pseudo approach of network



(i)

	S.IP	D.IP
D	10.0.0.1	172.16.0.5

Class B

It is a unicasting packet between the networks (different networks)

(ii)

	SIP	DIP
D	10.0.0.1	172.16.255.255

all 1's in host bit

Directed broadcast address

D	172.16.255.255 (X) (no. meaning)	10.0.0.1

signifies Direct broadcast address

→ It is a broadcasting packet on the other network

→ Direct Broadcast address will always be used as Destination IP.

(iii)

	S.IP	D.IP
D	10.0.0.1	10.0.0.9

It is a unicasting packet within the network. (same network)

(iv) Special case →

D	10.0.0.1	255.255.255.255

(Broadcast within the network) ↑ ↓ (limited broadcast address → scope is local (LAN))

→ limited Broadcast address will always be used as destination IP.

(used in LAN)

IP address

Private IP address

1. Scope is local
2. Work only in LAN
3. By loading networking operating system
4. Ranges of private IP

	No. of Network
10.0.0.0 — 10.255.255.255	1
172.16.0.0 — 172.31.255.255	16
192.168.0.0 — 192.168.255.255	256

5. free of cost
6. will not get Internet service

Public IP address

1. Scope is globally unique
2. used to get Internet service
3. Not free of cost
4. have control of ISP (Internet Service provider)

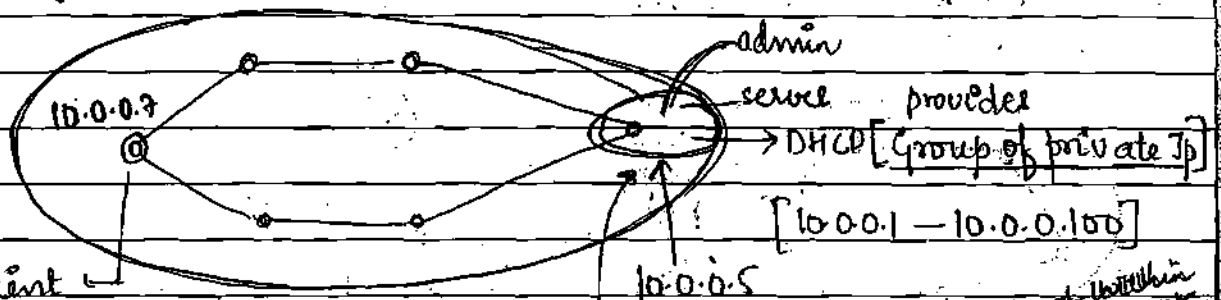
Page. No.
Date: / /

Client => DOS, XP [DOS commands]

Server => WindowsNT, 2003

[DOS commands + Networking protocols (Http, Ftp, Dhcp, —)]

Assigning Private IP addresses in LAN (stateful protocol)



D	0.0.0.0	10.0.0.5

→ DHCP client

Hac address
14:1A:12:1B:14:15
Who identify uniquely the system, mac address is added

broadcast within network

	S.I.P	D.I.P
D	10.0.0.5	255.255.255.255

(to inform all clients, its source IP

Mapping table

HAC	IP
14:1A:12:1B:14:15	10.0.0.7

(1) Once the server is loaded with network operating system, it will get group of private IP addresses, out of which one IP is assigned to server

(2) the servers IP is informed to all the clients using limited Broadcast address.

(3) Every client will put a request to the server using DHCP client as the source IP address, along with it Hac address is

★ Stateful → cannot be removed freely, but informed to mapping table as well that entry in mapping table too, then remove, other problem like Dangling pointer

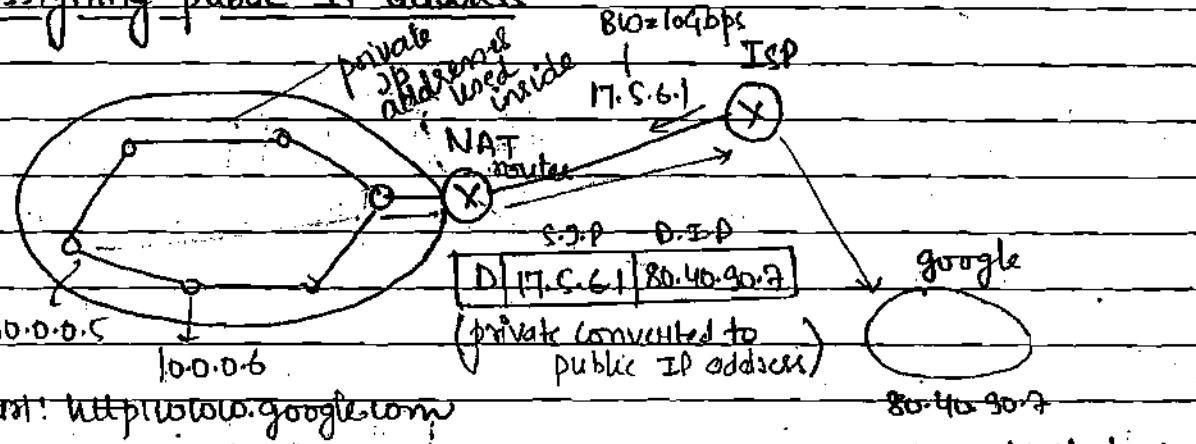
Page fault
TLB:

transmitted, so that server can understand which computer is requesting.

④ In response to it, server will assign 1 IP address to that client, by maintaining mapping table.

⑤ The purpose of mapping table is to identify which IP is assigned to which computer.

Assigning public IP address



NAT → Network address Translation

⊙ NAT Router converts private IP into Public IP, when the packet is going out of the network. It converts public IP into private IP, when the packet is coming inside the network.

① Public IP addresses are effectively utilized using private IP addresses. (If private IP not available, difficult to then only public IP addresses will be assigned and they will get exhaust easily)

→ Revision
2 (2-3 days)
→ short notes

Page. No.
Date: / /

DBMS

	More Practice	Less Practice
Normalization	• finding candidate key ① Lossless Join / DP ② INF to BCNF def / theory ③ Higher NF?	④ Canonical cover ⑤ Decomposition into higher NF.
Queries	⑥ RA / SQL	⑦ TRC (basics)
File & indexing Organisation	⑧ B / B+ tree [Theory / Numerical]	⑨ Dense index / Sparse index.
Transaction and concurrency	⑩ CSS / VSS	⑪ ACID Re / CLR / St lock TSO
ERD	⑫ Min RDBMS table for ERD	

DM and BA and Appli

(Algo) + C & DS

DBMS — (8–10)
ToC — (8–10)
CD — (3–5)
Digital — (3–5)

90%
(Parser & SDT) ⇐

20-30%
DAA
TH

60-70
DFA / NFA
min dfa

CN	CO	OS
(10 marks)	(8-10)	(8-10)

⑬ Top down Approach (Kruze)
⑭ Problem

Ravi Sir → 8024172708

DBMS [8–10 Marks]

Syllabus →
1. Integrity Constraints and ER-model — 2 Marks
2. Normalization. — 2 Marks
3. Queries [Relational Algebra, SQL, Relation calculus] — 4 Marks
4. File Organisation and Indexing. — 2–4 Marks
5. Transactions and concurrency control. — 2–4 Marks

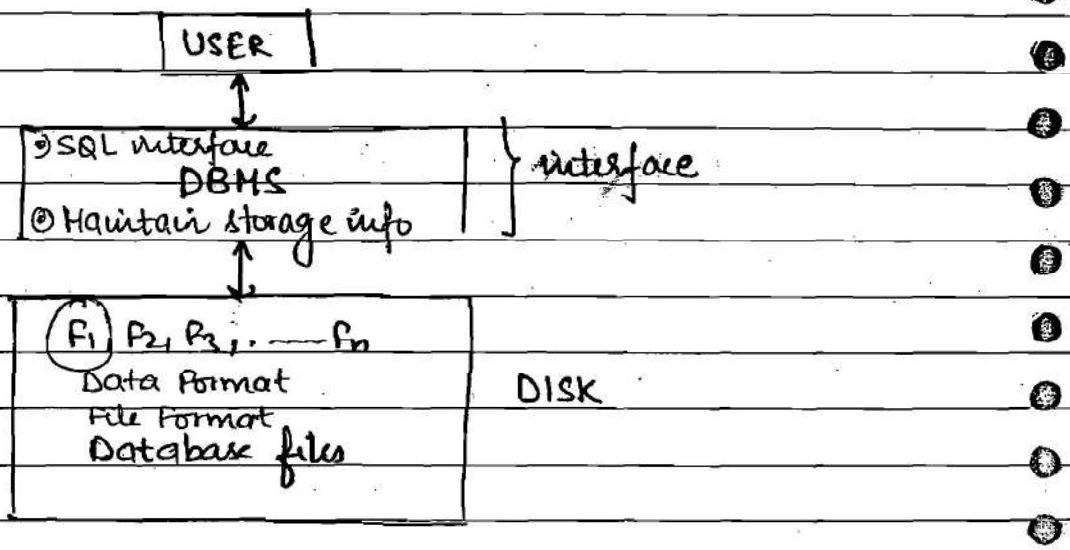
→ Notes (Revise 2 times a day)
→ WB/GATE
→ Text book Ex Problems
  → 1) DBMS by Raghuramakrishan
  2) DBMS by Navathe

Introduction to DBMS

Page. No.
Date : / /

① Database → Collection of related data
Ex → set of student's information

② Database Management System (DBMS) → Software used to manage and access database more efficient manner. In other words, acts as interface between user and database files

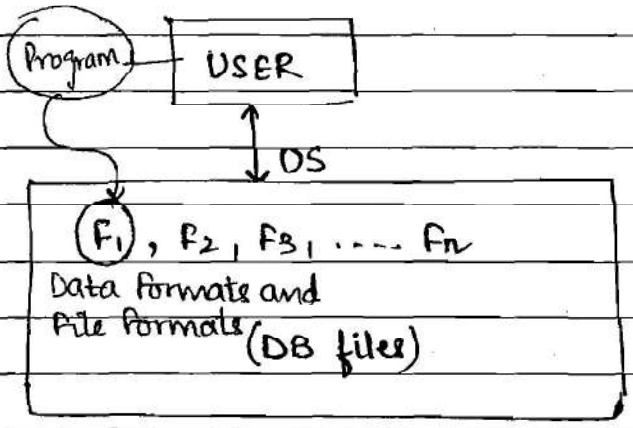


Users is

Data Independency → User can access data from database files without knowing storage information of file system. In other words, hidding storage information to the external users, is called data independency

Flat File System [DS files] → ~~when the user~~ Database files managed by user without DBMS software.

