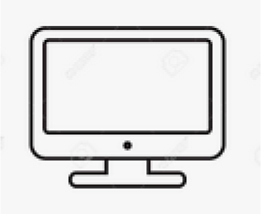
Data Representation & Boolean Algebra

- Number system
- Different Number system
- Base / Radix
- MSD / LSD
- Number Conversion
- Binary Arithmetic
- Representation of Numbers
- Representation of Characters
- Boolean Algebra
- Demorgan’s Theorm

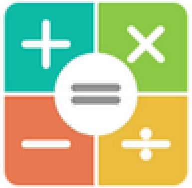
Number system

- Systematic way to represent numbers in different ways

Hello world



01001000 01100101 01101100 01101100 01101111 00100000 01110111 01101111 01110010 01101100 01100100



72 101 108 108 111 32 87 111 114 108 100

Different types of Number system

i. **Binary** Number system
   – 0, 1

ii. **Octal** Number system
   – 0, 1, 2, 3, 4, 5, 6, 7

iii. **Decimal** Number system
   – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

iv. **Hexadecimal** Number system
   – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Base / Radix

- The number of symbols used in a number system is called Base / Radix of a number system

- Eg. $(1301)_8$

- Identify the number system

$(1102)_{16}$ ⇨ Hexadecimal

$(1101)_{10}$ ⇨ Decimal

$(6327)_{8}$ ⇨ Octal

MSD & LSD

- **MSD** : Left most digit of a number is called MSD

  (Most Significant Digit)

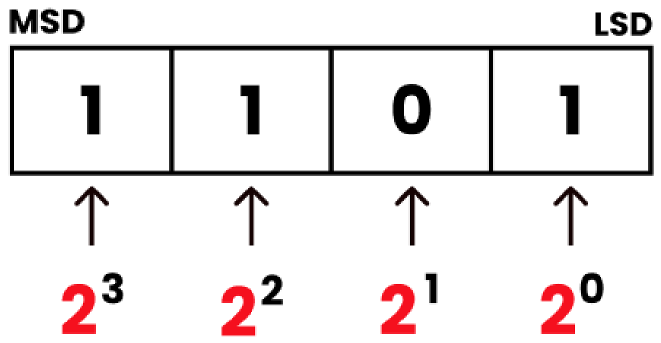
- **LSD** : Right most digit of a number is called LSD

  (Least Significant Digit)

