# Math related subject in CS

Around

- → Discrete math ✓
- → Operating system ✓
- → Digital logic design ✓
- → Software engineering ✓
- → Computer network ✓
- → Data communication
- → Computer architecture
- → VLSI
- → Artificial Intelligence
- → Data mining / Machine learning
- → Computer graphics & animations

Ismail

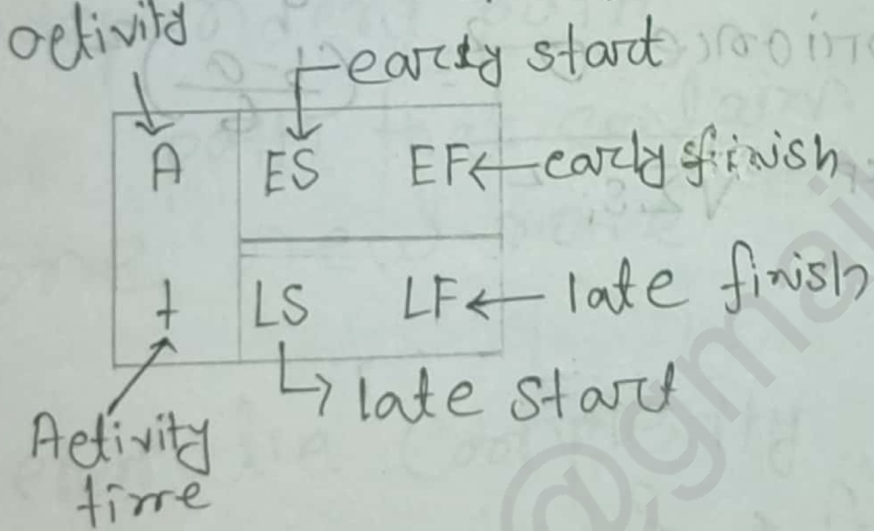
# Software Engineering

→ Estimation

→ S/W scheduling
  ↳ PERT
  ↳ CPM

## S/W scheduling

CPM → critical path method




1. $EF = FS + t$

2. $ES_j = max(E + t)$

3. $LS = LF - t$

4. $LF_j = min(LF_i - t)$

5. $Slack = LS - ES$
   $= LF - EF$

Slack = 0 critical activity

* slack is also called float

PERT → Project evaluation & review technique

6. time estimation or mean : $t_e = \frac{a+4m+b}{6}$

7. derivation $S_i = \frac{b-a}{6}$

8. variance $\sigma^2 = \left(\frac{b-a}{6}\right)^2$

9. $S_{cp} = \sqrt{\Sigma S_i^{\,2}}$

## Cyclomatic Complexity

Cyclomatic Complexity → from a $\boxed{\text{flow graph}}$ ← $\boxed{\text{Code}}$

edge node

$V(G) = E - N + 2$

$V(G) = P + 1$ → predicate node

$V(G) = \#$ Region

**Independent path:** path that contains at least one new node

Cyclomatic Complexity → from a $\boxed{\text{graph matrix}}$ → $\boxed{\text{Connection matrix}}$

Estimation

Line of Code (LOC)

1. Estimation Variable, $EV = \left( \frac{S_{opt} + 4S_m + S_{pess}}{6} \right)$

$S_{opt}$: optimistic estimation

$4S_m$: most likely ;;

$S_{pess}$: pessimistic ;;

2. productivity = KLOC / person-month

3. Quality = Defects / KLOC

4. Cost = $/LOC

5. Documentation = page of documentation / KLOC

KLOC: kilo line of code

defects: total number of error discovered

# Functional Point (FP) estimation

1. $FP = \text{Count-total} \times [0.65 + 0.01 \times \text{Sum}(F_i)]$

summation of (information domain × count)

* # user inputs
* # " outputs
* # user inquiries
* # files
* # external interface.

complexity adjustment based on response to question (1 - 14)

each question value on scale of 0 to 5

2. Productivity = FP / person-month

3. Quality = defects / FP

4. Cost = \$ / FP

5. Documentation = pages of documentation / FP

## ⊞ COCOMO model / Constructive Cost model:

COCOMO applies to 3 classes of S/W

| | | a | b | c | d |
|---|---|---|---|---|---|
| → organic projects | 3.6 | 2.4 | 1.05 | 2.5 | 0.38 |
| → semi-detached " | 3.0 | 3.0 | 1.12 | 2.5 | 0.35 |
| → Embedded projects | 2.4 | 3.6 | 1.20 | 2.5 | 0.32 |

basic COCOMO

1. $E = a \times (KLOC)^b$ (person-month)

2. $D = c \times (E)^d$ (months)

3. People required, $P = \frac{\text{Effort applied}}{\text{Development Time}}$ (count)

Development Time (D)

Effort Applied (E)

intermediate COCOMO

1. Effort applied, $E = a(KLOC)^b \cdot EAF$

2. Development time, $D = c * (E)^d$

EAF: Effort adjustment factor

dynamic multivariable model

## Software Equation

$$E = \left[LOC \times B^{0.33}/P\right]^3 \times (1/t^4)$$

Where,

E = effort in person-month

t = projected duration in months or year

P = "productivity parameter"

B = Special skills factor

P = 12000 for scientific

P = 2000 " real time embedded

P = 10000 " telecommunication & system s/w

P = 28000 " business s/w

minimum development time,

$$t_{min} = 8.14\,(LOC/P)^{0.43}\ \text{months}$$

for $t_{min} > 6$ months

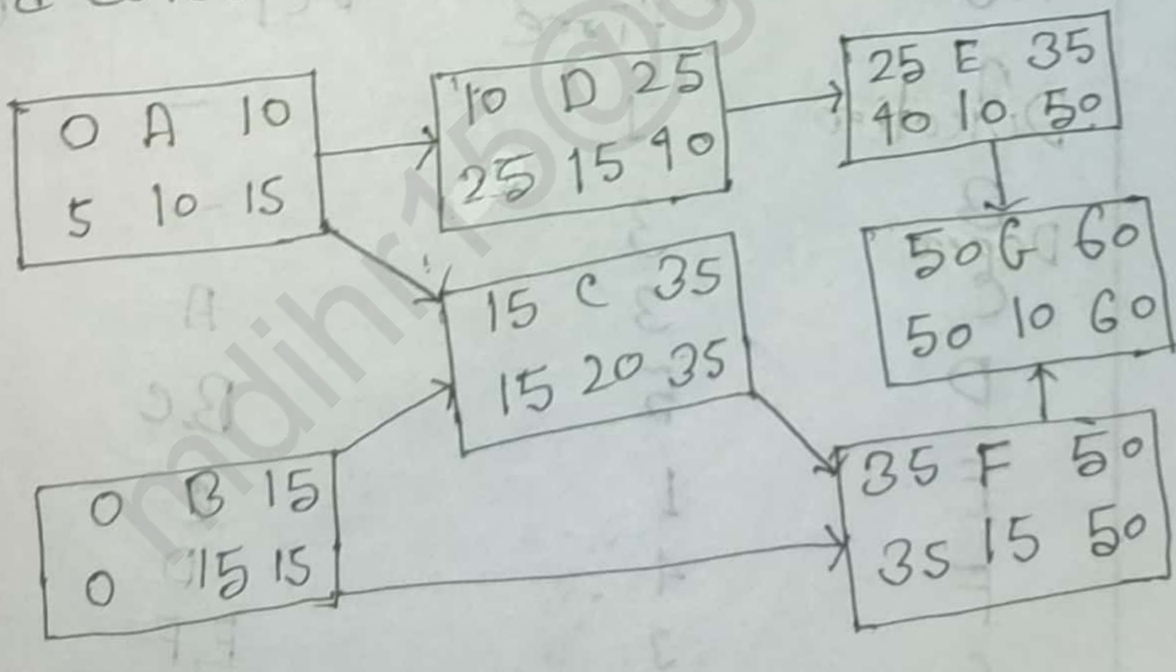
Effort, $E = 180\,B\,t^3$ person months,

for $E \geq 20$ person-months

## Example-1

| task | Predecessor | time |
|---|---|---|
| A | — | 10 |
| B | — | 15 |
| C | A, B | 20 |
| D | A | 15 |
| E | D | 10 |
| F | B, C | 15 |
| G | E, F | 10 |

From the above data find critical path and calculate slack time



Slack time of A, $St_A = 15-10=5$
" " of B, $St_B = 15-15=0$
" " of C, $St_C = 35-35=0$
" " of D, $St_D = 40-25=15$
" " of E, $St_E = 50-35=15$
" " of F, $St_F = 50-50=0$
" " of G, $St_G = 60-60=0$

∴ critical path is

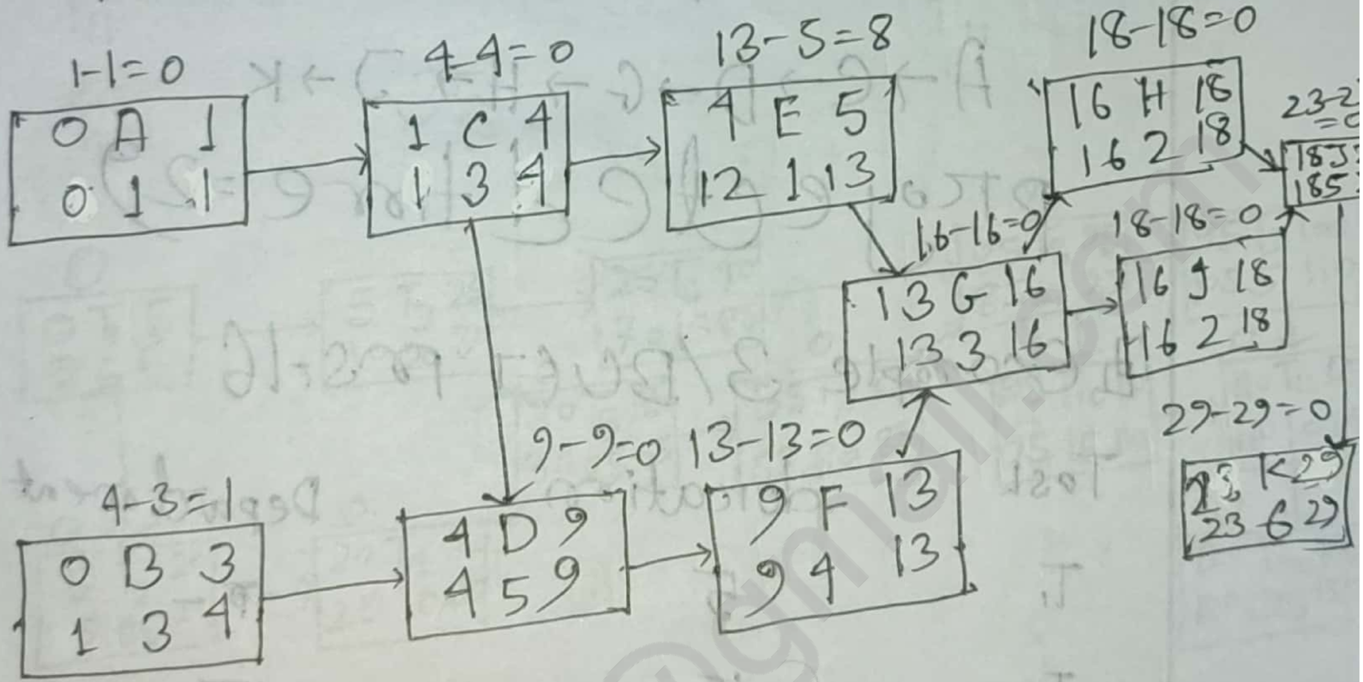
B → C → F → G

projected time = 60

Example-2 / BUET MS-18

| Task | Duration time | Dependency |
|---|---|---|
| A | 1 | |
| B | 3 | |
| C | 3 | A |
| D | 5 | B,C |
| E | 1 | C |
| F | 4 | D |
| G | 3 | E,F |

| task | duration | dependency |
|---|---|---|
| H | 2 | G |
| I | 2 | G |
| J | 5 | H, I |
| K | 6 | J |