Page. No.
Date : / /



- Flat file System can be used to manage small database.
- Flat file System fails to manage, if size of database is too huge.

Limitations of Flat File System Vs Advantage of DBMS file System

Limitation of Flat file System	Advantage of DBMS file system
1) Too Complex to manage and develop application programs	1) Because of data independency, easy to develop application programs (simple SQL Query required to access the data).
2) More I/O cost to access data from database files	2) Because of Indexing, less I/O cost required to access data from database files.
3) less degree of concurrency.	3) More degree of concurrency.
4) Too complex to maintain, ~~non~~ non-redundant data.	4) By using normalization of database, can maintain non-redundant data.

Page. No.
Date : / /

Integrity Constraints

Data Models

RDBMS | ODBMS (Object database management system) | NWDBMS (Network database management system) | Hierarchical DBMS

RDBMS ⇓

Relational DBMS (RDMS)

↳ Proposed by R.T. Codd [Codd's data Model]

↳ Codd proposed 12 Rules for RDMS software design (RDBMS Guidelines)

RDMS Guidelines

① Data in DB file must be in tabular format [i.e set of Rows and Columns].

② No two Row's of DB table should be same. [candidate key]

Stud	Sid (1st candidate key)	Sname	DOB (each column is called Attribute / Field)
Relational instance	S1	A	2000
	S2	B	1995 ← each row is called Record / tuple
	S3	B	1998
	S4	C	1995

RDBMS file

(3arity for above)

- Arity → No. of attributes of database table. (i.e - Sid, Sname, DoB - 3)
- Cardinality → No. of Records of database table. (above → 4)
- Relational Instance (Snapshot) → A record set of database table.
- Relational Schema → Defination or structure of database table.

Ex → Stud(Sid, Sname, DOB) → Relational Schema.

Key for a Relation is defined according to the requirements of user. Key can be one, which can uniquely distinguish it from others.

Page. No.
Date : / /

Candidate Key → Minimal set of attributes used to differentiate records of a relational schema, uniquely.

Minimal Attribute Set	+ No two records with same attribute value (unique)	Candidate Key

Ex → (1) Stud(Sid, Sname, DOB)

Sid : Candidate key.

Sid Sname : Not Candidate Key

(2) Enroll(Sid Cid Fee)

S1	C1	—
S1	C2	—
S2	C1	—
S2	C2	—

Sid Cid : Candidate Key

Ex → A student having student id can enroll in more than one course and a course can be enrolled by more than one student.

Primary key can be → student-id (here unique)

→ More than one candidate key is allowed

(3) Emp

eid	ename (Not NULL)	DOB	PanID	Aadhaar	IFSC	Accno.
c1	A		X2		SBI01	101
e2	B		NULL		SBI01	102
e3	A		X5		ICI01	101
e4	C		NULL		ICI01	102

→ NULL : Unknown / un existed value

→ Not Null constraint : Null values not allowed

Candidate key of Emp relation → {(eid), PanID, Aadhaar, ifsc Accno}

eid → Primary Key (Keyword used {PRIMARY KEY})

PanID, Aadhaar, ifsc Accno → Alternative Keys [UNIQUE] - Keyword used

[5-8 Marks]

Digital Logic Design

Sri Nivas Sir.

Syllabus:

Text Book
Modern Digital Electronics
- RP Jain
TMH Publication.

SRINIVAS BETHI
9959750099

1). Boolean Algebra
- Boolean Variables
- Boolean operators
- Logic Gates
- B.A. properties
- Derived operators
- Universal Logic Gates
- Boolean function using Universal Logic Gates
- Simplification
- Self Dual functions etc.

2). Number System
- Conversion $(\ )_{r_1} = (\ )_{r_2}$
- $(r-1)$'s and r's
- Signed B. No. System etc.

3). K-Map
- Implicant
- Prime implicant
- Essential Prime implicant.

4). Combinational Circuits
- Code converters
- Arithmatic Circuits
- MUX, Decoder,
- Encoder, DeMUX etc.

5) Sequential Circuits
- Binary Latch
- Flip-flops
- Flip Flop Conversions
  - FF1 $\longrightarrow$ FF2
  - $x_1 y_1 \longrightarrow x_2 y_2$
- Registers $\Rightarrow$ SISO, SIPO etc..

6). Counters
- a) Asynch.
- b) Synch.

Boolean Algebra ⟷ Chapter 1

$n=2$ ← Binary.

~~Binary~~

Boolean Variables ⇒ A, B, C ...
a, b, c ...

Operators ⇒ OR, AND (Binary), NOT (Unary).

OR Operator → +, ∪, ∨

$Y = A+B = A \cup B = A \vee B$

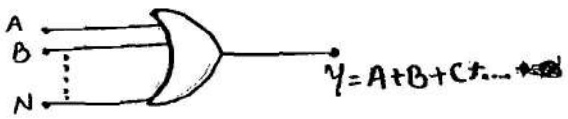
$Y = A+B+C+\ldots\ldots$

A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

A B C	Y = A+B+C
000	0
001	1
010	1
011	1
100	1
101	1
110	1
111	1

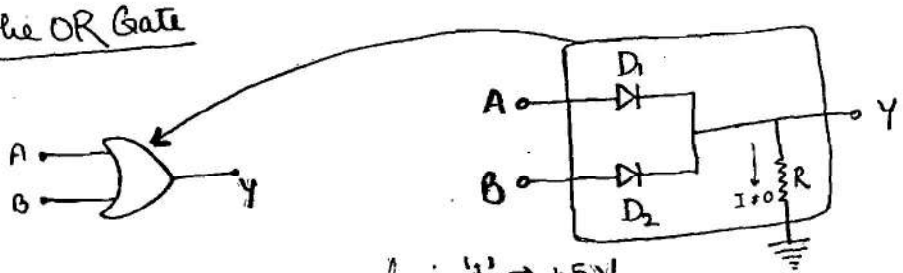
Note: • The result of OR operation is zero if and only if, all the variables are zero.

• OR Gate



• No. of inputs in the logic gate is known as Fanin of the logic gate.

for 2 Fanin of the OR Gate



logic '1' ⇒ +5V
logic '0' ⇒ 0V

Truth Table is one consisting of all possible combination of the variables along with the result.

for $n \Rightarrow 2^n$ Rows

values $\Rightarrow [0, 1, 2 \ldots\ldots (2^n - 1)]$

AND OPERATOR $\longrightarrow \cdot , \cap , \wedge$

$$Y = A \cdot B = A \cap B = A \wedge B$$

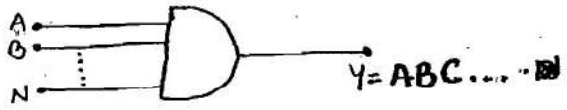
$$Y = A \cdot B \cdot C \ldots\ldots$$

A	B	$Y = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

A	B	C	$Y = A \cdot B \cdot C$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

NOTE : • The result of AND operation is zero, if atleast one of the variable is zero.

• AND Gate



#

	OR	AND
if $A = 0 \Rightarrow Y =$	B	0
if $A = 1 \Rightarrow Y =$	1	B
if $A = B = x \Rightarrow Y =$	x	x
if $A \neq B \Rightarrow Y =$	1	0
Enable input $\Rightarrow$	0	1
Disable input $\Rightarrow$	1	0

exp.
In OR Gate
$Y = A + B + C$
if $B = 0$, $Y = A + C$
if $B = 1$, $Y = 1$

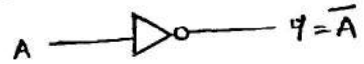
• Enable i/p is the one, it makes the device active.
Disable i/p is the one, it will make the device is to be inactive.

NOT Operator : '—' , ' ' ;

$$Y = \overline{A} = A'$$

A	Y = NOT A
0	1
1	0

Note : • NOT Gate



• NOT operator is also known as Inverter.