- Binary Number system

- Base : 2
- Symbols : 0 1
- Also called BIT
- Eg : $(1101)_2$

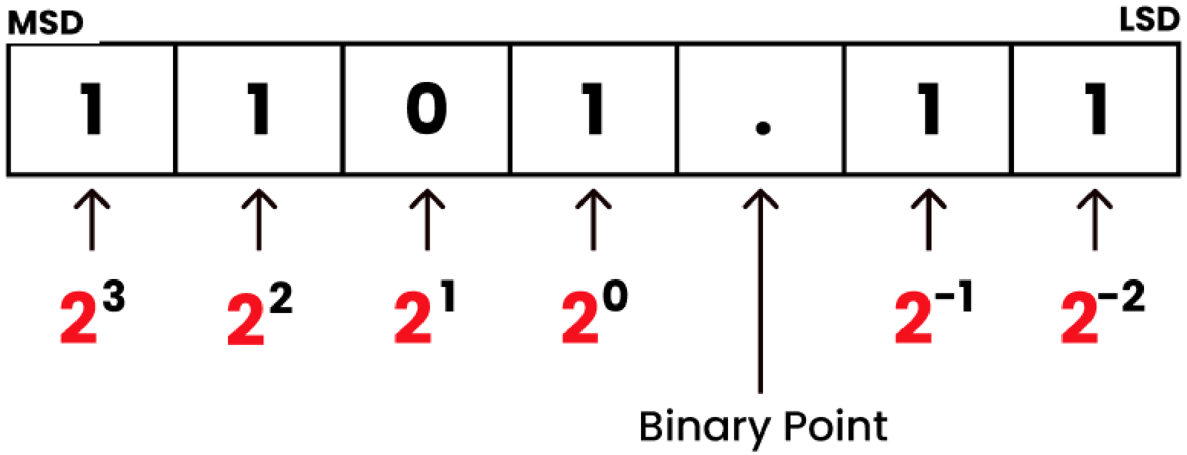
- Position Representation



- Binary Number system

- Eg : $(1101.11)_2$

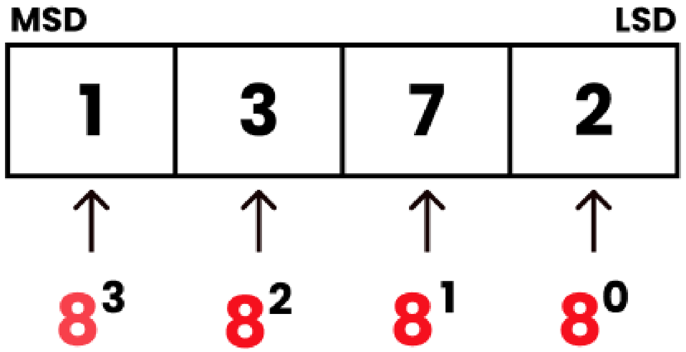
- Position Representation



- Octal Number system

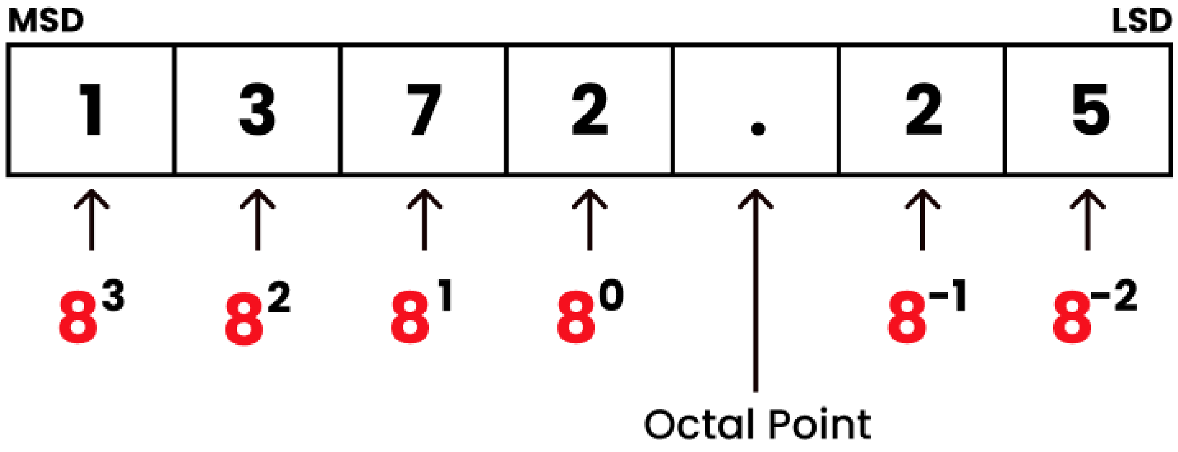
- Base : 8
- Symbols : 0 1 2 3 4 5 6 7
- Eg : $(1372)_8$

- Position Representation



- Octal Number system

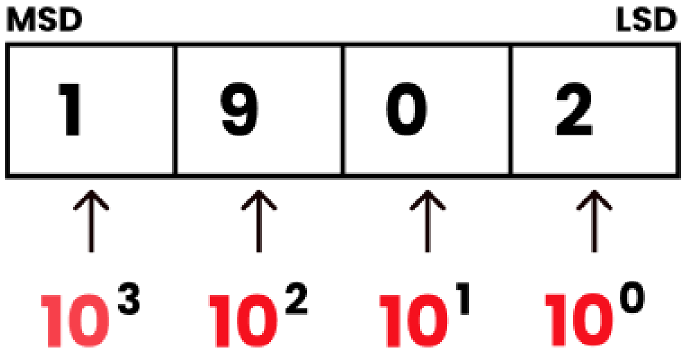
- Eg : $(1372 . 25)_8$
- Position Representation