Stack time of A, = 0

" " " B, = 1

" " " C, = 0

" " " D, = 0

" " " E, = 8

" " " F, = 0

" " " G, = 0

" " " H, = 0

Slack time of G = 0
" " of J = 0
" " of K = 0

∴ Critical path =

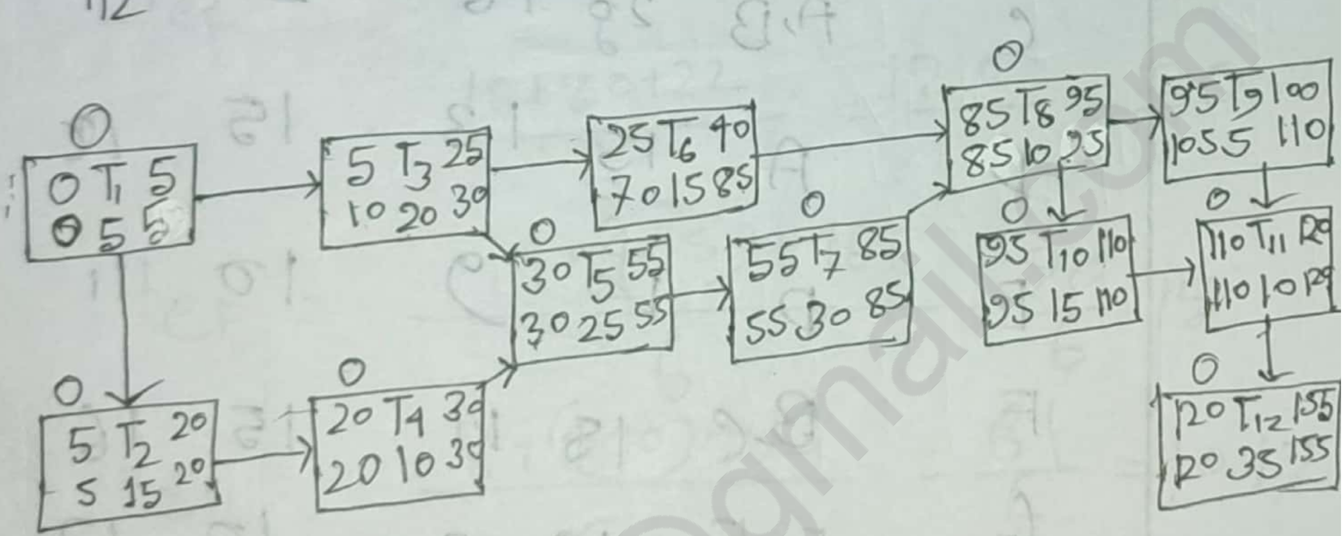
A → C → D → G → H → J → K

projected time = 29

Example 3/BUET MS-16

| Task | duration | Department |
|---|---|---|
| $T_1$ | 5 | – |
| $T_2$ | 15 | $T_1$ |
| $T_3$ | 20 | $T_1$ |
| $T_4$ | 10 | $T_2$ |
| $T_5$ | 25 | $T_3$, $T_4$ |
| $T_6$ | 15 | $T_3$ |
| $T_7$ | 30 | $T_5$ |

| Task | duration | department |
|---|---|---|
| $T_8$ | 10 | $T_6, T_7$ |
| $T_9$ | 5 | $T_8$ |
| $T_{10}$ | 15 | $T_8$ |
| $T_{11}$ | 10 | $T_9, T_{10}$ |
| $T_{12}$ | 35 | $T_{11}$ |




∴ critical path.

$T_1 \to T_2 \to T_4 \to T_5 \to T_7 \to T_8 \to T_{10} \to T_{11} \to T_{12}$

projected time : 155

Example - 4 :

| task | Predicessor | time | Duration a | m | b |
|---|---|---|---|---|---|
| A | — | 10 | 8 | 10 | 12 |
| B | — | 15 | 14 | 15 | 18 |
| C | A, B | 20 | 18 | 20 | 22 |
| D | A | 15 | 13 | 15 | 18 |
| E | D | 10 | 9 | 10 | 11 |
| F | B, C | 15 | 14 | 15 | 16 |
| G | E, F | 10 | 8 | 10 | 13 |

From the above data estimate time, deviation, find critical path

Soln Critical path finding like example 1

Critical path is

$$B \rightarrow C \rightarrow F \rightarrow G$$

$$t_{eB} = \frac{\cancel{8+}\, 14 + (4 \times 15) + 18}{6}$$

$$= \frac{14+60+18}{6} = \frac{94}{6} = 15.33$$

$$t_{eC} = \frac{18 + (4 \times 20) + 22}{6}$$

$$= \frac{18+80+22}{6} = \frac{120}{6} = 20$$

$$t_{eF} = \frac{14+(4 \times 15)+16}{6} = \frac{90}{6}\ 15$$

$$t_{eG} = \frac{8+(4 \times 10)+13}{6} = \frac{61}{6} = 10.17$$

$\cancel{60.67}$

60.5 Ans

time deviation of critical path

$$S_{iB} = \sqrt{\phantom{x}}\ \frac{18-14}{6} = \frac{4}{6} = 0.67$$

$$S_{iC} = \frac{22-18}{6} = \frac{4}{6} = 0.67$$

$$S_{IF} = \frac{16-14}{6} = \frac{2}{6} = 0.33$$

$$S_{iG} = \frac{13-8}{6} = \frac{5}{6} = 0.83$$

$$S_{CP} = \sqrt{(0.66)^2+(0.66)^2+(0.33)^2+(0.83)^2}$$

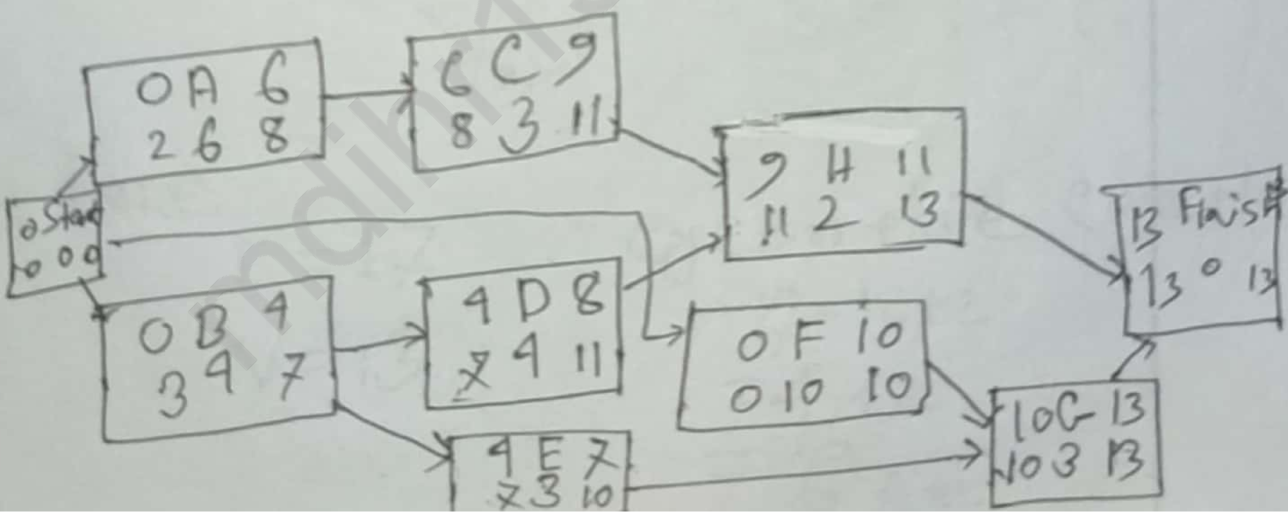
$$= \sqrt{1.669}$$

$$= 1.29$$

Ans

Example-5

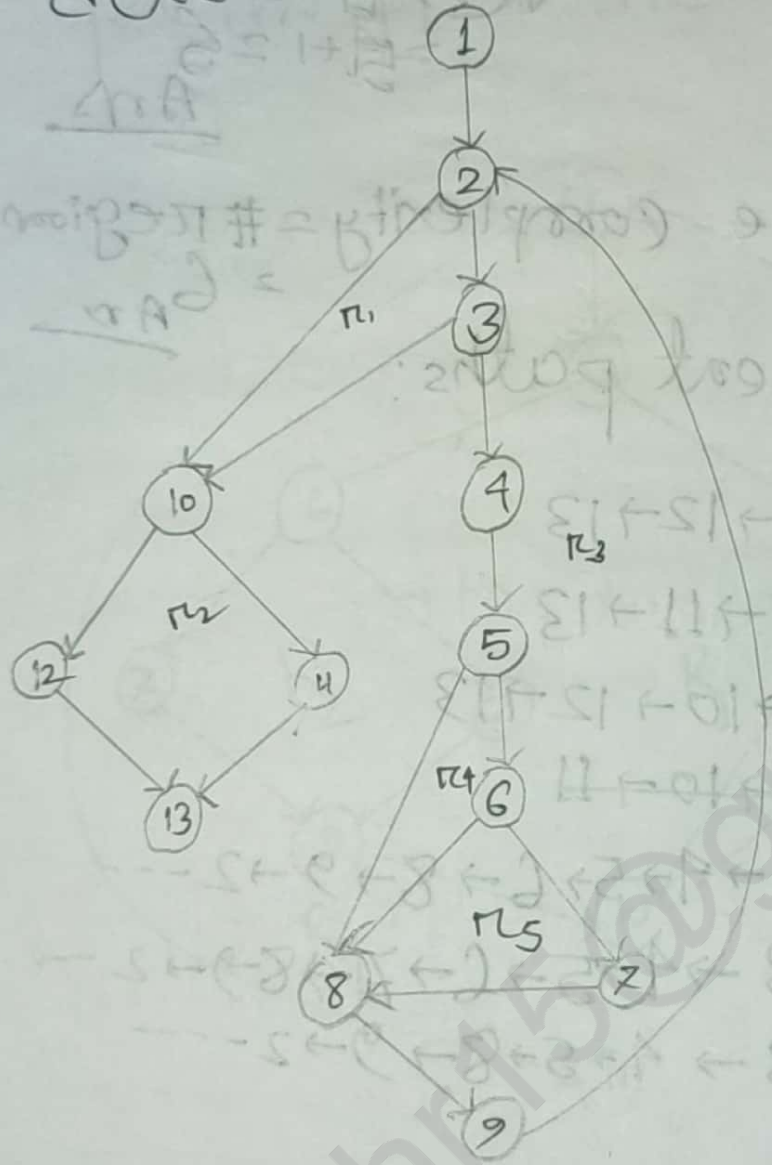
| Activity | Duration (in weeks) | Precedents |
|---|---|---|
| A | 6 | — |
| B | 4 | — |
| C | 3 | A |
| D | 4 | B |
| E | 3 | B |
| F | 10 | — |
| G | 3 | E, F |
| H | 2 | C, D |



∴ Critical path

Start → F → G → End

□ Find out the cyclomatic complexity from the following flow graph. also determine the independent path




here,

$E = 17$

$V = 13$

$\therefore$ cyclomatic complexity

$= 17 - 13 + 2$

$= 4 + 2$

$= 6$ Ans

here, predicate node, P = 5

∴ Cyclomatic complexity

$V(G) = P + 1$

$= 5 + 1 = 6$ Ans

Cyclomatic complexity = # region

= 6 Ans

Independent paths:

1 → 2 → 10 → 12 → 13

1 → 2 → 10 → 11 → 13

1 → 2 → 3 → 10 → 12 → 13

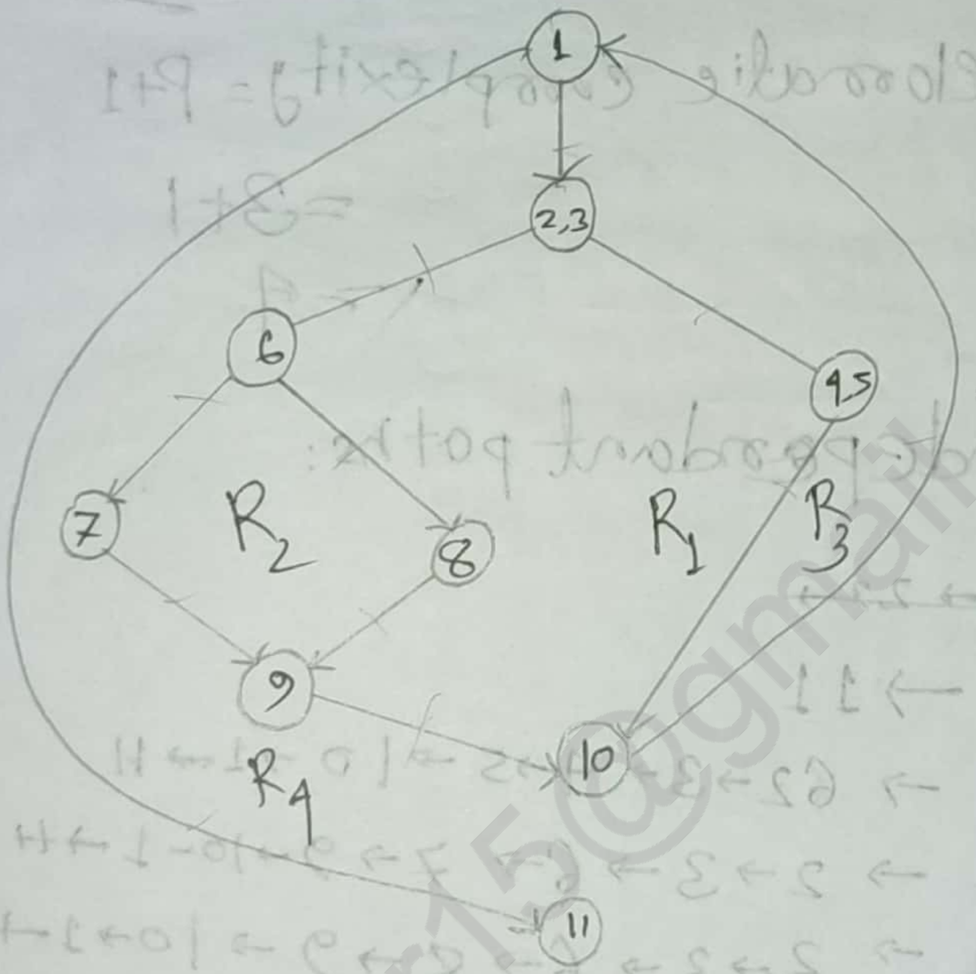
~~1 → 2 → 3 → 10 → 11~~

1 → 2 → 3 → 4 → 5 → 6 → 8 → 9 → 2 - - -

1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 2 —

1 → 2 → 3 → 4 → 5 → 8 → 9 → 2 - - -

☐ Determine the cyclomatic complexity and independent paths from the below graph.