BOOLEAN ALGEBRA PROPERTIES :

1). $A + A + \ldots = A$
$A \cdot A \cdot A \cdot \ldots = A$

2). $A + 0 = A$
$A \cdot 1 = A$

3). $A + 1 = 1$
$A \cdot 0 = 0$

4). $A \cdot \overline{A} = 0$
$A + \overline{A} = 1$

5). $\overline{\overline{A}} = A$

6). $A + BC = (A+B)(A+C)$ Distributivity.
$A \cdot [B+C] = AB + AC$ Dual of (6).

7). $A + \overline{A}B = A + B$
$A[\overline{A} + B] = AB$

8). $\overline{A} + AB = \overline{A} + B$
$\overline{A}[A+B] = \overline{A}B$

9). $A + AB = A$
$A[A+B] = A$

• **Dual Operation / Principle of Duality** → OR ⟷ AND, 0 ⟷ 1.

Ques. $\overline{A} + B[C + \overline{D}(\overline{E} + F)]$

Dual ⇒ $\overline{A}[B + C[\overline{D} + \overline{E}F]]$

• Variable 'x' ⇒ 'x'
Literal 'x' ⇒ 'x' or '$\overline{x}$'

• There is NO effect of the dual operation on the Literal.

- $A+BC = (A+B)(A+C)$ ← Property Proof.

A	B	C	LHS	RHS
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Dual

$A\cdot[B+C] = AB + AC$

- $A + \bar{A}B = (A+\bar{A})(A+B)$
  $= A+B$

- $A + AB = A[1+B]$
  $= A$

Ques① $\bar{x}\bar{y}z + xz + yz$ (The no. of gates required to perform this is :)

a). 0 b) 1 c). 3 d). 4

$z[\bar{x}\bar{y} + x + y]$

$z[x + \bar{y} + y]$

$z[x+1]$

z

Ques② The simplified form of the given Boolean expression. is :

a). $\bar{A}(B+C)$ b). $A(\bar{B}+\bar{C})$ c). $B(A+C)$ d). None.

$A\bar{B} + AB\bar{C} + A\bar{B}C\bar{D}$

$= A[\bar{B} + B\bar{C} + \bar{B}C\bar{D}]$

$= A[\bar{B} + \bar{C} + \bar{B}C\bar{D}]$

$= A[\bar{B} + \bar{C} + C\bar{D}]$

$= A\bar{B}[1 + C\bar{D}] + AB\bar{C}$

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

Ques③. The simplified form of given Boolean expression :

$AB + \bar{A}B + A\bar{B}$

$= B\cdot 1 + A\bar{B}$

$= A+B$

Properties

10). $\underbrace{AB + \bar{A}C + BC}_{\text{AND} \Rightarrow 3,\ 3\text{I/p OR} \Rightarrow 1} = \underbrace{AB + \bar{A}C}_{\text{AND} \Rightarrow 2,\ 2\text{I/p OR} \Rightarrow 1}$ (Redundant Term)

[Consensus property] OR [Redundancy property]

LHS: $AB + \bar{A}C + BC(A + \bar{A})$

$= AB + \bar{A}C + ABC + \bar{A}BC$

$= AB + \bar{A}C$

Dual: $(A+B)(\bar{A}+C)(B+C) = (A+B)(\bar{A}+C)$

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

LHS = $X\bar{X} + XZ + \bar{X}Y + YZ$

$X\bar{X} \rightarrow 0$

$= XZ + \bar{X}Y.$

Dual: $XY + \bar{X}Z = (X+Z)(\bar{X}+Y)$

12). Demorgan's Property.

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

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

Ques④ Write the Demorgan's result of the given Boolean expression

$\bar{A} + B[C + \bar{D}(\bar{E} + F)]$

Demorgan's $\Rightarrow A[\bar{B} + \bar{C}[D + E\bar{F}]]$

Ques⑤ if $f(A,B) = \bar{A} + B$.

then $f(f(x+y, y), z) = ? = f(\overline{x+y} + y, z)$

$= f(y + \bar{x}, z)$

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

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

a). $\bar{x}y + z$

b). $xy + z$

✓c). $x\bar{y} + z$

d). NONE.

Ques⑥ if $x * y = \bar{x} + y$ and $z = x * y$

then $z * x = ?$

$z * x = \bar{z} + x$

$= (x * y)' + x$

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

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

$= x$

a) $\bar{x}$

✓b). x

c). y

d). NONE

10-Sept

DISCRETE MATHEMATICS

Kenneth Rosen

9 Marks

Chapters:

1. Logic
2. Combinatorics
3. Set Theory [KOLMAN, BUSAN & ROSS]
4. Graph Theory [NARSINGH DEO]

[Theory]

LOGIC

1. Logical Statement ? [Proposition]
2. Logical Operators & their properties ($\vee, \wedge, \neg, \Rightarrow, \Leftrightarrow, \oplus \uparrow \downarrow$) — Disjunction, Conj., Negation
3. Tautology (T), Contradiction (C) & Contingency (CT) [Satisfiable (T or CT) / Unsatisfiable (C)]
4. Normal Forms & properties: PDNF (Principle Disjunctive Normal Form), PCNF (Principle Conjunction Normal Form)
5. Implications & Biconditional ($\Rightarrow, \Leftrightarrow$)
6. Arguments & Fallacy [Invalid Argument]
7. Rules of Inference
8. Predicate Logic - Quantifier ($\forall, \exists$)
   - Validity of a predicate
   - Properties
   - Translation

LOGIC

$(S, \circ)$ — $\circ$: operators, S: set of all logical stmt

Logical Statement (Proposition)

- Declarative sentence which can be either true or false but not both.

Ex - This board is white.

This Fan is Rotating [is/will tends to declaration]

- This sentence is true.

Not a Logical Statement

1. Questions - What is Your Name?
2. Command - Stand up.
3. Exclamation - Oh! That's great.
4. $x+2=4$

   (it is not proposition bcoz for some x value it is true ——— it is false)
5. (He) is tall. (unless he is specified)
6. Today is Wednesday.

   [Not a proposition bcoz today may be true, but tommorow it will become false]
7. Tommorow it will rain.

   [Not a proposition.]
8. This sentence is false.

   [Negative Self Referential Sentence]

Logical Operators :

A proposition is written in the following way:

$p: 2+2=4$

$q(x): x+2=4$ (Predicate) but not a proposition

~~False~~ True — $\forall x\, P(x)$

True — $\exists x\, P(x)$] — propositions

Negation — $(\sim, \bar{P}, \neg, P')$ · Unary operators

P	$\bar{P}$
0	1
1	0

P	Negation
is	is not
isnot	is
$=$	$\neq$
$<$	$\geq$
$>$	$\leq$
$p \vee q$	$p' \wedge q' \Rightarrow p \downarrow q$
$p \wedge q$	$p' \vee q' \Rightarrow p \uparrow q$

P	Negation
$P \Rightarrow q$	Pq'
$p \Leftrightarrow q$	$p \oplus q$
$p \oplus q$	$p \Leftrightarrow q$
$p \uparrow q$	$p \wedge q$
$p \downarrow q$	$p \vee q$

- if $p \vee q = 1$
  than $p = \sim q$ is one possibility but not the sure thing. it also allow some other thing
- if $p \wedge q = 0$
  $\Rightarrow [p = \sim q]$ not allways.
- If $p \vee q = 1$ & $p \wedge q = 0 \Rightarrow \boxed{p = \sim q}$
- If $p \Rightarrow 2+2 = 4$ or $3+7 = 10$
  $\beta \Rightarrow$ "$2+2 \neq 4$" and $3+7 \neq 10$
- If $p \Rightarrow 2+2 = 4$ and $3+7 = 10$
  $\bar{p} \Rightarrow 2+2 \neq 4$ OR $3+7 \neq 10$
- p: 2 is even & divisible by 4.
  p': 2 is odd or not divisible by 4.
- p: if it rains, I will carry umbrella. [Either it does not rain OR I will carry Umbrella]
  p': It rains and I will not carry Umbrella

$\wedge$ - all must be true
$\vee$ - atleast one is true
$\oplus$ - Exactly one is true
$\uparrow$ - Atleast one is false

Conversion of Secondary operators into Basic operators:

- $p \rightarrow q = p' + q$
- $p \Leftrightarrow q = p'q' + pq = (p \oplus q)' = p' \Leftrightarrow q' = p' \oplus q = p \oplus q'$
- ~~$p \Rightarrow$~~ $p \oplus q = pq' + qp' = p' \Leftrightarrow q = p \Leftrightarrow q' = p' \oplus q'$
- $p \Leftrightarrow q = (p' + q)(p + q')$ $[(p \Rightarrow q) \wedge (q \Rightarrow p)]$
- $p \oplus q = p' \oplus q'$
- $p' \oplus q = p \Leftrightarrow q = p \oplus q'$

- p: A number is even if and only if divisible by 2. $[\bar{p} \Rightarrow \bar{q} \wedge p \Leftarrow q]$
  p': A number is even or it is divisible by 2, but not both.