- Decimal Number system

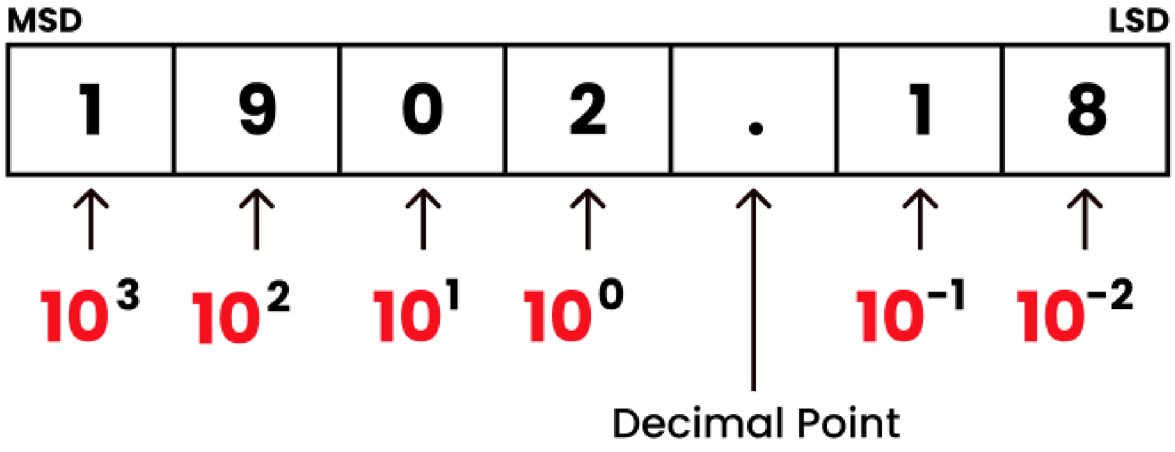
- Base : 10
- Symbols : 0 1 2 3 4 5 6 7 8 9
- Eg : $(1902)_{10}$

- Position Representation



- Decimal Number system

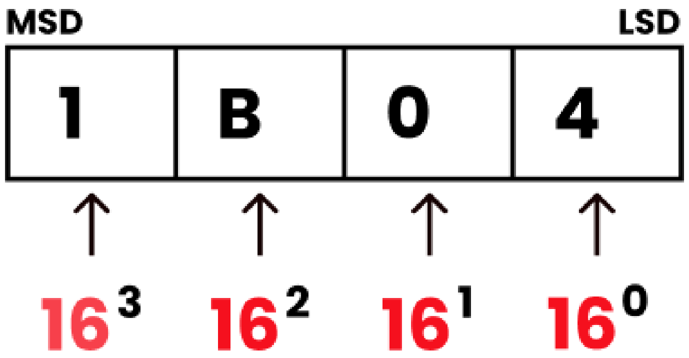
- **Eg : (1902 . 18)$_{10}$**
- **Position Representation**