Here,

# edges = 11

# Nodes = 9

∴ Cyclomatic complexity $V(G) = E - V + 2$

$= 11 - 9 + 2$
$= 2 + 2 = 4$ Ans

Cyclomatic complexity = # of region
$= 4$ Ans

Cyclomatic complexity = P+1
$= 3 + 1$
$= 4$

Independant paths:

~~1 → 2 3 →~~

1 → 11

1 → 2 → 3 → 4 → 5 → 10 → 1 → 11

1 → 2 → 3 → 6 → 7 → 9 → 10 – 1 → 11

1 → 2 → 3 → 6 → 8 → 9 → 10 → 1 – 11

4) From the following graph ~~make~~ a connection ~~m~~atrix and determine cyclomatic complexity.

⊞ Suppose you have to develop 1000 functions in java. each functions contains 600 lines of codes. Software development team has average experience on similar types of projects. Project schedule is not very tight. How would you calculate effort, development time, average staffsize and productivity of the project with COCOMO model.

Soln

Ans :

Line of Code $= (1000 \times 600)/1000$ KLOC

$= 600$ KLOC

Effort $= a \times (KLOC)^b$ person-month

$= 3.0 \times (600)^{1.12}$

$= 3878.38$

development time $= c \times (E)^d$

$= 2.5(3878.38)^{0.35}$

$\Rightarrow 45.078$ month

Average staff size $= E/D$ ~~person~~

$= \dfrac{3878.38}{45.078}$

$= 86.04$ person

productivity $= KLOC/E$ Ans

$= 600/3878.38$

$= 0.155$ Ans

# UML (Unified Modeling Language):

It is a visual diagramming language used to represent the design of systems or process

* way to visualize a system architecture blue prints in a diagram.

## Classification

2 types:

1. structure diagram
2. Behavior "

## Structural UML diagram:

→ class diagram
→ object "
→ package "
→ component

→ Composite structure diagram
→ Deployment diagram.

Behavioral UML diagram:

→ Activity diagram
→ Sequence "
→ Use case "
→ State "
→ Communication "
→ Interaction overview
→ Timing diagram.

→ white box modeling → black box modeling

| Structural | Behavioral |
|---|---|
| → visualize, specify construct the Static aspect of a system | → visualize, speify, construct the dynamic aspect of a system |
| → represents the framework for the system | → describes the interaction in the system |

## Use case diagram:

It is a diagrammatic representation of a use-case.
It summarizes the details of system's users (actor) and their interaction with the system.

## Components

Actor → users that interact with a system

Use case → Represents the system function (use case)

System → place actor outside the system boundaries

system name

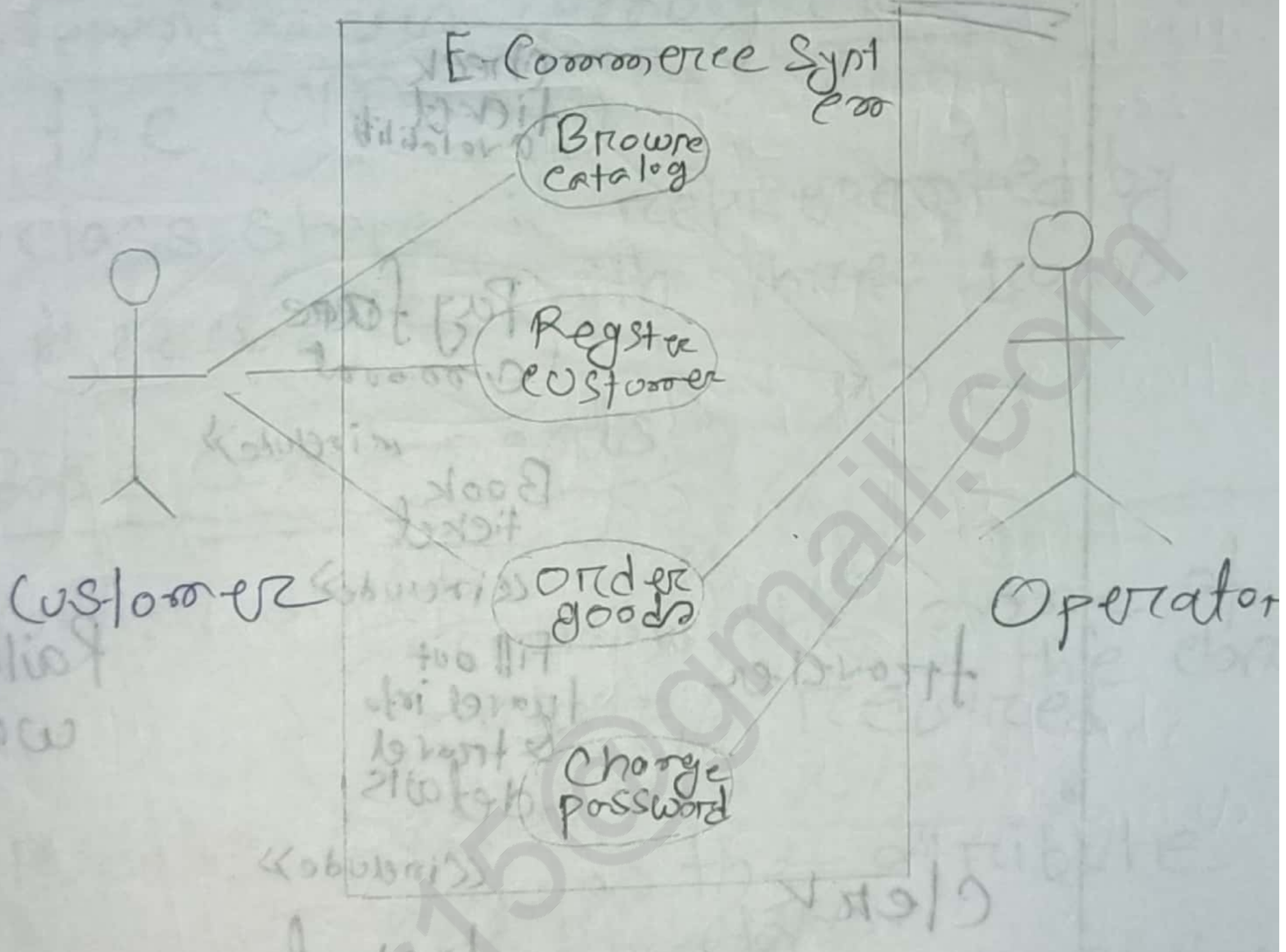
## Relationship

~~Rel~~

| Relationship | Function | Notation |
|---|---|---|
| Association | indicate the communication between actors & use case | ——— |
| Extend | indicate the insertion of additional behavior into a base use case | - - - - -> << extend>> |
| Include | describe a behavior that is inserted explicitly into a base case | - - - -> <<include>> |
| Generalizes | to indicate the communication between a general use case (actor) and a more specific use case that inherits and adds features to it. | ———▷ |

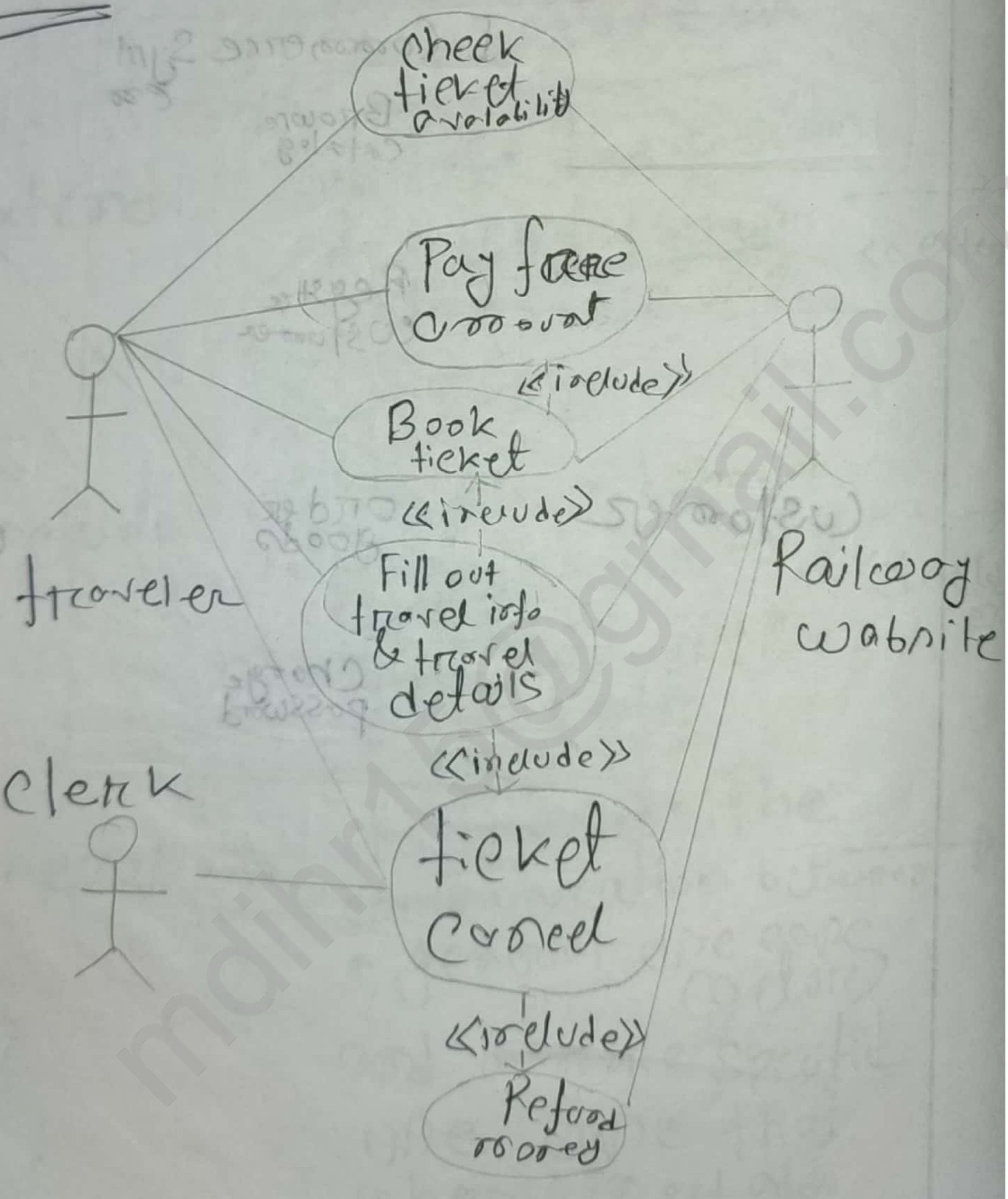
# Use case diagram of an e-commerce service

Soln



# Use case diagram for railway ticket reservation

Soln



## Class diagram

It is an illustration of the relationships and source code dependencies among classes in the UML.

* class shape is represented by a rectangle with three rows.

### Basic components:

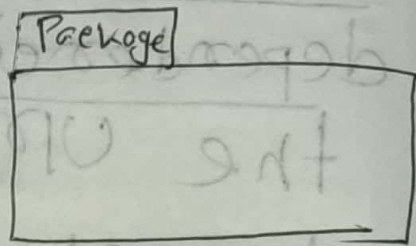
→ Upper section
  * contains the name of the class
  * always required

→ middle section
  * contains the attribute of the class.
  * describes the quality of class.

→ Bottom section
  * includes class operations (methods)

## Member access modifiers:

All classes have different access levels depending on the access modifier.




- → public (+)
- → private (−)
- → protected (#)
- → package (~)
- → Derived (/)
- → Static (underlined)

## Member scope

two scopes

- → Classifiers
- → instances

## Classdiagram relationship

**Inheritance** Child ——▷ parent

**Dependency** - - - - →

a change in one class cause change in another class.

**Association** ————

It describes static physical connection between two or more objects.

**Aggregation** ————◇

Subset of association.

In this kind of relationship, child class can exist independently of its parent class.

Multiplicity

one to many 1...* zero to one 0..1

zero to " 0...* defined m

Composition

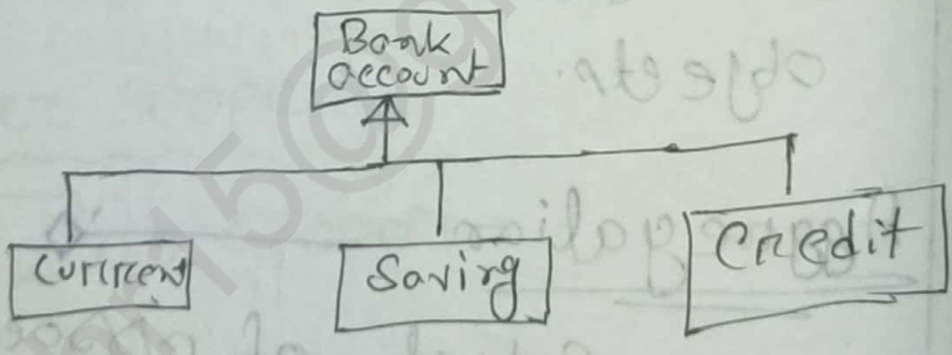
Subset of aggregation.

If parent is destroyed

Child will be "

Generalization

Relationship between parent class & child class. Child class is inherited from the parent class.




multiplicity.

Specific range of allowable instances of attribute

Show the class diagram of a library management system.