- NOR - Neither ... NOR
- OR - Either ... OR

Negation for predicate-

P(x)	$\sim P(x)$
$\forall x\, P(x)$	$\exists x \sim P(x)$
$\exists x\, P(x)$	$\forall x \sim P(x)$
$\forall x \sim P(x)$	$\exists x\, P(x)$
$\exists x(\sim P(x))$	$\forall x\, P(x)$

$\sim(\forall x(P(x)\to Q(x))) \equiv \exists x(\sim[P(x)\to Q(x)])$

$\equiv \exists x(P(x)\wedge\sim Q(x))$

$\sim(\forall x\exists y\, P(x,y)) \equiv \exists x\forall y \sim P(x,y)$

$\sim(\exists x\,\forall y\forall z\,(P(x,y,z)\oplus Q(x,y,z))) \equiv \forall x\exists y\exists y\,(P(x,y,z)\Leftrightarrow Q(x,y,z))$

$\not\Rightarrow \sim(p\Rightarrow q) = \sim(p'+q)$

$p\Rightarrow q$	[stmt]	$(p=0 \text{ or } q=0)\Rightarrow(pq=0)$
$q\Rightarrow p$	[converse]	$(pq\neq 0)\Rightarrow(p=0 \text{ or } q=0)$
$\sim p\Rightarrow\sim q$	[inverse]	$(p\neq 0 \text{ and } q\neq 0)\Rightarrow(pq\neq 0)$
$\sim q\Rightarrow\sim p$	[contrapositive]	$(pq\neq 0)\Rightarrow(p\neq 0 \text{ and } q\neq 0)$

Binary operators-

p	q	p+q = p∨q	p·q = p∧q	p⇒q	p⇔q	p⊕q	p↑q	p↓q
0	0	0	0	1	1	0	1	1
0	1	1	0	1	0	1	1	0
1	0	1	0	0	0	1	1	0
1	1	1	1	1	1	0	0	0

if two prepositions are equivalent (X, Y)

then $[X\Leftrightarrow Y \equiv 1]$ $[X\equiv Y \text{ iff } X\Leftrightarrow Y=1]$

Ex Let $b\Leftrightarrow c$ and $a\Leftrightarrow(b\vee\sim b)$ is tautology what can be inferred about $a\vee(b\wedge c)$?

$b\Leftrightarrow c \Rightarrow b\equiv c$

$a\Leftrightarrow(b\vee\sim b)\Rightarrow a=1$

$\therefore a\vee(b\wedge c) = a\vee(b\wedge b) = 1\vee b = 1$ (Tautology)

Boolean Algebra: $(S,+,\cdot,')$ [Logic, Digital Logic, Set theory]

$(S,\vee,\wedge,\sim)$

$(S,\cup,\cap,A^c)$

- No of elements in set of Boolean Algebra must be in power of 2.
- Q_n is a boolean Algebra.

- $a-b = a\wedge b'$
- $A-(B\cup C) = (A-B)\cup(A-C)$ (Test T or F)
- $a-(b+c) = (a-b)+(a-c)$
- $ab'c' = ab'+ac'$ (false)

Properties of Operators- • operators are also known as logical connectives.

1. Closure- $\forall x, y$, $\begin{bmatrix} x+y \in S \\ x \cdot y \in S \\ \sim x \in S \end{bmatrix}$ OR $\begin{bmatrix} x \vee y \in S \\ x \wedge y \in S \\ \neg p \in S \end{bmatrix}$

$\forall, A, B \in S$ $\begin{bmatrix} A \cup B \in S \\ A \cap B \in S \\ A^c \in S \end{bmatrix}$

2. Commutative :

$\forall x, y \in S$ $\begin{bmatrix} x+y = y+x \\ x \cdot y = y \cdot x \end{bmatrix}$ $\forall A, B \in S$ $\begin{bmatrix} (A \cup B) = (B \cup A) \\ (B \cap A) = (A \cap B) \end{bmatrix}$

$\forall x, y \in S$ $\begin{bmatrix} x \wedge y = y \wedge x \\ x \vee y = y \vee x \end{bmatrix}$

3. Associative :

$\forall x, y, z \in S$ $\begin{bmatrix} x+(y+z) = (x+y)+z \\ x \cdot (y \cdot z) = (x \cdot y) \cdot z \end{bmatrix}$

$\forall x, y, z \in S$ $\begin{bmatrix} x \wedge (y \wedge z) = (x \wedge y) \wedge z \\ x \vee (y \vee z) = (x \vee y) \vee z \end{bmatrix}$

$\forall A, B, C \in S$ $\begin{bmatrix} A \cup (B \cup C) = (A \cup B) \cup C \\ (A \cap B) \cap C = A \cap (B \cap C) \end{bmatrix}$

4. Distributive : $\forall x, y, z \in S$ $\begin{bmatrix} x+(y \cdot z) = (x+y)(x+z) \\ x \cdot (y+z) = xy + xz \end{bmatrix}$

$\forall x, y, z \in S$ $\begin{bmatrix} x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z) \\ x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z) \end{bmatrix}$

$\forall A, B, C \in S$ $\begin{bmatrix} A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \\ A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \end{bmatrix}$

5. Identity : $\begin{bmatrix} \exists_0 \forall x \; x+0 = x = 0+x \\ \exists_1 \forall x \; x \cdot 1 = x = 1 \cdot x \end{bmatrix}$ & $0 \neq 1$

$\forall x \in S$ $\begin{bmatrix} x \wedge T = x = T \wedge x \\ x \vee F = x = F \vee x \end{bmatrix}$

$\exists \phi \; \forall A \in S$ $\begin{bmatrix} A \cup \phi = A = \phi \cup A \\ A \cap S = A = S \cap A \end{bmatrix}$ [S → Universal Set]

9440509626

ENGINEERING MATHEMATICS.

— Dinesh Sir.

Trapti Singh

LINEAR ALGEBRA [MATRICES]

- **Properties of Determinant**

1). If 2 rows/columns of a matrix are identical, then their determinant is zero.

$$\Delta = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 0 & 1 & 2 \end{bmatrix} = 0$$

2). If 2 rows/columns of a matrix are interchanged, the ~~range~~ sign of determinant is changed.

$$\Delta = \begin{vmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{vmatrix} \quad \text{then} \quad \begin{vmatrix} 3 & 4 & 5 \\ 0 & 1 & 2 \\ 6 & 7 & 8 \end{vmatrix} = -\Delta$$

3). If 3 rows/columns of a matrix are interchanged, then the sign of determinant is unaltered.

$$\Delta = \begin{vmatrix} 3 & 4 & 5 \\ 6 & 7 & 8 \\ 0 & 1 & 2 \end{vmatrix}$$

4). In the determinant of a matrix, if any column containing the sum or difference of 2 elements, then it can be split into sum or difference of two determinants.

$$\begin{vmatrix} a & a^2 & a^3+1 \\ b & b^2 & b^3+1 \\ c & c^2 & c^3+1 \end{vmatrix} = \begin{vmatrix} a & a^2 & a^3 \\ b & b^2 & b^3 \\ c & c^2 & c^3 \end{vmatrix} + \begin{vmatrix} a & a^2 & 1 \\ b & b^2 & 1 \\ c & c^2 & 1 \end{vmatrix}$$

5). Determinant of :

$$\boxed{|kA| = k^n |A|}$$

where $k \Rightarrow$ scalar

$A \Rightarrow$ matrix of order $n \times n$

6). $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \Longrightarrow \Delta = ad - bc$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \Longrightarrow \Delta = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}.$$

7). Lower Triangular Matrix : If all elements above the principal diagonal are 0, then it is said to L·T·M.

$$L\cdot T\cdot M\ A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 4 & 5 & 6 \end{bmatrix}$$

8). Upper Triangular Matrix : If all elements below the principal diagonal are 0, then it is said to be U·T·M

$$U\cdot T\cdot M = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix}$$

Note: If a Matrix is either L·T·M or U·T·M, then determinant is the product of principal diagonal elements.

$$\Delta(L\cdot T\cdot M) = 1*8*6 = 18$$

$$\Delta(U\cdot T\cdot M) = 1*4*6 = 24$$

Ques. Find the determinant of the matrix.

$$\begin{vmatrix} 1 & a & a^2 \\ 1 & b & b^2 \\ 1 & c & c^2 \end{vmatrix} = \begin{vmatrix} 0 & a-b & a^2-b^2 \\ 0 & b-c & b^2-c^2 \\ 1 & c & c^2 \end{vmatrix} = (a-b)(b-c)\begin{vmatrix} 0 & 1 & a+b \\ 0 & 1 & b+c \\ 1 & c & c^2 \end{vmatrix}$$

Minimum (n-1) operations can be performed only.