- Hexadecimal Number system

- Base : 16
- Symbols : 0 1 2 3 4 5 6 7 8 9 A B C D E F
- Eg : $(1B04)_{16}$

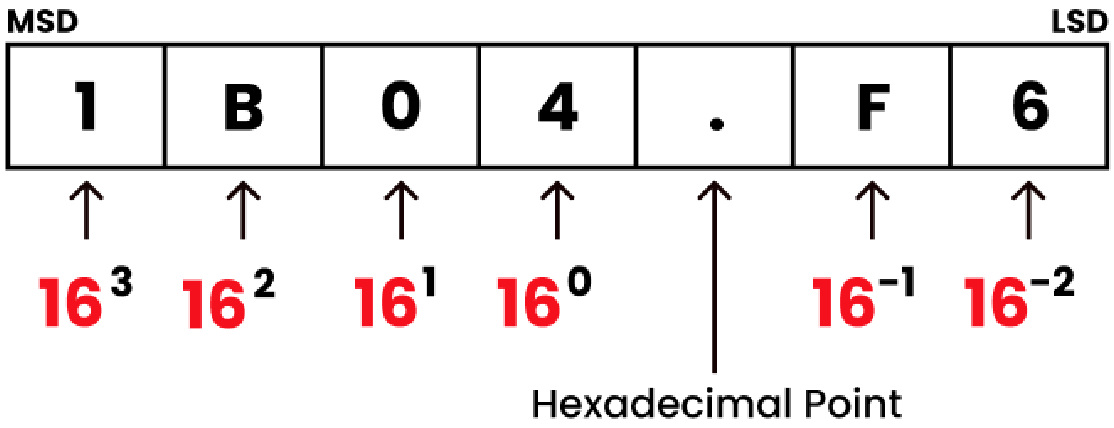
- Position Representation



- Hexadecimal Number system

- Eg : $(1B04\ .\ F6)_{16}$

- Position Representation



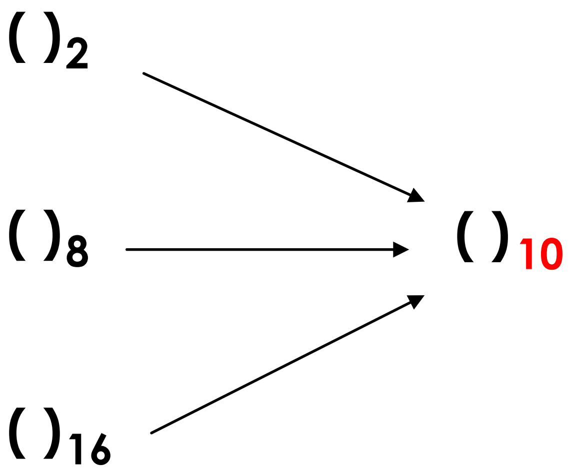
Number Conversion

Convert one number system to another number system

- Binary to Decimal
- Octal to Decimal
- Hexadecimal to Decimal
- Decimal to Binary
- Decimal to Octal
- Decimal to Hexadecimal
- Octal to Binary
- Hexadecimal to Binary
- Binary to Octal
- Binary to Hexadecimal
- Octal to Hexadecimal
- Hexadecimal to Octal