# Operating System

→ CPU scheduling algorithm

* FCFS
* SJF [ preemptive / non-preemptive
* SRTF
* priority
* Round robin

waiting time, WT = TAT − BT

Turn around time, TAT = CT − AT

= BT + WT

→ Deadlock

→ Resource allocation graph from

* vertex's set for
  process
  Resource
* Edge set

→ Banker's algorithm

→ Resource allocation graph.

→ page replacement algorithm

* FIFO
* LRU
* OPR

# Digital Logic Design (DLD)

→ Number system

→ positional number system

* Decimal
* Binary
* Octal
* Hexadecimal

→ Binary addition & subtraction

1's complement

2's ''

→ parity bit

* odd parity
* even ''

→ Boolean algebra

→ Karnaugh map

* 2-variable K-map
* 3-variable K-map

→ Conversion of SOP & POS

SOP ⇌ POS

→ Logic gates

Basic gates

→ AND

→ OR

→ NOT

→ Universal gates

→ NAND

→ NOR

→ Special gates

→ Ex-OR

→ Ex-NOR

→ Binary adder

→ multiplexer / Demultiplexer

4×1

8×1

→ Flip flop

• SR flip flop

* D flip flop
* JK "
* "

→ Encoder

→ Decoder

→ Counter

# Conversion of number system

Decimal → any other

~~Any other no~~

Any number system → decimal

Octal / Hexadecimal → binary

Binary → octal / Hexadecimal

Octal / Hexadecimal → Hexadecimal / Octal

# $(17.125)_{10} = (?)_2$

integer part

$$\begin{array}{r|l} 2 & 17 \\ 2 & 8-1 \\ 2 & 4-0 \\ 2 & 2-0 \\ 2 & 1-0 \\ & 0-1 \end{array}$$

LSB ↑ MSB

$(17)_{10} = (10001)_2$

fractional part

| | integer |
|---|---|
| $0.125 \times 2 = 0.250$ | 0 |
| $0.250 \times 2 = 0.500$ | 0 |
| $0.50 \times 2 = 1.00$ | 1 |

MSB ↓ LSB

$(0.125)_{10} = (001)_2$

$\therefore (17.125)_{10} = (10001.001)_2$

Ans

# $(423.150)_{10} = (?)_8$

$$\begin{array}{r|l} 8 & 423 \\ \hline 8 & 52 - 7 \\ \hline 8 & 6 - 4 \\ \hline & 0 - 6 \end{array}$$

LSB ↑ MSB

$.150 \times 8 = 1.200$
$.200 \times 8 = 1.600$
$.600 \times 8 = 4.800$
$.800 \times 8 = 6.400$

| integer |
|---|
| 1 |
| 1 |
| 4 |
| 6 |

MSB ↓ LSB

$\therefore (423)_{10} = (647)_8$ $\therefore (.150)_{10} = (0.1146)_8$

$\therefore (423.150)_{10} = (647.1146)_8$ Ans

# $(423.125)_{10} = (?)_{16}$

$$\begin{array}{r|l} 16 & 423 \\ \hline 16 & 26 - 7 \\ \hline 16 & 1 - 10 \\ \hline & 0 - 1 \end{array}$$

LSB ↑ MSB

$.125 \times 16 = 2.000$

| integer | fr |
|---|---|
| 2 | 0 |

96
16
2

103
96
7

125
16
750
125
2.000

$\therefore (423)_{10} = (1A7)_{16}$ $\therefore (.125)_{16} = (0.2$

$\therefore (423.125)_{10} = (1A7.2)_{16}$ Ans

# $(1101.11)_2 = (?)_{10}$

$= (1\times10^3)+(1\times10^2)+(0\times10^1)+(1\times10^0)+(1\times10^{-1})+(1\times10^{-2})$

$= 1000+100+1+\frac{1}{10}+\frac{1}{100}$

$= 1101 + \frac{10+1}{100}$

$= \frac{110100+11}{100}$

$= \frac{110111}{100}$

$= 1101.11$

# $(1101.11)_2 = (?)_{10}$

$= (1\times2^3)+(1\times2^2)+(0\times2^1)+(1\times2^0)+(1\times2^{-1})+(1\times2^{-2})$

$= 8+4+0+1+\frac{1}{2}+\frac{1}{4}$

$= 13+\frac{2+1}{4}$

$= \frac{52+3}{4} = \frac{55}{4}$

$= 4 \,|\, 55 \,|\, 13.75$

4

15 38

$(1101.11)_2 = (13.75)_{10}$

# $(145.23)_8 = (?)_{10}$

$= (1\times 8^2) + (4\times 8^1) + (5\times 8^0) + (2\times 8^{-1}) + (3\times 8^{-2})$

$= 64 + 32 + 5 + \frac{2}{8} + \frac{3}{64}$

$= 101 + \frac{16+3}{64}$

$= 101 + \frac{19}{64}$

$= \frac{19}{64} + 101$

$= 64 ) 190 ( 0.29$
$128$
$620$
$576$
$240$

$= 101.29$

$= (101.3)_{10}$ Ans

# $(1A5.2)_{16} = (?)_{10}$

$= (1\times16^2) + (17\times16^1) + (5\times16^0) + (2\times16^{-1})$

$= 256 + (10\times16) + 5 + \cancel{\frac{2}{16}}\,8$

$= 256 + 160 + 5 + \frac{1}{8}$

$= 416 + 5 + \frac{1}{8}$

$= 421 + \frac{1}{8}$

$= 421 + 8\,)\,10\,(\,.125$
$\quad 8$
$\quad 20$
$\quad 16$
$\quad 40$
$\quad 40$
$\quad 0$

$= 421 + 0.125$

$= 421.125$

$= (421.125)_{10}$ Ans

$16 \times 16 = 96 + 160 = 256$

421

# $(756.24)_8 = (?)_2$

$7 \rightarrow 111$

$5 \rightarrow 101$

$6 \rightarrow 110$

$2 \rightarrow 010$

$4 \rightarrow 100$

$\therefore (756.24)_8 = (111101110.010100)_2$

Ans

# $(3F5.4C)_{16} = (?)_2$

| 3 | F | 5 | . | 4 | C |
|---|---|---|---|---|---|
| ↓ | ↓ | ↓ | | ↓ | ↓ |
| 0011 | 1111 | 0101 | | 0100 | 1100 |

$\therefore (3F5.4C)_{16} = (001111110101.01001100)_2$

# $(11101100.11001)_2 = (?)_8$

$\underline{011}\;\underline{101}\;\underline{100} \cdot \underline{110}\;\underline{010}$

$= 3\;5\;4 \cdot 6\;2$

$= (354.62)_8$

Ans

# $(11110100010.01011101)_2 = (?)_{16}$

$\underset{\downarrow}{\underline{0111}}\;\underset{\downarrow}{\underline{1010}}\;\underset{\downarrow}{\underline{0010}} \cdot \underset{\downarrow}{\underline{0101}}\;\underset{\downarrow}{\underline{1101}}$

$= 7\;A\;2 \cdot 5\;D$

$= (7A2.5D)_{16}$

# $(675.24)_8 = (\ )_{16}$

110 111 101 · 010 100

$\therefore (675.24)_8 = (110111101.010100)_2$

Now

0001 1011 1101 · 0101 0000

= 1 B D · 5 0

$= (1BD.50)_{16}$

# Binary addition- subtraction

## Binary addition

| Binary Number | Sum | Carry |
|---|---|---|
| 0+1 | 1 | 0 |
| 0+0 | 0 | 0 |
| 1+0 | 1 | 0 |
| 1+1 | 0 | 1 |
| 1+1+1 | 1 | 1 |

## Binary subtraction

| Binary number | Subtraction | Carry |
|---|---|---|
| 0-0 | 0 | 0 |
| 1-0 | 1 | 0 |
| 0-1 | 1 | 1 |
| 1-1 | 0 | 0 |

# $(1011.0110)_2 + (1101.0111)_2 = (?)_2$

$$
\begin{array}{r}
1011.0110 \\
1101.0111 \\
\hline
11000.1101
\end{array}
$$

$= (11000.1101)_2$ Ans

# $(110101.110)_2 - (010110.101)_2 = (?)_2$

$$
\begin{array}{r}
110101.110 \\
(-)\ 010110.101 \\
\hline
10\,0\,1\,1\,1.001
\end{array}
$$

2's complement = 1's complement + 1

# Boolean algebra:

| logical OR | logical AND |
|---|---|
| $A+0=A$ | $A\cdot 1=A$ |
| $A+1=1$ | $A\cdot 0=0$ |
| $A+A=A$ | $A\cdot A=A$ |
| $A+\overline{A}=1$ | $A\overline{A}=0$ |

Associative law:

$$A+(B+C)=(A+B)+C$$
$$A\cdot(B\cdot C)=(A\cdot B)\cdot C$$

De morgan's law

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

Distributive law

$$A(B+C)=AB+AC$$
$$A+B\cdot C=(A+B)(A+C)$$

Commutative law

$$A+B=B+A$$
$$A\cdot B=B\cdot A$$

☐ minterm: ~~AND opera~~

if $x$ and $y$ are two binary variable then, $(xy, xy', x'y, x'y')$ each of the four AND terms called minterm or a standard product.

☐ maxterm: if $x$ and $y$ are two binary variable then $(x+y, x+y', x'+y, x'+y')$ each of the four OR terms called maxterm or a standard sum.

☐ Canonical form:

Boolean function expressed as a sum of product or minterm or product of sum of maxterms are said to be in canonical form.

Express the Boolean function

# $F = A + B'C$ in a sum of minterms

$A = A(B+B')$
$= AB + AB'$

$A = (AB+AB')(C+C')$
$= ABC + ABC' + AB'C + AB'C'$

$B'C = B'C(A+A')$
$= B'CA + B'CA'$

$\therefore F = A + B'C$
$= ABC + ABC' + AB'C + AB'C' + B'CA + B'CA'$
$= ABC + ABC' + AB'C + AB'C' + AB'C + A'B'C$
$= ABC + ABC' + AB'C + AB'C' + A'B'C$
$= m_7 + m_1 + m_5 + m_4 + m_6$
$= m_1 + m_6 + m_4 + m_5 + m_7$

$\therefore F(A,B,C) = \Sigma(1,4,5,6,7)$

Ans

⊞ Express the Boolean function $F = xy + x'z$ in a product of maxterm.

First convert the function into OR terms.

$$F = xy + x'z$$
$$= (xy + x')(xy + z) \quad \text{(distribution law)}$$
$$= \cancel{(x'+x)}(x'+y)(y+z)(x+z)$$
$$= \cancel{x'+x+x'}$$
$$= (x'+y)(y+z)(x+z)$$
$$= (x'+y+zz')(y+z+xx')(x+z+yy')$$
$$= (x'+y+z)(x'+y+z')(y+z+x)(y+z+x')(x+z+y)(x+z+y')$$
$$= (x+y+z)(x'+y+z)(x'+y+z')(x'y+z)(x+y'+z)$$
$$= (x+y+z)(x'+y+z)(x'+y+z')(x+y'+z)$$

$= \cancel{M_7 M_3 M_2 M_5}$

$= M_0 M_4 M_5 M_2$

$\therefore \cancel{F(x, y, z) = \prod (M_7, M_3, M_2, M_5)}$

$\therefore F(x, y, z) = \prod(0, 2, 4, 5)$

Am

NB total number of maxterms or minterms is $2^n$, where $n$ is the number of binary variables in the function.

**POS → SOP**

$F(x, y, z) = \prod(0, 2, 4, 5)$

$\therefore F(x, y, z) = \sum(1, 3, 6, 7)$

**SOP → (SOP)' → POS**

$F(A, B, C) = \sum(1, 4, 5, 6, 7)$

$F'(A, B, C) = \sum(0, 2, 3)$

$= m_0 + m_2 + m_3$

$F = (m_0 + m_2 + m_3)'$

$= m_0' \cdot m_2' \cdot m_3' = M_0 M_2 M_3$

$= \prod(0, 2, 3)$

## Map method:

A simple straightforward procedure for minimizing Boolean functions.