$R_1 \to R_1 - R_2$

$R_2 \to R_2 - R_3$

$= (a-b)(b-c)(b+c-a-b)$

$= (a-b)(b-c)(c-a)$

$$= \begin{vmatrix} 1 & 1 & 1 \\ a & b & c \\ a^2 & b^2 & c^2 \end{vmatrix}$$

Note: $\boxed{|A| = |A^T|}$

Ques. Find the determinant of :

$$\begin{vmatrix} 1 & a & b+c \\ 1 & b & c+a \\ 1 & c & a+b \end{vmatrix} = \begin{vmatrix} 1 & a & a+b+c \\ 1 & b & a+b+c \\ 1 & c & a+b+c \end{vmatrix} = (a+b+c)\begin{vmatrix} 1 & a & 1 \\ 1 & b & 1 \\ 1 & c & 1 \end{vmatrix} = \underline{0}.$$

$C_3 \to C_3 + C_2$

Ques. Find the determinant of :

$$\begin{vmatrix} \frac{1}{a} & a & bc \\ \frac{1}{b} & b & ca \\ \frac{1}{c} & c & ab \end{vmatrix} = \frac{1}{abc}\begin{vmatrix} bc & a & bc \\ ca & b & ca \\ ab & c & ab \end{vmatrix} = 0$$

Ques. Find determinant of :

$$\begin{vmatrix} 1 & 1 & 1 \\ 1 & 1+a & 1 \\ 1 & 1 & 1+b \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 \\ 0 & a & 0 \\ 0 & 0 & b \end{vmatrix} = \underline{ab}.$$

$R_2 - R_1$, $R_3 - R_1$

$$\therefore \begin{vmatrix} 1 & 1 & 1 \\ 1 & 5 & 1 \\ 1 & 1 & 6 \end{vmatrix} = 4*5 = \underline{20}.$$

Ques. Find the determinant of :

$$\begin{vmatrix} 1+a & 1 & 1 \\ 1 & 1+b & 1 \\ 1 & 1 & 1+c \end{vmatrix} = abc \begin{vmatrix} 1+\frac{1}{a} & \frac{1}{a} & \frac{1}{a} \\ \frac{1}{b} & \frac{1}{b}+1 & \frac{1}{b} \\ \frac{1}{c} & \frac{1}{c} & 1+\frac{1}{c} \end{vmatrix}$$

~~abc~~

$$= abc\left(1+\frac{1}{a}+\frac{1}{b}+\frac{1}{c}\right)\begin{vmatrix} 1 & 1 & 1 \\ \frac{1}{b} & \frac{1}{b}+1 & \frac{1}{b} \\ \frac{1}{c} & \frac{1}{c} & 1+\frac{1}{c} \end{vmatrix}$$

$$= abc\left(1+\frac{1}{a}+\frac{1}{b}+\frac{1}{c}\right)\begin{vmatrix} 1 & 0 & 0 \\ \frac{1}{b} & 1 & 0 \\ \frac{1}{c} & 0 & 1 \end{vmatrix}$$

$$= abc\left(1+\frac{1}{a}+\frac{1}{b}+\frac{1}{c}\right)$$

$$\therefore \begin{vmatrix} 1+a & 1 & 1 & 1 \\ 1 & 1+b & 1 & 1 \\ 1 & 1 & 1+c & 1 \\ 1 & 1 & 1 & 1+d \end{vmatrix} = abcd\left(1+\frac{1}{a}+\frac{1}{b}+\frac{1}{c}+\frac{1}{d}\right)$$

also. $\begin{vmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{vmatrix} = 5.$

Ques. Find the determinant of:

$$\begin{vmatrix} a & a^2 & a^3+1 \\ b & b^2 & b^3+1 \\ c & c^2 & c^3+1 \end{vmatrix} = \begin{vmatrix} a & a^2 & a^3 \\ b & b^2 & b^3 \\ c & c^2 & c^3 \end{vmatrix} + \begin{vmatrix} a & a^2 & 1 \\ b & b^2 & 1 \\ c & c^2 & 1 \end{vmatrix}$$

$$= abc \begin{vmatrix} 1 & a & a^2 \\ 1 & b & b^2 \\ 1 & c & c^2 \end{vmatrix} + 1 \begin{vmatrix} 1 & a & a^2 \\ 1 & b & b^2 \\ 1 & c & c^2 \end{vmatrix}$$

$$= (abc+1)(a-b)(b-c)(c-a).$$

Ques If a, b, c are all different and Non-zero.

If $\begin{vmatrix} a & a^2 & a^3+1 \\ b & b^2 & b^3+1 \\ c & c^2 & c^3+1 \end{vmatrix} = 0$, then $abc = ?$

here $(abc+1) = 0$

$abc = -1$

Ques. Determinant of the matrix is:

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 3 & 1 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 0 & -5 & -8 \end{bmatrix}$$

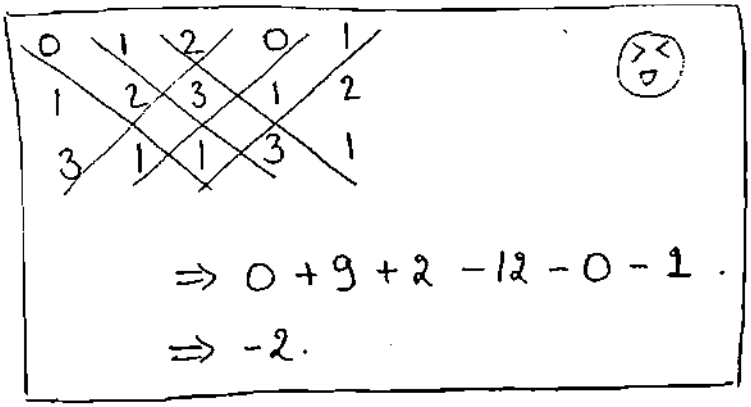
$R_3 - 3R_2$

$\Rightarrow -1[-8+10]$

$\Rightarrow -2.$

[Also by formula. it gd the same ans.]

Trick.



$\Rightarrow 0 + 9 + 2 - 12 - 0 - 1.$

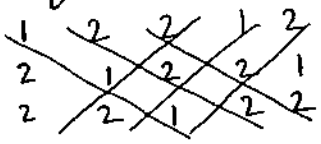
$\Rightarrow -2.$

Jitna bhi slow karo 8 min lagega.

Maggi Taiyaar.

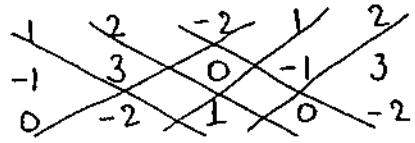
Ques. Find the determinant of.

(a) $\begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}$



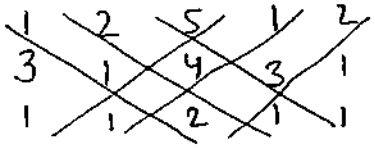
$1 + 8 + 8 - 4 - 4 - 4 \Rightarrow \underline{5}$

(b) $\begin{bmatrix} 1 & 2 & -2 \\ -1 & 3 & 0 \\ 0 & -2 & 1 \end{bmatrix}$



$3 + 0 - 4 + 0 - 0 + 2 \Rightarrow \underline{+1}$.

(c) $\begin{bmatrix} 1 & 2 & 5 \\ 3 & 1 & 4 \\ 1 & 1 & 2 \end{bmatrix}$



$2 + 8 + 15 - 5 - 4 - 12 = \underline{4}$.

Note: This formula/Trick is applicable on only 3×3 Matrix.

★ INVERSE OF A MATRIX:

$$A^{-1} = \frac{\text{Adj } A}{\Delta}$$

Note:⇒ Adj A of 2×2 matrix can be find like.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Ques. Find inverse of the Matrix.

(a) $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ $\quad A^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

(b) $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $\quad A^{-1} = \frac{1}{4-6}\begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix}$

$= -\frac{1}{2}\begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix}$

(c) $B = \begin{bmatrix} 5 & 4 \\ 1 & 2 \end{bmatrix}$ $\quad B^{-1} = \frac{1}{6}\begin{bmatrix} 2 & -4 \\ -1 & 5 \end{bmatrix}$

(d) $A = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ $\quad A^{-1} = \frac{1}{1}\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$

(e) $B = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$ $\quad B^{-1} = \frac{1}{ab}\begin{bmatrix} b & 0 \\ 0 & a \end{bmatrix} = \begin{bmatrix} 1/a & 0 \\ 0 & 1/b \end{bmatrix}$

Operating Systems

Trapti Singh..

Teaching Schedule

I. Introduction and Background.

II. Process Management
- → process concept
- → CPU scheduling ✓
- → Synchronization ✓
- → Concurrent Programming.
- → Deadlocks
- → Threads.

III. Memory Management.
- → RAM Chip Implementation
- → Loading, Linking & Address Binding
- → Techniques
  - paging
  - Multilevel paging.
  - Inverted paging
  - Segmentation
  - Segmented Paging.
- → Virtual Memory.

IV. File Systems.

Textbooks

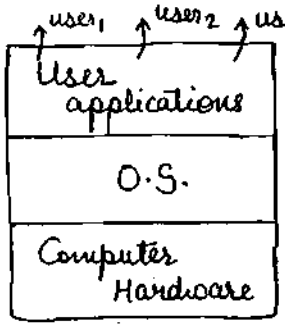
1. OS by Galvin.
2. Modern OS by A.S. Tenenbaum.
3. OS by William Stallings.

Chapter 1

Introduction and Background

Q. What is an OS?

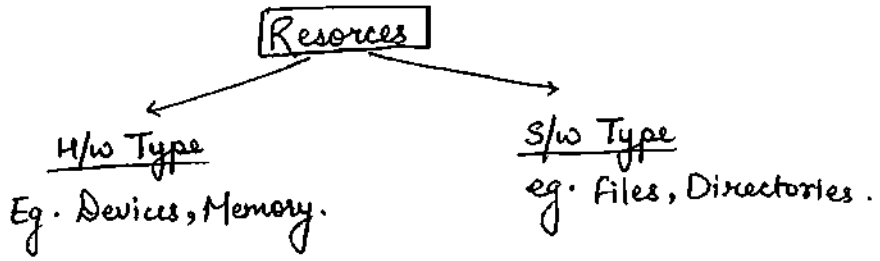
OS is an "Interface between user and computer hardware".




```
main()
{
  int x;
  printf("Hello");
}
```

→ internally calls write() System Call inorder to communicate with the monitor.

- System Call: System call is the request made by the user program to the OS in order to get any kind of service.
- Operating System is also called as Resorce Allocator because it is responsible for allocating resorces of a computer.



Goals of O.S.

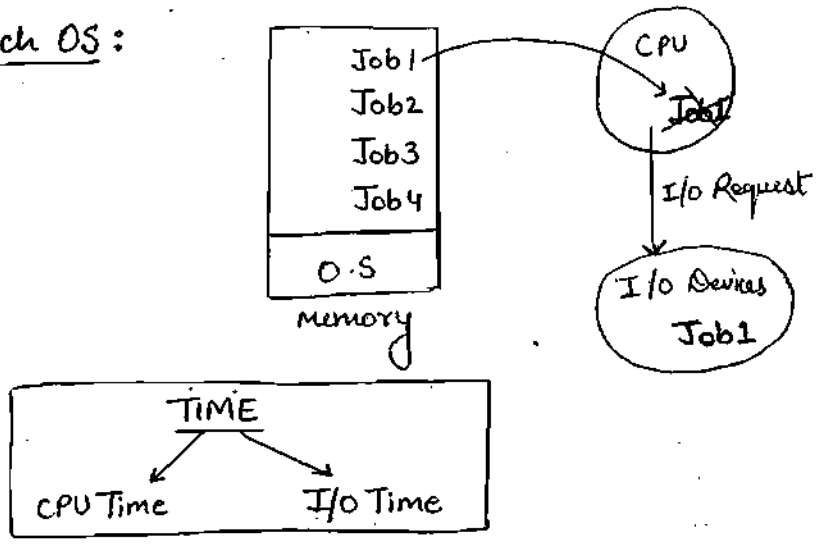
1. The primary goal is convenience. (easy to use)
2. The secondary goal is efficiency. (Stability).

~~Types of O.S~~

1).

Types of OS

(1). The Batch OS:

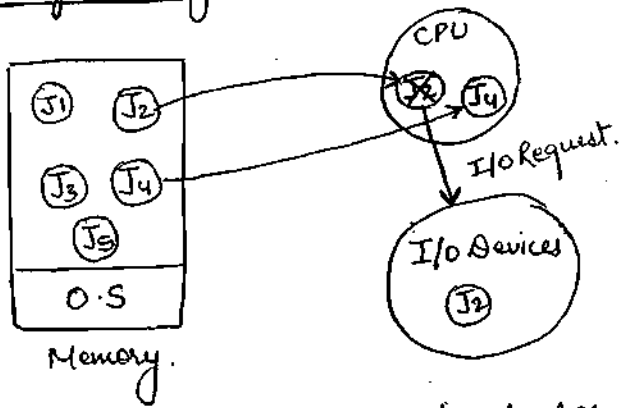


- If the Job is completed completely then only another Job will be scheduled onto CPU.
- increased CPU idleness.
- Decreased throughput of the system.

Throughput: No. of jobs completed per unit time is called throughput of the system.

Exp: IBM OS/2

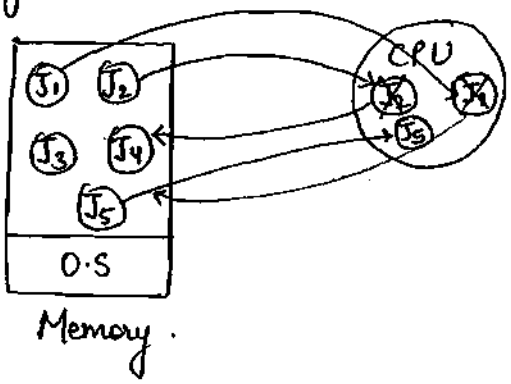
(2). Multiprogramming O.S.:



- If the job is leaving the CPU to perform IO operation, then another job which is ready for execution will be scheduled onto CPU.
- Advantage
  - Increased CPU Utilization.
  - Increased throughput of the system.

Exp: Windows, UNIX.

(3). Multitasking OS :

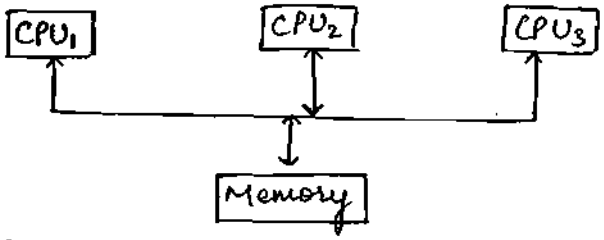


2ms
Time Stamp.

- Multitasking is an extension of multiprogramming OS.
- The jobs will be executed in the time sharing mode.

Exp: Windows, Unix

(4). Multiprocessor Systems :

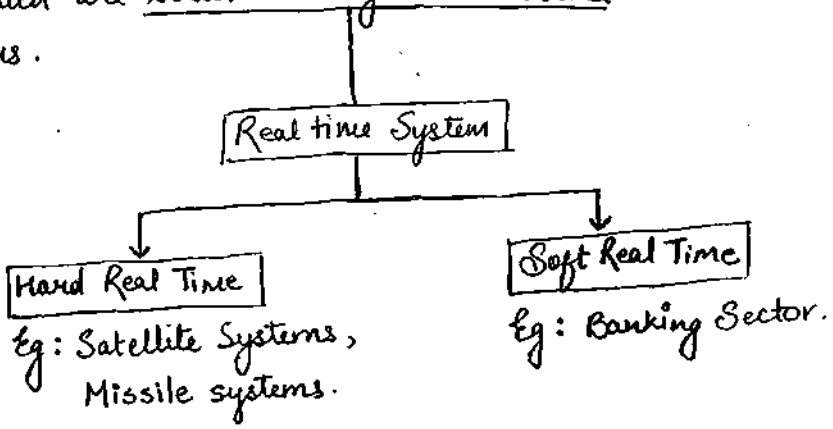


- Advantage
  - Increase the throughput of the system
  - Reliability
    - → Fault Tolerant Systems.
  - Economical.

Exp: UNIX.

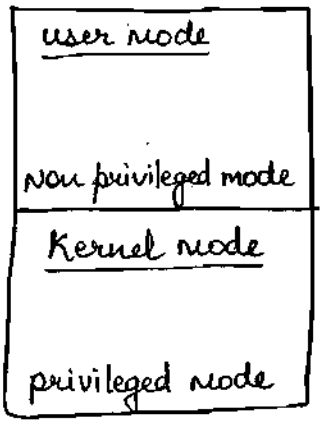
(5). Real Time Systems :

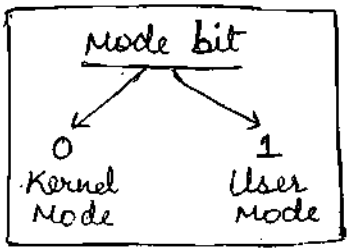
- The systems which are strict deadly time bound are called as real time systems.



Exp: Sx works, Vx works, RTO's.

Dual Mode Operation:





- In the hardware Level, the instructions are executed by using dual mode operation like
  1. user mode / Non privileged mode
  2. Kernel mode / priviledged mode / system mode / monitor mode.
- The dual mode operation is used in order to provide protection & security to the user programs. and also to the operating system from "errant users" (unauthorized users).
- It is purely the decision of the operating system in which particular mode, the instruction has to be executed.
- The mode bit is used to identify in which particular mode, the current instruction is executing.
- The privilised instructions are executed in the kernel mode & Non privilised instructions are executed in the user mode.
- In the Boot time, the system always starts only in the Kernel mode.
- The operating system always runs only in the kernel mode

Note: The mode switching takes very less time compared to process switching.

Array :

(Ex1) int a[10] = {10, 20, 30, 40, 50 ..... 100}

(declaration)

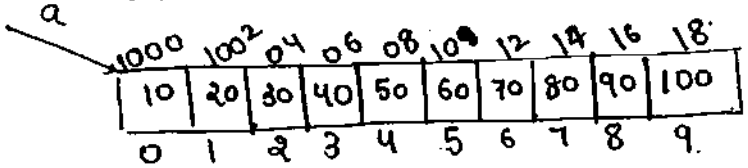
↓ identifier / variable / ph name.

iden

→ scan from left due to LL or LR parser (Top down, Bottom up)

int a [10] — a is array of (size 10) having elements as integer.

winner



a is array which contain 10 elements where everyone is integer

integer = 2 Bytes

j - i → gives elements before j

j - i + 1 → gives elements including j

(Bracket) ( ) have Left to Right associative. (If come first it will be done).

X { (int a) [10] — stop. a is a Integer

print a : 1000 will print (array name print gives Base address will print)

array name.

print variable name. : print the value of that value

$$\boxed{LOC(a[3]) = 1000 + (5-0) * 2 = 1010}$$

for LOC → (&) reference.

* Dereference

$$\boxed{LOC(a[9]) = 1000 + (9-0) * 2 = 1018}$$

using this formula anyone can be accessed

⇓

Random access.

EX2 A[75 ..... 330]

starting index → 75 ; ending index → 330

330 − 75 + 1 = 276 elements

BA = 1000 , C = 10 (size)

[a[290]] = 1000 + ~~290*10~~ ..... (290 − 75)

1000 + ~~2900~~ ..... 215 * 10

(a[290]) = ~~3900~~ (3150) .