2 types

1) veitch diagram → Veitch
2) Karnaugh map → Karnaugh

K-map

2 variable
3 variable
4 variable

2 variable K-map

| $y \backslash z$ | 0 | 1 |
|---|---|---|
| 0 | $m_0$ | $m_1$ |
| 1 | $m_2$ | $m_3$ |

3 variable K-map

| $x \backslash yz$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 3 | 2 |
| 1 | 4 | 5 | 7 | 6 |

4-variable K-map

| $wx \backslash yz$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 3 | 2 |
| 01 | 4 | 5 | 7 | 6 |
| 11 | 12 | 13 | 15 | 14 |
| 10 | 8 | 9 | 11 | 10 |

Simplify the following function

$$f(\omega, X, Y, Z) = \omega \bar{X} \bar{Y} + \omega Y + \bar{\omega} Y \bar{Z}$$

Sol:

$$f(\omega, X, Y, Z) = \omega \bar{X} \bar{Y} + \omega Y + \bar{\omega} Y \bar{Z}$$

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

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

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

| wx \ yz | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | | | 1 |
| 01 | | | | 1 |
| 11 | | | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

∴ Simplified boolean function is

$$f = w\overline{x} + y\overline{z} + wy$$

Am

□ Simplify the Boolean function

$$F = x'yz + x'yz' + xy'z' + xy'z$$

Soln

| x \ yz | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | | 1 | 1 |
| 1 | 1 | 1 | | |

∴ Simplified form is $f = \overline{x}y + x\overline{y} + \cancel{\ }$

Am

Simplify the Boolean function.

$$F = x'yz + xyz' + xyz + xyz'$$

| x \ yz | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | | 1 | |
| 1 | 1 | | 1 | 1 |

∴ Simplified boolean function

$$F = YZ \cancel{+XY} + X'\bar{Z}$$

Ans

Q. Simplify the boolean function.

$$F = A'C + A'B + AB'C + BC$$

**Sol^n**

$$F = A'C + A'B + AB'C + BC$$

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | 1 | 1 | 1 |
| 1 | | 1 | 1 | |

∴ Simplified boolean function

$$f = C + A'B$$

Ans

CET - 2017

Simplify the following canonical sum of product of a function using K-map.

$$F(A, B, C, D) = \prod(1, 3, 5, 7, 13, 15)$$

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

$\therefore$ Simplified form is $= (A + \bar{D})(\bar{B} + \bar{D})$

Simplify the following boolean function.

$$F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$$

| $x \backslash yz$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | | | 1 |
| 1 | 1 | 1 | | 1 |

$\therefore$ Simplified function is $f = \bar{z} + xy'$ Ans

Simplify the following boolean function

$$F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$$

| $wx \backslash yz$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | 1 |
| 01 | 1 | 1 | | 1 |
| 11 | 1 | 1 | | 1 |
| 10 | 1 | 1 | | |

∴ Simplified boolean function,

$$f = \overline{Y} + \overline{WZ} + X\overline{Z}$$

Ans

⊞ Simplify the Boolean function

$$f_{(x,y,z)} = \prod m(0,1,2,4)$$

| x \ yz | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | | 0 |
| 1 | 0 | | | |

∴ Simplified boolean function is,

$$F = (X+Y)(X+Z)(Y+Z)$$

Ans

BUET-MS-16

Simplify the following K map

(Don't care condition)

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | X | X | 0 | X |
| 01 | 1 | 1 | 1 | 0 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 1 | 1 |

∴ Simplified form is $f =$

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

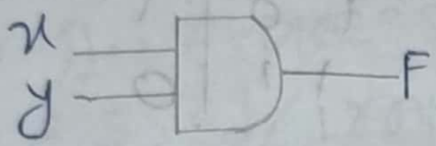
Ans

don't care condition

Simplify the boolean function $F(w,x,y,z) = \Sigma(1,3,7,11,15) + d\Sigma(0,2,5)$

# Logic gates:

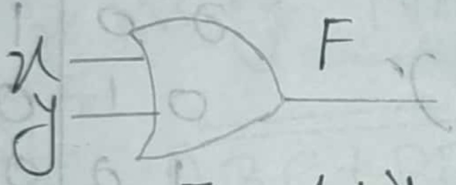
AND




$F = xy$

| x | y | F |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

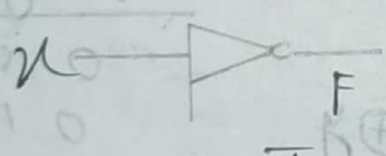
OR




$F = x + y$

| x | y | F |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Inverter




$F = \bar{x}$

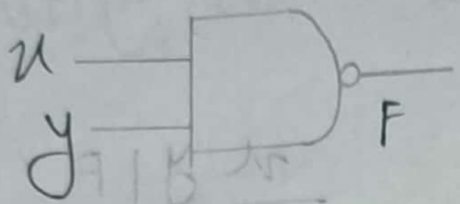
| x | F |
|---|---|
| 0 | 1 |
| 1 | 0 |

Buffer


$F = x$

| x | F |
|---|---|
| 0 | 0 |
| 1 | 1 |

NAND




$F = (xy)'$

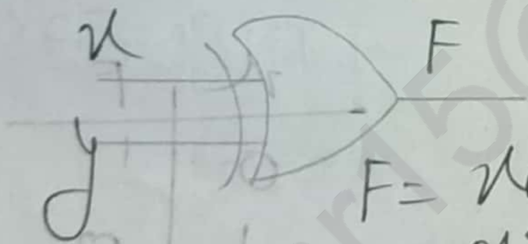
| x | y | F |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

NOR


$F = (x+y)'$

| x | y | F |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

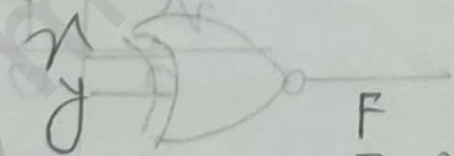
Ex-OR




$F = x \oplus y$
$= xy' + x'y$

| x | y | F |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Ex-NOR




$F = xy + x'y'$

| x | y | F |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Implementation of NAND, NOR gate

De morgan

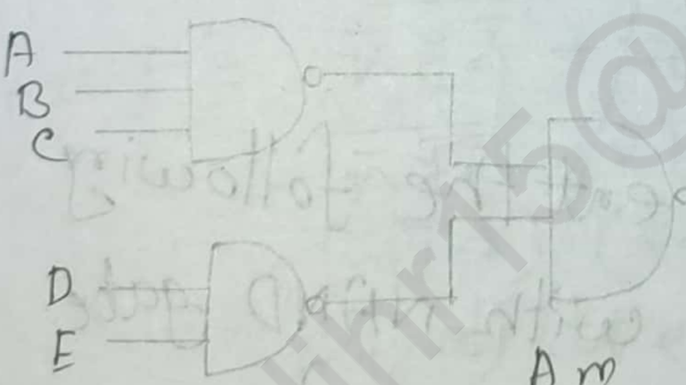
* An expression of SOP can be expressed / implemented with NAND gate
* An expression of POS can be expressed/implemented with NOR gate

Q. $F = ABC + DE$, Implement this by using NAND gate

Soln

$F = \overline{\overline{ABC + DE}}$

$= \overline{\overline{ABC} \cdot \overline{DE}}$




$F = \overline{\overline{ABC} \cdot \overline{DE}}$

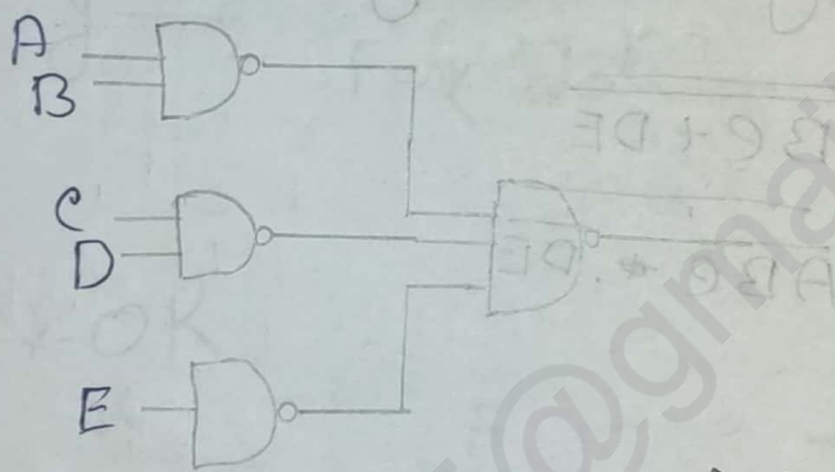
Ans

Implement the boolean function using NAND gate.

$$F = AB + CD + E$$

Soln

$$F = \overline{\overline{AB + CD + E}}$$
$$= \overline{\overline{AB} \cdot \overline{CD} \cdot \overline{E}}$$




Implement the following function with NAND gate.

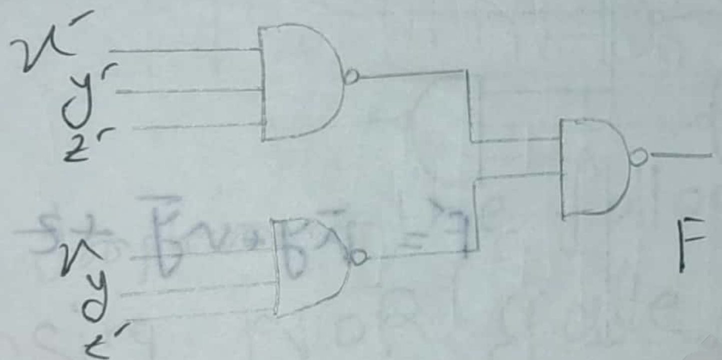
$$F(x, y, z) = \Sigma(0, 6)$$

| $x \backslash yz$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | | | |
| 1 | | | | 1 |

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

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

$$= \overline{\overline{\bar{x}\bar{y}\bar{z}} \cdot \overline{xy\bar{z}}}$$




$$F = \overline{\overline{x'y'z'} \; \overline{xy\bar{z}}}$$

$$\text{Now } F' = \overline{\bar{x}\bar{y}\bar{z} + xy\bar{z}}$$

$$= \overline{\bar{x}\bar{y}\bar{z}} \cdot \overline{xy\bar{z}}$$

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

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

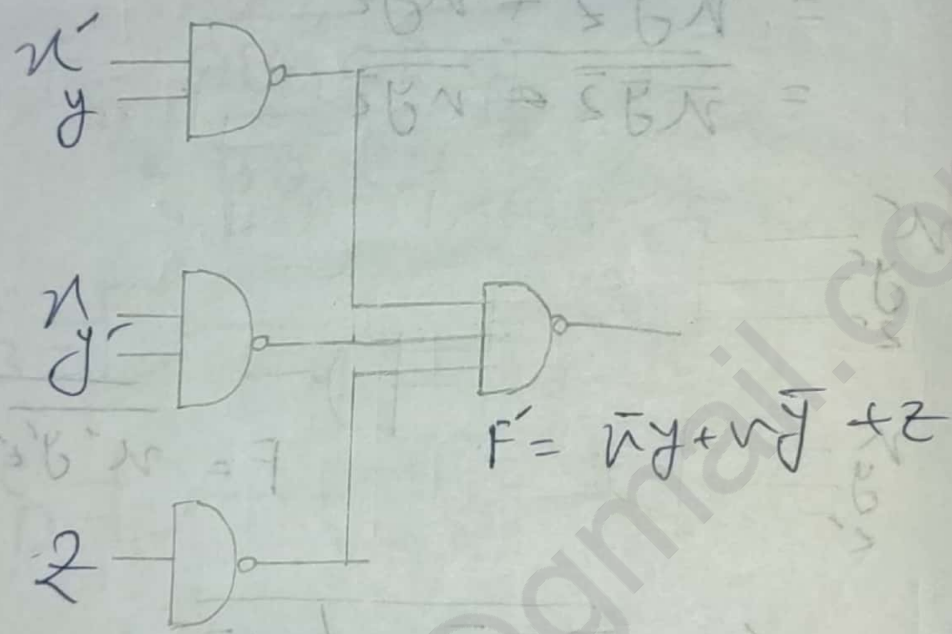
$$= 0 + x\bar{y} + xz + \bar{x}y + 0 + yz + z^2$$

$$= x\bar{y} + \bar{x}y + xz + yz + z$$

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

Now $F(x,y,z) = \Sigma(1,2,3,4,5,7)$

Simplified complement of F is $\overline{F} = \bar{x}y + x\bar{y} + z$



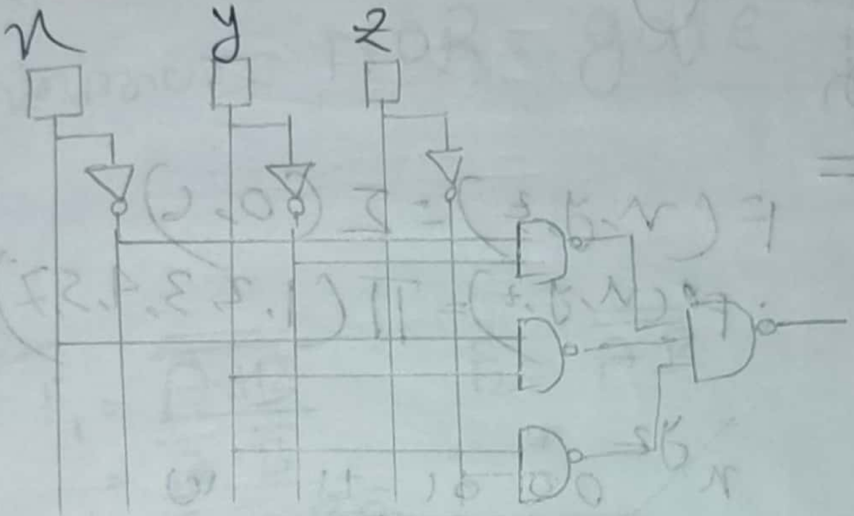

BUET 2016

Given the following functions $F = xy + x'y' + y'z$, Implement it with NAND gate.