2150
1000
3150

EX3 A[-90 ... 500] (591)

500 − (−90) + 1 ⇒ 591

BA = 0 , C = 5 Bytes

Loc(A[393]) = 0 + (92 − 393) * 5

= 0 + (393 + 90) * 5

= 2415

393
92
485
*5
2 5

NOTE

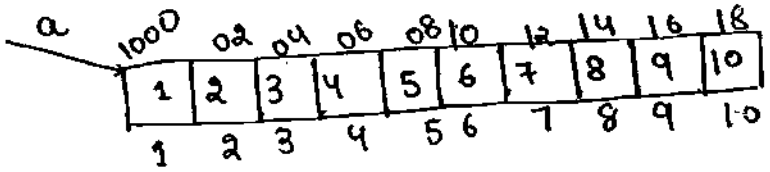
A[lb .... ub]

BA , C.

$$Loc[A[i]] = BA + (i - lb) * C$$

Loc(a[3]) = 1000 + (5 − 1) * 2 = 1008

extra.



NOTE.

By default array index start from 0 not from 1 because no need to calculate offset value (substraction) (No need to perform extra substraction)

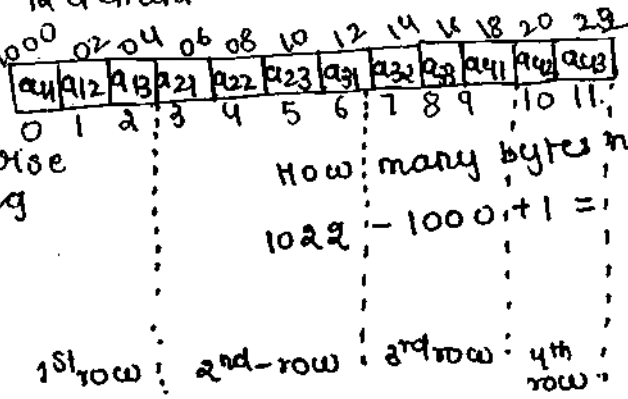
2D-Array:

int A[1.....4, 1.....3] → 4-1+1 = 4 rows; 3-1+1 = 3 columns.

	1	2	3
1	a_{11}	a_{12}	a_{13}
2	a_{21}	a_{22}	a_{23}
3	a_{31}	a_{32}	a_{33}
4	a_{41}	a_{42}	a_{43}

Row major order

4 × 3 = 12 elements = 12 slots needed to store



$LOC(A[4][3]) = 1000 + 3*4(2)$

$= 1000 + 12*2$

$= 1024$

(4-1) (3-1)
3 * 2 * 2
6 * 2
= 12

(3-1) * 4 (2)
2 * 8

$LOC[A[4][3] = 1000 + [(4-1)*3 + (3-1)]*2$

$= 1022$

Ans $LOC(a[2][3]) = 1000 + [(2-1)*3 + (3-1)]*2$

$= 1000 + [3 + 2]*2$

$1000 + 10$

$= 1010$

(3) * 3
9 + 2
11 * 2
1000 + 22
= 1022

(j-i) * j +

ex(2) A = [29 .... 76, 93 ... 200]

row ⇒ 76-29+1 = 48 ⇒ 108 c.

BA = 1000, c = 10 Row major order

$LOC(A[70][190]) = 1000 + [(70-29)*108 + (190-93)]*10$

$= 1000 + [4428 + 20]*10$

$= 1000 + 44480$

$= 45480.$

76 − 29 = 47 ; 200 − 93 = 107 ; 190 − 93

70 − 29 = 41 ; 108 × 41 = 108, 432, 4428

44480 + 1000 = 454

40 250

ex(3) $A[-200 \cdots +200, -300 \cdots -150]$

$200-(-200)+1$ $\quad -150-(-300)+1$

$-150+300+1$

$150+1$

$=151$

BA = 0, C = 1, RMO

$LOC[A[-3][-170]] = D + [(-3-(-200))*151 + [-170-(-300)]]$

$= 197*151 + 130$

$= 29747 + 130$

$= 29877$

200 − 3 = 197

300 − 5 = 295

300 − 170 = 130

197 × 151 = 29747

NOTE: $ub_1 - lb_1 + 1 = nr$

$A[lb_1 \cdots ub_1, lb_2 \cdots ub_2]$ nc

BA, C, RMO

$LOC(A[i][j])$

$= BA + [(i-lb_1)*nc + (j-lb_2)]*C$

NOTE:

column Major order = 48 = 108

ex(4) $A[29 \cdots 76, 93 \cdots 200]$

BA = 1000, C = 10 CMO

$L(A[70][190]) = 1000 + [(190-93)*48 + (70-29)]*10$

$= 1000 + [87*48 + 41]*10$

$= 47970$

190 − 93 = 87

70 − 29 = 41

87 × 48

last element address in RMO = last elem add in CMO

$A = [-200 \cdots +200 \;(\Rightarrow 401),\; -300 \cdots -150 \;(\Rightarrow 151)]$

BA = 0 c = 1 cmo

$Loc(A[-3][-170]) = 0 + [(-170+300)\times 401 + (-3+200)]\times 1$

$= 130 \times 401 + 197$

$= 52327$

300 − 170 = 130

401 − 13 = 388

130×401:
1203
401 x
52130
197
52327

NOTE: $A(lb_1 \cdots ub_1 \; lb_2 \cdots ub_2)$ BA c nr = ?

$Loc(A[i][j]) = BA + [(j - lb_1) * nr + (i - lb_1)]c$ cmo

3D-array

Ex①

$A[23 \cdots 49,\; 2 \cdots 19,\; 11 \cdots 29]$

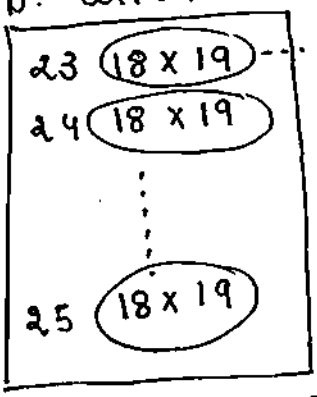
no of 2D = 49 − 23 + 1 = 26 + 1 = 27

$n_1 = 19 - 2 + 1 = 17 + 1 = 18$

$n_2 = 29 - 11 + 1 = 18 + 1 = 19$

$18 \times 19 \Rightarrow$ 2D.

3D: collections of 2D.



① collects of elements ⇓ 1D

② collection of 1D ⇓ 2D

BA = 1000, c = 10, RMO

size of this 18 x 19.

$Loc(A[40][15][20]) = 1000 + [(40-23)*27 + (15-2)*19 + (20-11)]*10$

40 − 23 = 17

27 − 18 = 9

$= 61700$

Software is a program, when it is executed the associated functionally must be satisfied.

S/w is a data structure used to manipulate the information

S/w consist the operational procedure, used to indicate the operational guideline of the s/w....

ie... user manual, beginners guide, system overview... etc

Software consists the documentation manual, used to indicate the list of the activities maintained to develop the s/w....

Software consists the documentation manual, used to indicate the list of the activities maintained to develop the s/w

ie....

- Analysis Specification
  - Formal specification
  - Context Program
  - Data flow Diagram
- Design
  - Flow chart
  - ER Diagram

Code ——— Logical Inst

Test ——— Test Data
└ Test Result

S/W is a combination of Program, operational Procedure & Instruction Manuals, Finally we can conclude the S/W is a logical Component rather then Physical Component.

Software = Programs + operational + Documentation, Procedure Manuals

Character of Software-

1. S/W is developed or engineered but not manufactured in the classic sense.

i) S/W development doesn't have the assembly line.

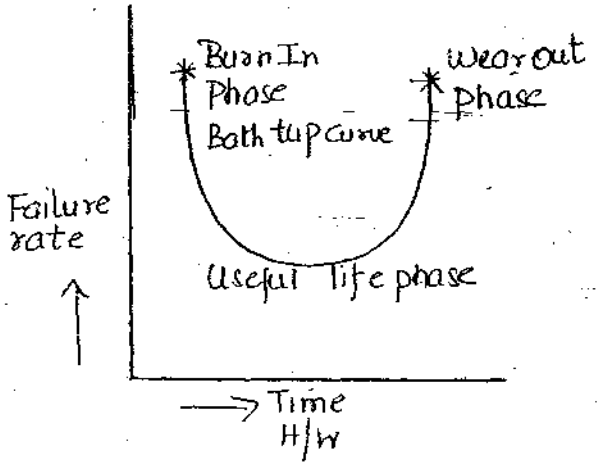
ii) Logic of the program is developed only once & that can be copied into any no. of the copies.