Sol$^n$

$$F = \overline{\overline{xy + x'y' + yz}}$$

$$= \overline{\overline{xy} \cdot \overline{x'y'} \, \overline{yz}}$$



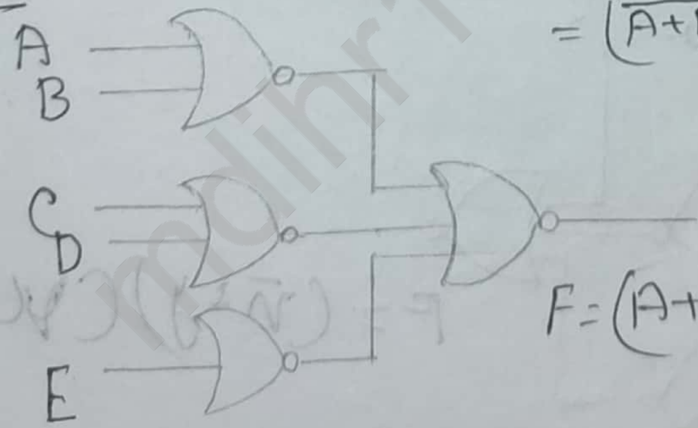

Implement the following function using NOR gate

$$F = (A+B)(C+D)E$$

$$F = \overline{\overline{(A+B)(C+D)E}}$$

$$= \overline{\overline{(A+B)} + \overline{(C+D)} + \overline{E}}$$

Sol$^n$




$F = (A+B)(C+D)E$

☐ Implement the function $F(x,y,z)=\Sigma(0,6)$ with NOR gates.

**Soln**

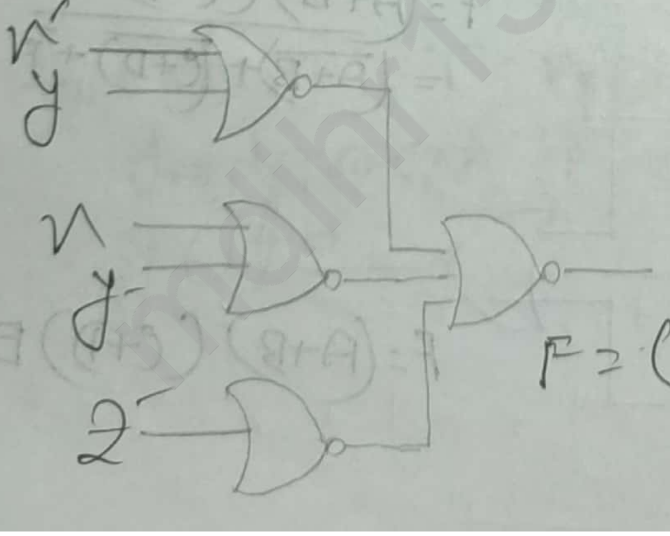
$F(x,y,z)=\Sigma(0,6)$

$\therefore F'(x,y,z)=\Pi(1,2,3,4,5,7)$

| $x \backslash yz$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | |

$\therefore F'(x,y,z)=\cancel{z+xy'+x'y}$

$\overline{z}(\bar{x}+y)(x+\bar{y})$




$F=(\bar{x}+y)(x+\bar{y})\overline{z}$ Ans

BUET ms-2019

$F_1 = A+B$ and $F_2 = AB$

a) use the minimum NAND gate for $F_1$ and $F_2$ and b) use the minimum NOR gate for $F_1$ and $F_2$.

Sol$^n$ a

$F_1 = \overline{\overline{A+B}}$

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

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

$F_2 = \overline{\overline{AB}}$

b

$F_1 = A+B$

$= \cancel{B}$

$F_2 = \overline{\overline{AB}}$

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

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

# Combinational logic Circuits

data transfer

- Encoders ✓
- Decoders ✓
- Multiplexer ✓
- Demultiplexer

Arithmatic & Logical Functions

- Adder
- Half adder
- Ful adder
- Subtractor
- Half - subtractor
- Ful - subtractor
- Programmable Logic Devices (PLD)
- Code converters

## Decoder:

Combinational circuit that converts binary information from $n$ input line to a maximum $2^n$ output lines.

| | # input line | # output line |
|---|---|---|
| Decoder → | $n$ | $\leq 2^n$ |
| Encoder → | $\leq 2^n$ | $n$ |
| DeMux → | 1 | $2^n$ |
| Mux → | $2^n$ | 1 |

# Selection line $n$

BUET MS-2016

Implement 2×4 Decoder using only basic gates (AND, OR, NOT)

Sol$^n$

truth table

2 input line
4 output line

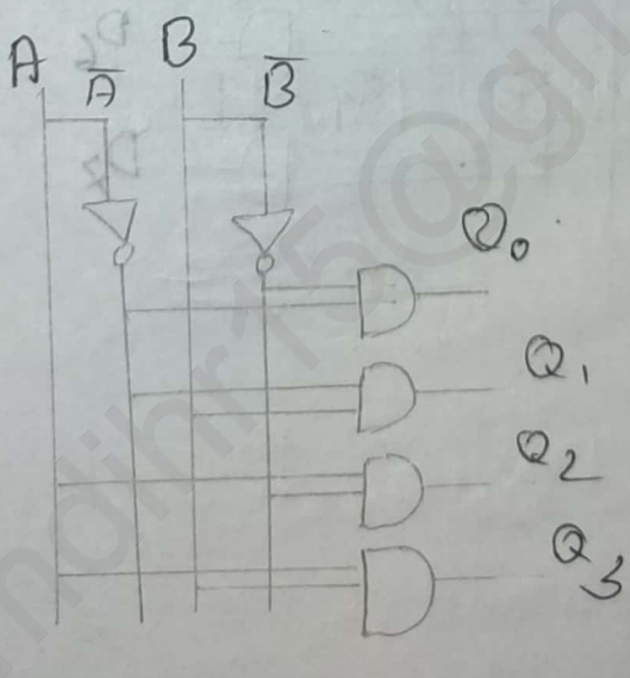
| A | B | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

$Q_0 = \overline{A}\,\overline{B}$

$Q_1 = \overline{A}B$

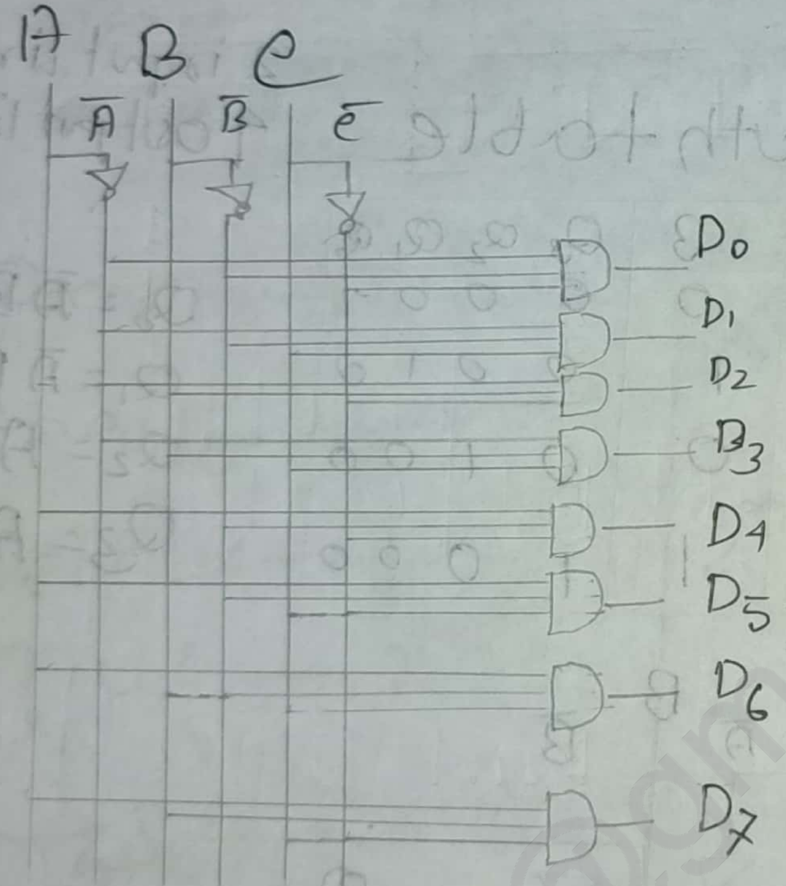
$Q_2 = A\overline{B}$

$Q_3 = AB$



Implement 3 to 8 decoder using (AND, OR, NOT) gate.

Sol$^n$:

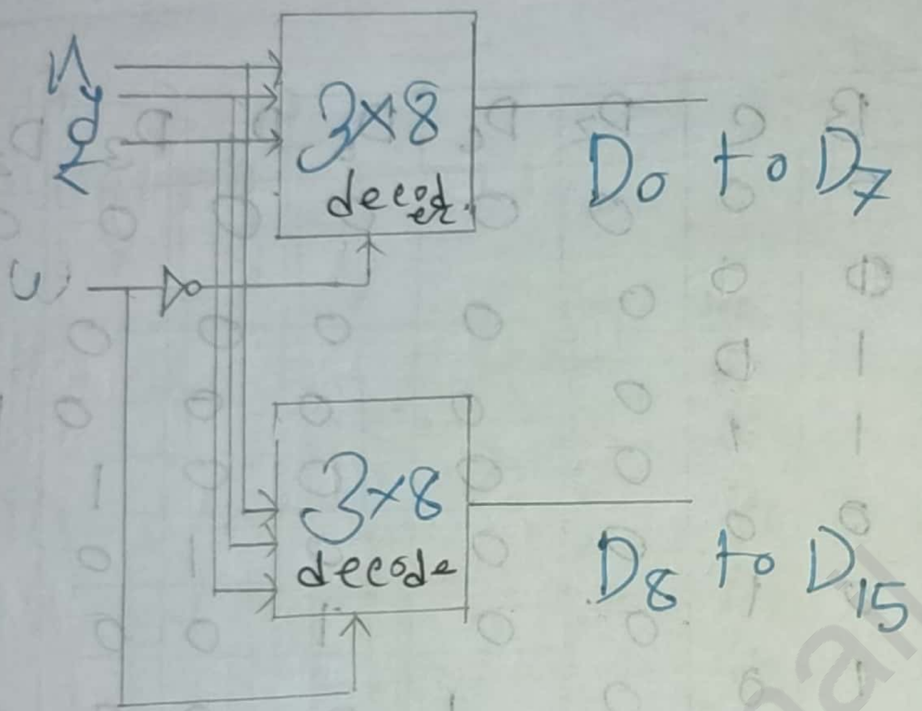


3 input line
8 output line

truth table

| A | B | C | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Construct a 4 to 16 decoder using two 3×8 decoder




if $w=0$ first 3×8 decoder will

if $w=1$ second 3×8 " "

NB

Decoder/demultiplexer circuit can be connected together to form a larger decoder circuit.

# Encoder:

An encoder is a digital function that produces a a reverse operation from that decoder.

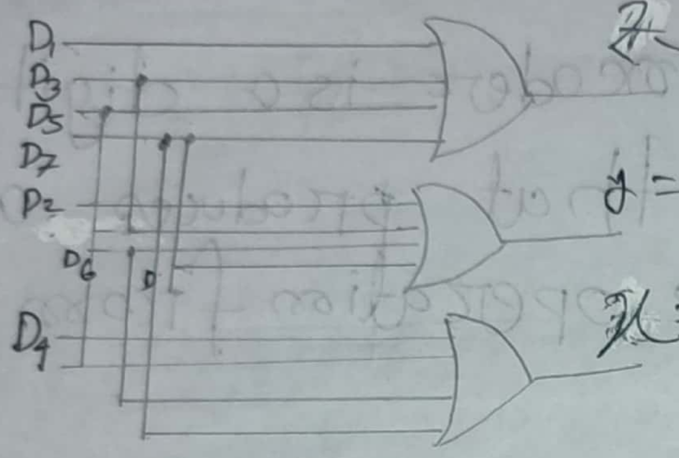
→ receives $2^n$ input line

→ $n$ output line

□ Implement octal to binary encoder.

truth table

| Inputs | | | | | | | | Output | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | x | y | z |
| 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 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 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 |




$z = D_1 + D_3 + D_5 + D_7$

$y = D_2 + D_3 + D_6 + D_7$

$x = D_6 + D_7 + D_4 + D_5$

Priority encoder:

If both $D_2$ and $D_7$ are logic-1 simultaneously, the output will be 101 because $D_5$ has a higher priority over D.

digital multiplexer choose ~~one~~ binary information from one of many input lines.

## Multiplexer:

transmitting a large number of information units over a smaller number of channels or lines.