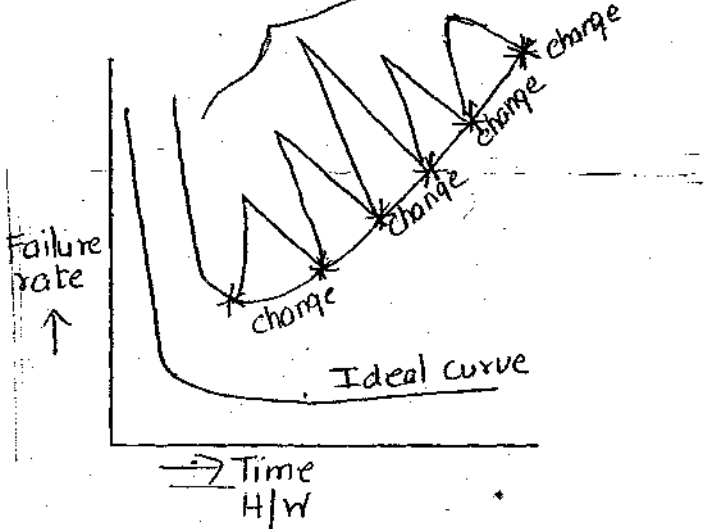
iii) Physical Components consists the assembly line to manufacture the product, during this process every time raw materials are included.

iv). After implementation of design document if it is generate the logical Component called as the development

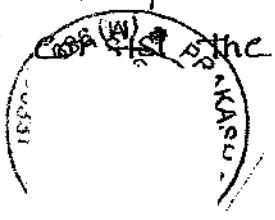
Process else that process is called as manufacture.

2. S/W doesn't WEAROUT but 'DETERIORATE' → Deterioration





The life cycle of H/W component consist of the 3 stages :-



Burn in Phase

In the early stage of development due to the more numbers of errors the product failure rate is high.

In the Burn-In-Phase the product is present in the developer's site.

After various test operations the failure rate is decrease & it will be established at one point.

Later deployee the product into customers place.

Useful life Phase

In this Phase the product is there in operational state.

Wear Out Phase

After the continuous usexs of the product over a perod of a time the life time of the product is decreases due to the environmental changes.

Ie... Temp, Vibrations, Dust Etc...

Therefore the failure rate of the product increases.

NOTE :-

When the H/W component is undergoes wear out Condition, then replace the component with new Component.

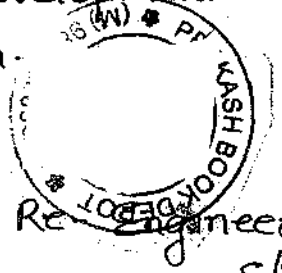
S/W is undergoes deterioration

Ie... At the early stage of operation the product failure rate is high.

After the testing operation the product will be deployed in to the customers site.

S/W doesn't affected by the environmental changes but the customer requirements are not static.

When the customer requirements are changed frequently then the maintenance cost of the S/W becomes twice or thrice than the development cost. This condition is called deterioration.



Note :-
When the S/W is deteriorate then Re-Engineer the S/W

3. Industry is moving towards component based development. Still the S/W is customer build.

Component is a reusable code or error free code or risk free code or fully tested code.

During the Physical component manufacturing process we can directly use the component without changes.

ie .... Working models contains different I.C to implement the model directly purchase the I.C from market.

In the S/W development process, supplication to application the functionality will be differ therefore the S/W components are undergoes customization according to project functionally.

In the S/W development 4 Types of components are used :-

THEORY OF COMPUTATION

- GODEL : Logic is limited
- Turing : Model for computation
- POST
- chomsky Heirarchy

→ 2 types of computer

Acceptors: Yes/NO ; given lang. Accept or Not Accept.

Transducer: computational
x is given f(x) can be computed.

- every problem has associated with language.. we bother about acceptance of Language. If we can accept the language - we can say problem is solvable.

chapter - 0

- Language : collection of strings
  grammar :
  Machine:

Language can be described by grammar.

- (L) Informal (can't list all the things in language)

(compact) generative. Formal. its kind of formula.

regular expression — only RE covered By only (RL). for RL we have

(L)

Machine Formal. cognitive device (recognize the lang.).

Grammar Formal generative device (More compleed) Here we have Bunch of Rules.

These two generate every grammar. (But NOT RE, upto RE) Because we dont have Machine.

1. Alphabet = $\Sigma = \{a, b\}$
2. string
3. concatenation
4. Reversal
5. Length of a string
6. NULL string. = "ϵ"
7. PREFIX
8. SUFFIX
9. substring
10. substring.
11. Powers of a string $(\omega)^n$.
12. Σ^*, Σ^+
13. $L \subseteq \Sigma^*$.
14. CHOMSKEY HEIRARCHY
14. ~~operat~~ Representations of Language

- Language

1. alphabet:

14. Representation of Language { 3 Formal, 3 Informal }

15. Operations on Language:
union, intersection, $\bar{L}$, $L_1 - L_2$, $L_1 \oplus L_2$

16. concatenation of Lang.

$$L_1 \cdot L_2 = \{uv \mid u \in L_1, \& v \in L_2\}$$

17. $L^R = \{u^R \mid u \in L\}$ Reversal of language.

18. L^*, L^+

L^* contain $\in$ every possible combination of strings.

L^+ Not contain $\in$

$L^+ = L^* - (\varepsilon)$ ⟶ This is Not correct statement.

• **Alphabet**: a Non-empty finite set of symbols.

$\Sigma = \{\}$ Not alphabet

• $\Sigma = \{a\}$, $\Sigma = \{1\}$, $\Sigma = \{2\}$ $\Sigma = \{3\}$ (1 symbol) unitary alphabet

• $\Sigma = \{a, b\}$, $\Sigma = \{1, 0\}$ binary alphabet (2 symbol).

$\Sigma = \{1, 11, 111 \ldots\}$ This is Not allowed, No of symbols should be finite.

$\{0, 1, 2\} = \{0, 1, 2\}$

both alphabet same, order dont matter.

symbol $\Sigma = \{\underline{01}, \underline{10}\}$.

compound symbol

$\Sigma = \{01, 10, (1), (0)\}$

This is Not valid symbol 01 or 10 cant break further.

• **String**: sequence of 0 or more finite symbols taken from the alphabet

sequence : order is important

$\Sigma = \{a, b\}$.

a^{100} is valid string ? ⇒ Yes.

aaa ... 100 times valid.

aabb(1) valid ? ⇒ No symbols can be taken from alphabet.

baab = baba Not equal string, 'sequence should be follow'

$(ab)^2 \neq a^2b^2$ Not valid in TOC.

$abab \neq aabb$

• **concatenation**:

if $u = 01$

$v = 100$

$uv = \underbrace{01}\underbrace{100}$ concatenation

Here $uv \neq vu$.

for all (u, v)

where $u = 00$

$v = 000$.

$uv = vu$ True Here

so Not for all (u, v); $uv \neq vu$.

• it is associative

$u(vw) = (uv)w$

• Not communicative.

• The length of $u \cdot v$ will always be equal to $(u + v)$.

$u = 01 = 2$

$v = 100 = 3$

$uv = 01100 \Rightarrow$ length is $(2+3) = 5$

• The length of u^i

$u = 100$ $u^R = 001 \Rightarrow u^R \neq u$

for all u and v $uv \neq vu$ False

• $v = v^R$ iff v is a palindrome

Palindrome
- even (because length is integer.) : EP
- odd. : OP.

Palindrome Language = $\{ \underset{EP}{ww^R} \cup \underset{OP}{w x w^R} \mid w \in \Sigma^*, x \in \Sigma \}$ (x is 1 bit)

$\Downarrow$ 1001 $\quad$ $\Downarrow$ 10001

$\{ w \in \Sigma^* \mid w = w^{\overline{x}} \}$

• Properties of Reversal

$(v^R)^R = v$
(Reversal of Reversal).

$(u \cdot v)^R = v^R u^R$

$(xyz)^R = z^R y^R x^R$.

• Length :
No of symbols present in strings.

if $\Sigma = \{0,1,2\}$.

→ How many 5 length string possible. if $\Sigma = \{a,b\}$.

0 length → 1 (i.e ε)

1 length → a, b → $|\Sigma| = 2$

2 length → ab, ba, aa, bb → $|\Sigma|^2$. (cardinality of Σ.)

⋮

n length → $|\Sigma|^n$.

So of length '3' 2^3 ⇒ for $\Sigma = \{a,b,c\} \Rightarrow |\Sigma| = 3$
length '5' 2^5

ww^R How many even palindrome of length 10

$\Sigma = \{0,1\}$

$\underline{2}\ \underline{2}\ \underline{2}\ \underline{2}\ \underline{2} \mid - - - - -$

(25)

even palindrome $= |\Sigma|^{n/2}$

- odd ⇒ length 13. $\Sigma = \{0,1\}$
  check even → 12.

$|\Sigma|^{\frac{12}{2}}$ $\quad |2|^6 \Rightarrow 2^6 \times 2$ odd palindrome.

upto length 10, even palindrome. $\Sigma \{0,1\}$

0	length ⟶	$2^{0/2} = 1$
2	"	$2^{2/2} = 2$
4	"	$2^{4/2} = 4$
8	"	$2^{8/2} = 2^4$
10	"	$2^{10/2}$

- **NULL string:**
  it is only string of length '0'

lang with Null string. $|\{\varepsilon\}| \rightarrow$ cardinality.

$|\varepsilon| = 0$.

$|\{\}| = 0$.
empty language.

- language has cardinality.
  where
  Null Reverse is Reverse
  Null is identity element for concatenation.

- $(\Sigma^*, \cdot, \emptyset)$ ← groupoid, semi-group.
  ↓ operator
  Algebra.

- $|\{0,1\}|^5 \rightarrow |\{0,1\}|$
  How many fn possible
  all string of length 5.

$|\Sigma|^5 \rightarrow \Sigma$ possible function.

cardina. $= 2^5$ $\qquad 2$.

$|B|^{|A|}$ function possible $= \underline{\underline{2^{(2^5)}}}$