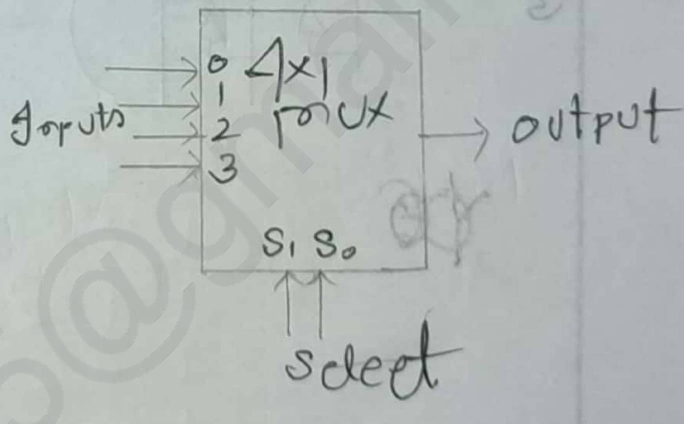
$2^n$ input line

1 output line

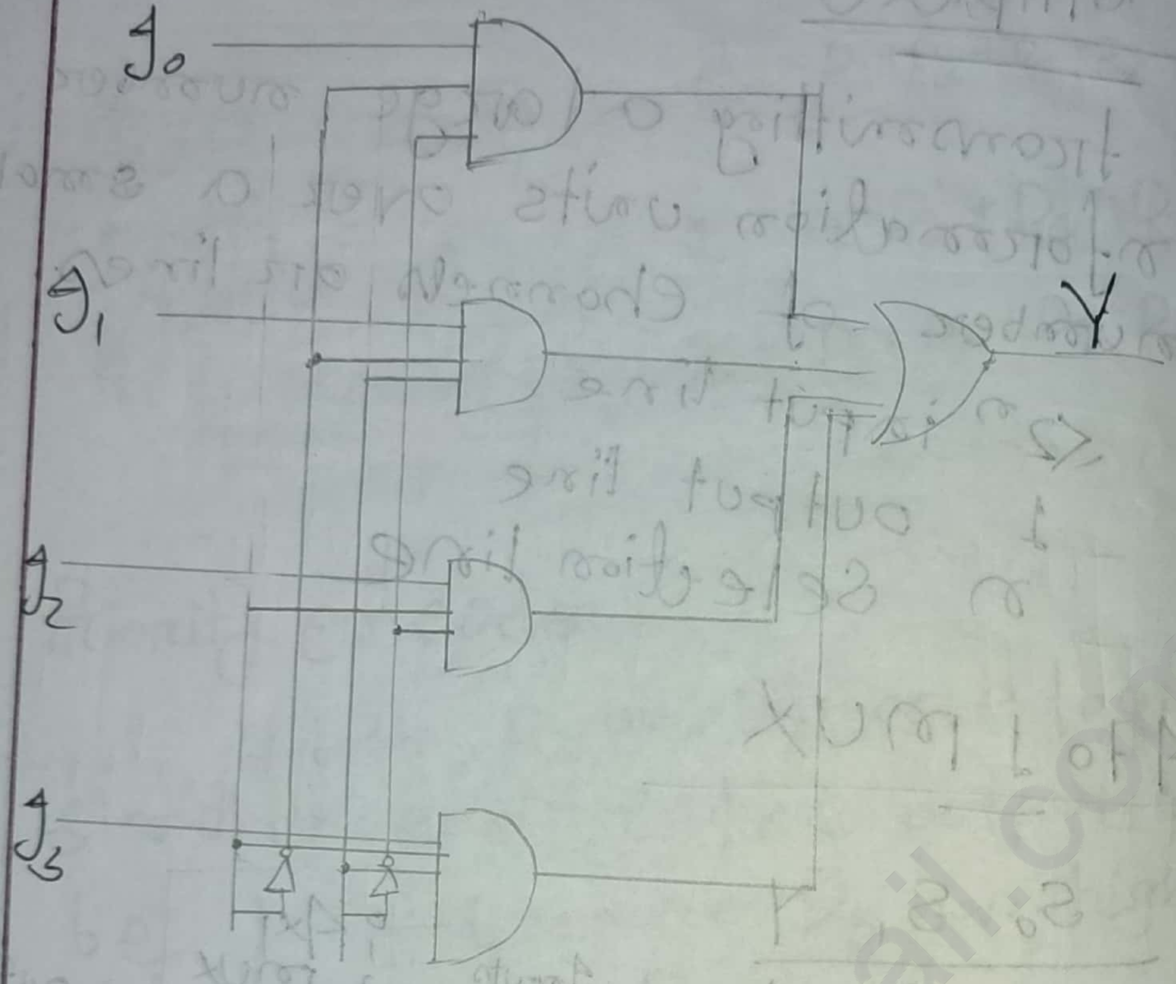
$n$ selection line

## 4 to 1 MUX

| $S_0$ | $S_1$ | $Y$ |
|---|---|---|
| 0 | 0 | $I_0$ |
| 0 | 1 | $I_1$ |
| 1 | 0 | $I_2$ |
| 1 | 1 | $I_3$ |




$$Y = \bar{S}_0 \bar{S}_1 I_0 + \bar{S}_0 S_1 I_1 + S_0 \bar{S}_1 I_2 + S_0 S_1 I_3$$


I0
I1
I2
I3
Y

## Half adders:

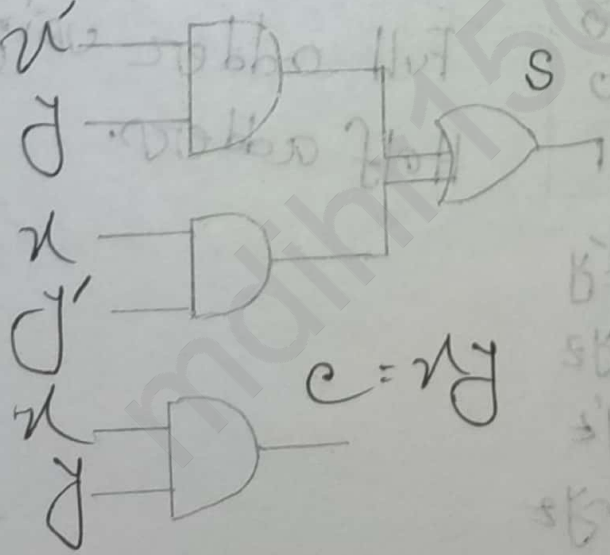
A combinational circuit that performs the addition of two bits.

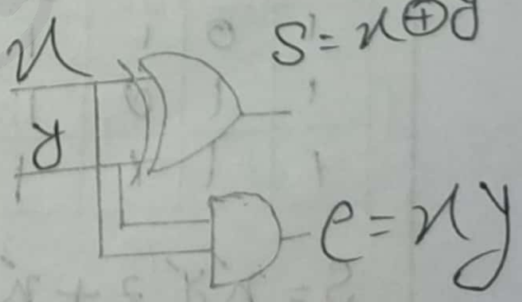
### truth table

| x | y | c | s |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

$S = x'y + xy'$
$= x \oplus y$
$C = xy$

Now
$S = x'y + xy'$
$= \cancel{(x'+y)(x+y')}$
$= \Sigma(1,2)$
$= \Pi(0,3)$



$$S=\overline{z}(x'y+xy')+z(xy+x'y')$$
$$=x\oplus y\oplus z$$

## Full adder:

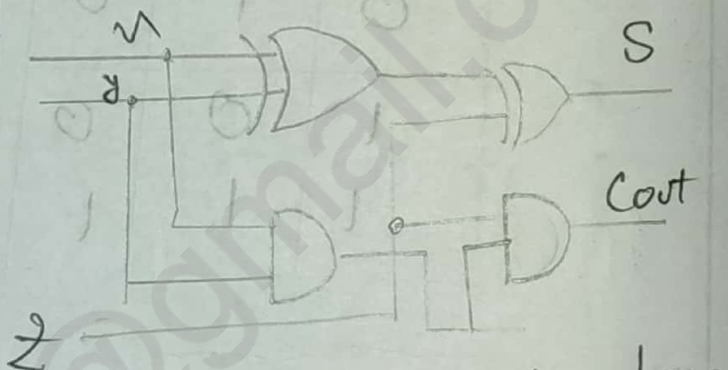
A combinational circuit that performs the addition of three bits (two significant bits and a previous carry) is a full adder.

| $z$ \ $xy$ | $x'y'$ | $x'y$ | $xy$ | $xy'$ |
|---|---|---|---|---|
| $\overline{z}$ | | | 1 | |
| $z$ | | 1 | 1 | 1 |

$\therefore C_{out}=xy+yz+xz$

truth table

| $x$ | $y$ | $z$ (in) | $C_{out}$ | $S$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |




Full adder using two Half adders.

$$S=x'y'z+x'z'y+xy'z'+xyz$$

$$C_{out}=x'yz+xy'z+xyz'+xyz$$

## Half subtractor

# Computer Network

**IP address:** Unique global address for a network interface.

* $IPV_4 \rightarrow 32$ bits
* $IPV_6 \rightarrow 128$ "

**IPV4:**

2 notations
→ dotted decimal notation
→ hexa decimal "

→ 32 bits long
→ encodes a network number
→ " a host "
→ written in dotted decimal notation
→ network portion has the same value for all hosts.

| network prefix | host number |
|---|---|

## Formula

# network address = $2^{(\text{Network length} - \text{fixed bits})}$

# host per network = $2^{(\text{host id length})} - 2$

**Network address:** all bits of host portion is zero (0)

**Broadcast address:** all bits of host portion is one (1)

**Mask:** A binary number that gives the network address in the address block when AND operation is bitwise.

Network Host

| Class name | application | N.ID length | H.ID length | High order bit | First octet range | # Network Add. | # Host address/Net |
|---|---|---|---|---|---|---|---|
| A | → government<br>→ large organization | 8 | 24 | 0 | 0–127 | 128 | 16777214 |
| B | medium company | 16 | 16 | 10 | 128–191 | 16384 | 65534 |
| C | Small company | 24 | 8 | 110 | 192–223 | 2097152 | 254 |
| D | multi Casting | – | – | 1110 | 224–239 | | |
| E | Reserved for expe | – | – | 11110 | 240-255 | | |

BUET'17

Consider an IP address (223.16.37.0/29). Find the subnet address and last address for this network. If we want 12 address in the subnet what will be the value of the subnet mask?

Soln: ~~Network~~

Subnet mask: 11111111.11111111.11111111.11111

= 255.255.255.248

Ans

IP address = 223.16.37.0/29

∴ First address = 223.16.37.1/29

usable

∴ Last usable " = 223.16.37.6/29

Ans

For 12 address in the subnet we require $\lceil \log 14/\log 2 \rceil = 4$ bits in host portion.

So, subnet mask: 255.255.255.240/28

Ans

E1 ms-17

3) Tables of 4 distinction IP addresses are given associated with each interface. Another table was given with 4 different IP address with the blank field of interface. Fill the interfaces column for each destination IP using 1st table.

| Network No | Net mask | Interface |
|---|---|---|
| 128.96.170.0 | 255.255.254.0 | Interface0 |
| 128.96.168.0 | 255.255.254.0 | Interface1 |
| 128.96.166.0 | 255.255.254.0 | $R_2$ |
| 128.96.164.0 | 255.255.252.0 | $R_3$ |
| 0.0.0.0 | Default | $R_4$ |

Solⁿ

| IP address | interface |
|---|---|
| 128.96.171.92 | interface 0 |
| 128.96.167.151 | $R_2$ |
| 128.96.163.151 | $R_4$ |
| 128.96.164.121 | $R_3$ |

For interface 0 3rd octate range $\geq 170$

" " 1 3rd octate range 168-169

" " $R_2$ " octate range 166-167

" " $R_3$ " octate range 164-165

" $R_4$ " octate range $< 164$

BU(19)

⊞ Given a network address 172.16.0.0 where 512 subnets and 100 hosts per subnet are needed. What should be the output subnet mask!

Sol^n:

Network address = 172.16.0.0

First octate range 128-191

So it is a class B IP address.

- For 100 ~~subnet~~ hosts we require

$\left\lceil \frac{\log 100}{\log 2} \right\rceil$ bits in host portion

= 7 host bit

Remaining bit = 32 - (16+7)

= 32 - 23

= 9

For 512 subnet we need $\left\lceil \frac{\log 512}{\log 2} \right\rceil$ bits

= 9 bits

∴ So Number of bits in subnet portion

16+9 = 25

∴ subnet mask: 11111111 . 11111111 . 11111111 . 10000000/25

= 255.255.255.128/25

Ans

BUET MS-19

Q. If a network address of 29.0.0.0 is given and we require 512 subnet and 160 hosts per subnet, then what will be the mask subnet.

Soln:- First octate range 0-127

So it is class A IP.

For 512 subnet we require $\lceil \log 512 / \log 2 \rceil$ bits

= 9 bits

" 160 hosts we require $\lceil \log 160 / \log 2$

$= 8$

$\therefore$ total bit is subnet portion

$= (8+9)$

$= 17$ bits.

$\therefore$ Subnet mask: 11111111.11111111.10000000.00000000

$= 255.255.128.0/17$

Ans

---

In the following IP address 172.20.0.0/27 Find how many subnets and hosts per subnet.

Answer:

First octate range 128-191

So it is class B IP address

$\therefore$ ~~Number of bits in host~~ Network position $= 9$

$\therefore$ For subnets number of bits $= 27-16$

$= 11$

$\therefore$ " hosts " of bits $= 32-27$

$= 5$

$\therefore$ Number of subnets $= 2^{11}$

$= 2048$

$\therefore$ Number of hosts/subnet $= 2^5$

$= 32$

$\therefore$ " of usable hosts $= 32-2$

$= 30$

Ans

# Discrete mathematics

- → Counting basic
- → Permutation & combination
- → the pigeonhole principle
- → Recurrence relations
- → Divide and Conquer algorithms
- → Inclusion exclusion

Prove the following lemma by contradiction:

$a^2+b^2=c^2$, there exists only 1 case such that a, b, c are consecutive non negative integers (3, 4, 5).

Sol:

Let, $a = n$

$b = n+1$

$c = n+2$

$\therefore$ ~~a+b~~ $n^2 + (n+1)^2 = (n+2)^2$

$\Rightarrow n^2 + \cancel{n^2} + 1 + 2n = \cancel{n^2} + 4n + 4$

$\Rightarrow n^2 - 2n - 3 = 0$

$\Rightarrow n^2 - 3n + n - 3 = 0$

$\Rightarrow n(n-3) + 1(n-3) = 0$

$\Rightarrow (n-3)(n+1) = 0$

$\therefore n = 3, -1$

as $-1$ is ~~not~~ negative

So $n = 3$

$\therefore a = n = 3$

$b = n+1 = 4$

$c = n+2 = 5$

$\therefore (a, b, c) = (3, 4, 5)$

Ans

4. In any 6 numbers, show at least 2 of them heaving same reminder after dividing 5.

Soln:

There are 5 possible reminders when a number is divided by 5: 0, 1, 2, 3, 4.

Let, ~~take~~

reminders = pigeonhole

numbers = pigeon.

As we have 6 numbers, then by the Pigeon-Hole Principle there are at least two pigeon sharing the same ~~va~~ reminder.

4 Prove that there is no positive integer such that $n^2+n^3=100$.

**Soln**

Let, $n = 0, 1, 2, 3, \ldots$

Now if, $n=0$

$n^2+n^3=0+0=0\neq 100$

$n=1,\ n^2+n^3=1+1=2\neq 100$

$n=2,\ n^2+n^3=2^2+2^3=4+8=12\neq 100$

$n=3,\ n^2+n^3 = 3^2+3^3 = 9+27 = 36 \neq 100.$

$n=4,\ n^2+n^3 = 4^2+4^3 = 16+64 = 80 \neq 100$

$n=5,\ n^2+n^3 = 5^2+5^3 = 25+125 = 150 \neq 100$

As for $n=4$ $f(n) < 100$

and if $n=5$ $f(n) > 100$

So there is no positive integer such that $n^2+n^3 = 100$

(proved)

Find the number of integers not divisible by 3, 5 or 7 which are $<= 100$

Soln Let A, B, C represents the set of integers divisible by 3, 5, 7, which are <=100

$\therefore n(A \cup B \cup C) = n(A) + n(B) + n(C)$
$- n(A \cap B) - n(A \cap C) - n(B \cap C)$
$+ n(A \cap B \cap C)$

$= 33 + 20 + 14 - 6 - 4 - 2$
$+ 0$

$= 67 - 12$
$= \cancel{45}\ 55$
$= \cancel{75}$

$\therefore$ so number of integers not divisible by 3, 5 or 7 is
$(100 - 55) = 45$ Ans

Compute the sum of palindrome numbers between 100 and 999.

Prove that the value of $n$ always can be represented as the sum of three's and eight's where value of $n \geq 14$.

Soln:

~~Let $n = 3x + 8y$ for $n$~~ $\geq 14$

$n = 3$

# Computer Architecture & Design

→ cycle time $\tau = \frac{1}{f}$ → frequency

constant cycle time

→ Average cycle per instruction

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times I_i)}{I_c}$$

→ processor time, $T = I_c \times CPI \times \tau$

→ processor time, $T = I_c \times [p + (m \times k)]\tau$

→ Million of instruction per second.

$$MIPS = \frac{I_c}{T \times 10^6}$$

$$= \frac{\tau}{CPI \times 10^6}$$

→ Millions of floating-point operations per second,

$$MFLOPS = \frac{\text{Number of floating point opc}}{\text{Execution time} \times 10^6}$$

where,

$I_c$ = total instruction count

$I_i$ = Number of executed instruction of type i

$CPI_i$ = Number of required cycle for instruction type i

T = processor time

P = Number of processor cycles needed to decode

m = number of memory references needed.

$$K = \frac{\text{memory cycle time}}{\text{processor cycle time}}$$

□ From the following table calculate CPI and MIPS.

| Instruction type | CPI | Instruction mix |
|---|---|---|
| Arithmetic & logic | 1 | 60% |
| Load/Store with catch hit | 2 | 18% |
| Branch | 4 | 12% |
| Memory ref. with catche miss | 8 | 10% |

frequency = 400MHz

Soln:

~~CP~~ Average cycle per instruction,

$$CPI = \frac{(1\times0.6)+(2\times0.18)+(4\times0.12)+(8\times0.1)}{1.00}$$

$$= 0.6+0.36+0.48+0.8$$

$$= 2.24$$ Ans

$$MIPS = \frac{I_C}{T \times CPI \times 10^6}$$

$$= \frac{I_C}{\cancel{I_C \times \frac{1}{f}}}$$

$$= \frac{\cancel{I_C}}{\cancel{I_C} \times f \times CPI \times 10^6}$$

$$= \frac{f}{CPI \times 10^6}$$

$$= \frac{400 \times \cancel{10^6}}{2.24 \times \cancel{10^6}}$$

$= 178.57$

$\simeq 178$ <u>Ans</u>

Consider two different machines with two different instruction sets, both of which have Clock rate of 200 MHz. Now from the following table determine effective CPI, MIPS rate and execution time for each machine.

| Instruction type | Instruction count (million) | Cycle per instruction |
|---|---|---|
| Machine A | | |
| Arithmetic & logic | 8 | 1 |
| Load & Store | 4 | 3 |
| Branch | 2 | 4 |
| others | 4 | 3 |
| Machine B | | |
| Arithmetic & logic | 10 | 1 |
| Load & Store | 8 | 2 |
| Branch | 2 | 4 |
| others | 4 | 3 |

Soln:

$$CPI_A = \frac{\{(8\times1)+(4\times3)+(2\times4)+(4\times3)\}\cancel{\times10^6}}{(8+4+2+4)\cancel{\times10^6}}$$

$$= \frac{8+12+8+12}{18}$$

$$= 40/18 = 2.22$$

Ans

$$CPI_B = \frac{\{(10\times1)+(8\times2)+(2\times4)+(4\times3)\}}{(10+8+2+4)}$$

$$= \frac{10+16+8+12}{24}$$

$$= \frac{46}{24}$$

$$= 1.92$$

Ans

$MIPS_A = \frac{I}{CPI_A \times 10^6}$

$= \frac{200 \times \cancel{10^6}}{2.22 \times \cancel{10^6}}$

$= 90.090$

$\simeq 90$ An

$MIPS_B = \frac{I}{CPI_B \times 10^6}$

$= \frac{200 \times \cancel{10^6}}{1.92 \times \cancel{10^6}}$

$= 104.17$

$\simeq 104$ An

execution $time_A = (8+4+2+4) \times \cancel{10^6} \times 2.22 \times \frac{1}{200 \times \cancel{10^6}}$

$= 18 \times 2.22 \times \frac{1}{200}$

$= 0.1998$ s An

execution $time_B$

$= (10+8+2+4) \times \cancel{10^6} \times 1.92 \times \frac{1}{200 \times \cancel{10^6}}$

$= \frac{24 \times 1.92}{200}$

$= 0.2304$ An

Q) From the following table calculate CPI

| Type | Instruction mix | Clock cycle |
| --- | --- | --- |
| Loads | 25% | 5 |
| Stores | 10% | 4 |
| ALU inst. | 52% | 4 |
| Branches | 11% | 3 |
| Jumps | 2% | 3 |

Soln: $CPI = (0.25 \times 5) + (0.1 \times 4) + (0.52 \times 4) + (0.11 \times 3) + (0.02 \times 3)$

$= 1.25 + 0.4 + 2.08 + 0.33 + 0.06$

$= 4.12$ Ans

Suppose we have the following instruction mix data and the cycle per instruction in multi-cycle architecture. How faster it is than a single cycle version.

| Type | Instruction mix | Clock cycle |
|---|---|---|
| Memory | 30% | 6 |
| ALU instr | 60% | 4 |
| Jumps | 10% | 2 |

Soln: Average cycle per instruction, $CPI = \frac{(0.3\times6)+(0.6\times4)+(0.1\times2)}{1.0}$

$= 1.8 + 2.4 + 0.2$

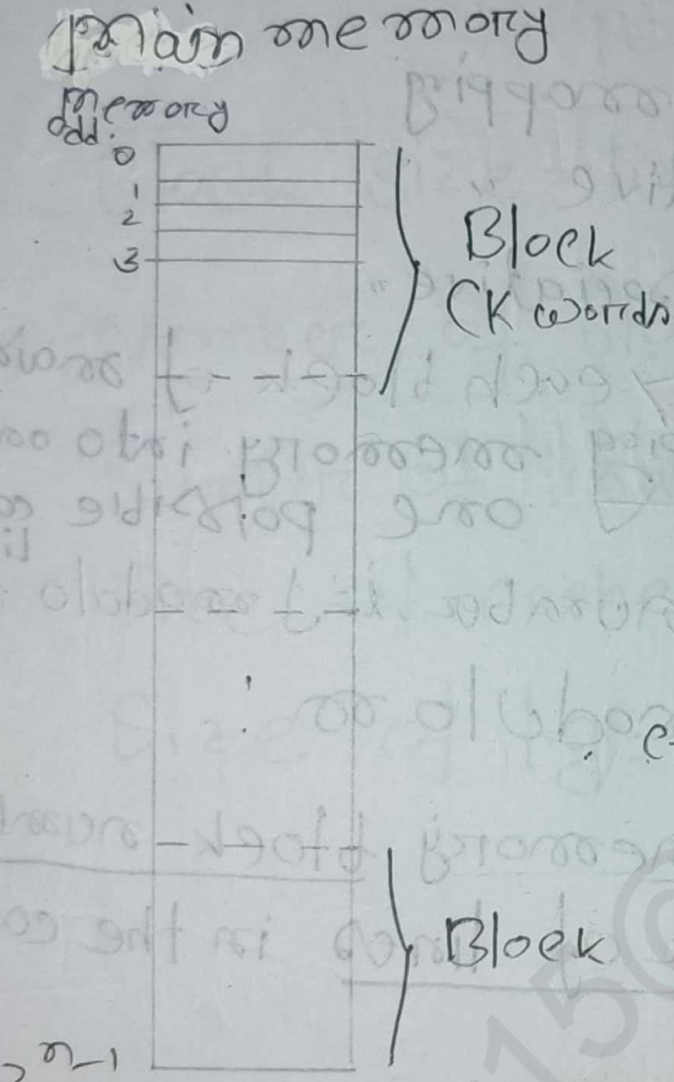
$= 4.4$

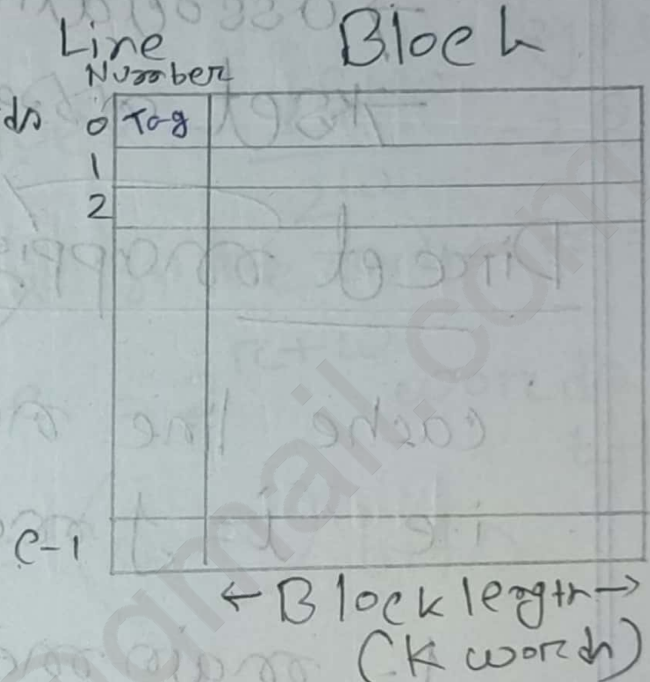
$$\therefore MIPS = \frac{I}{CPI \times 10^6}$$

$$= \frac{4}{4.4 \times 10^6}$$

$$=$$

# Cache / Main memory structure




word size = n bit

# words = $2^n$

# Block size = K words

$\therefore$ # Block, $M = \frac{2^n}{K}$

Cache consists of
→ m blocks → lines
→ each line contains
K words
tag
control bits

mapping main memory block into cache lines

## Cache mapping techniques:

3 types
→ direct mapping
→ associative "
→ set associative "

**Direct mapping** each block of main memory into only one possible cache lines.

cache line number, $i = j$ modulo $m$

i.e $i = j$ modulo $m$

$j$ = main memory block number

$m$ = number of lines in the cache

Address length = $(s+w)$ bits

Number of addressable units = $2^{s+w}$ words / bytes

Block size = line size = $2^w$ words or bytes

Number of block = $\frac{2^{s+w}}{2^w} = 2^s$

Number of lines in the cache, $m = 2^r$

Size of cache = $2^{r+w}$ words or bytes.

Size of tg = $(s-r)$ bits

each main memory block to be loaded into any line of the cache.

## Associative mapping

Address length = $(S+w)$ bits

Number of addressable units $= 2^{S+w}$ words or bytes

Block size = line size $= 2^{w}$ words

Number of block in main memory $= \frac{2^{S+w}}{2^{w}} = 2^{S}$

Number of lines in the cache = undetermined

Size of tag = $S$ bits

exhibits the strengths of both the direct and associative

## k-way set associative mapping:

$$m = v \times k$$

$$i = j \text{ modulo } v$$

where,
$i$ = cache set number
$j$ = main memory block number
$v$ = Number of sets
$k$ = " of lines in each set
$m$ = " of " in the cache

Address length = $(s+w)$ bits

Number of addressable units = $2^{s+w}$ words or bytes.

Block size = line size = $2^{w}$ words or bytes.

Number of blocks in main memory $= \frac{2^{s+w}}{2^{w}} = 2^{s}$

Number of lines in Sd = K

Number of sets = v = $2^d$

Number of lines in cache = m

= Kv = $K \times 2^d$

Size of cache = $K \times 2^{d+w}$ words or bytes.

Size of tag = (s-d) bits.