- **Binary to Decimal**
- **Octal to Decimal**
- **Hexadecimal to Decimal**

- Multiplication Method



- Decimal to Binary
- Decimal to Octal
- Decimal to Hexadecimal

- Repeated Division Method

$(\ )_2 \leftarrow (\ )_{10}$

$(\ )_8 \leftarrow (\ )_{10}$

$(\ )_{16} \leftarrow (\ )_{10}$

- **Octal to Binary**
- **Hexadecimal to Binary**

- Conversion Method

$$(\ )_8 \rightarrow (\ )_2$$

$$(\ )_{16} \rightarrow (\ )_2$$

- **Binary to Octal**
- **Binary to Hexadecimal**

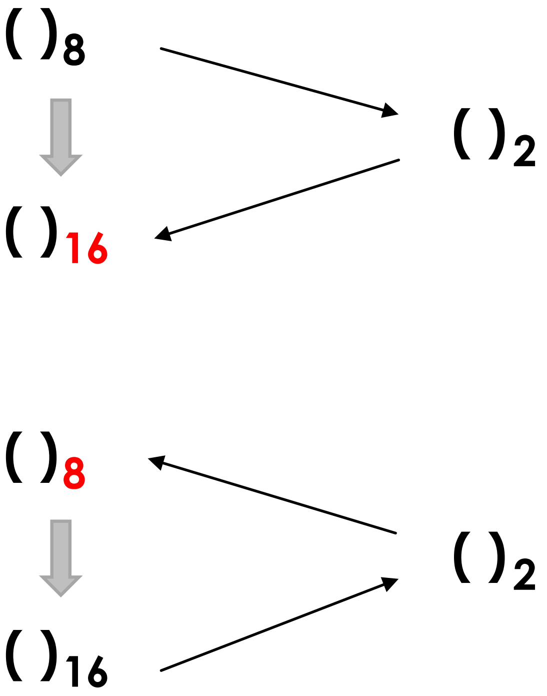
$$(\ )_8 \leftarrow (\ )_2$$

$$(\ )_{16} \leftarrow (\ )_2$$

- Grouping Method

- Octal to Hexadecimal
- Hexadecimal to Octal

  - Covert to Binary
    then Binary to Number system

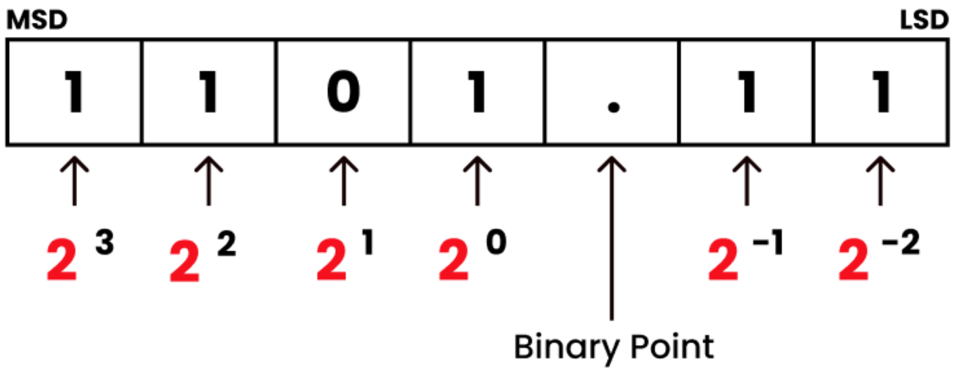


Number Conversion
Lets Starts…

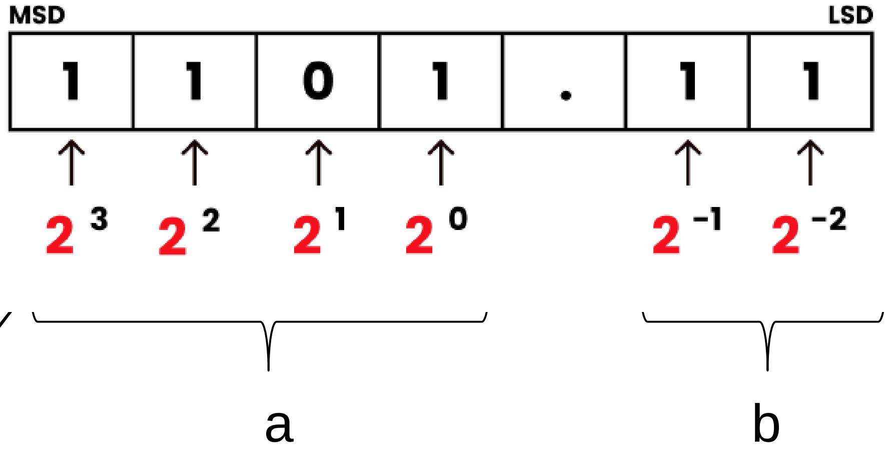
- Binary to Decimal Conversion

- **Multiply each bits by its positional value and add**
- **Eg : convert $(1101.11)_2$ to Decimal.**

- **Position Representation**



- Binary to Decimal Conversion



$$a = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$
$$= 8 + 4 + 0 + 1$$
$$= 13$$

$$b = 1 \times 2^{-1} + 1 \times 2^{-2}$$
$$= 0.5 + 0.25$$
$$= 0.75$$

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

- Octal to Decimal Conversion

- **Multiply each octal digit by its positional value and add**
- **Eg : convert $(256.11)_8$ to Decimal.**

- **Position Representation**

2	5	6	.	1	1
8^2	8^1	8^0	Octal Point	8^{-1}	8^{-2}

- Octal to Decimal Conversion

2	5	6	.	1	1
8^2	8^1	8^0		8^{-1}	8^{-2}

$$\underbrace{8^2 \quad 8^1 \quad 8^0}_{a} \qquad \underbrace{8^{-1} \quad 8^{-2}}_{b}$$

$$a = 2 \times 8^2 + 5 \times 8^1 + 6 \times 8^0$$
$$= 64 + 40 + 9$$
$$= 113$$

$$b = 1 \times 8^{-1} + 1 \times 8^{-2}$$
$$= 0.125 + 0.0156$$
$$= 0.1406$$

$$(256.11)_8 = (113.1406)_{10}$$

- Hexadecimal to Decimal Conversion

- **Multiply each hexadecimal digit by its positional value and add**
- **Eg : convert $(3A6.13)_{16}$ to Decimal.**

- **Position Representation**

3	A	6	.	1	3
16^2	16^1	16^0	Hexadecimal Point	16^{-1}	16^{-2}

- Hexadecimal to Decimal Conversion

3	A	6	.	1	3

$$\underbrace{16^{2} \quad 16^{1} \quad 16^{0}}_{a} \qquad \underbrace{16^{-1} \quad 16^{-2}}_{b}$$

$$a = 3 \times 16^{2} + (10) \times 16^{1} + 6 \times 16^{0}$$

$$= 768 + 160 + 6$$

$$= 934$$

$$b = 1 \times 16^{-1} + 3 \times 16^{-2}$$

$$= 0.0625 + 0.004$$

$$= 0.0665$$

$$(256.11)_{16} = (934.0665)_{10}$$

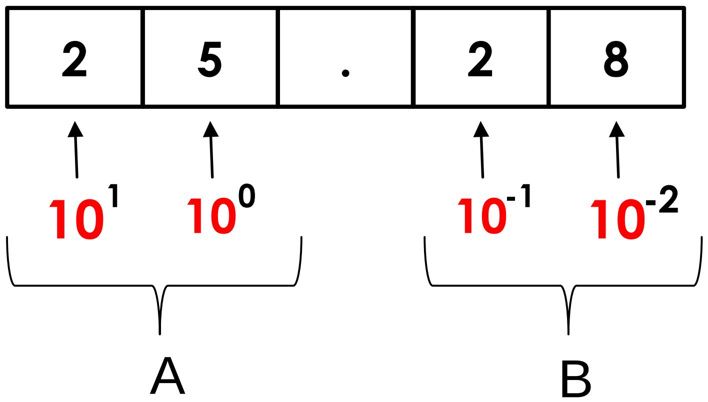
- Decimal to Binary Conversion

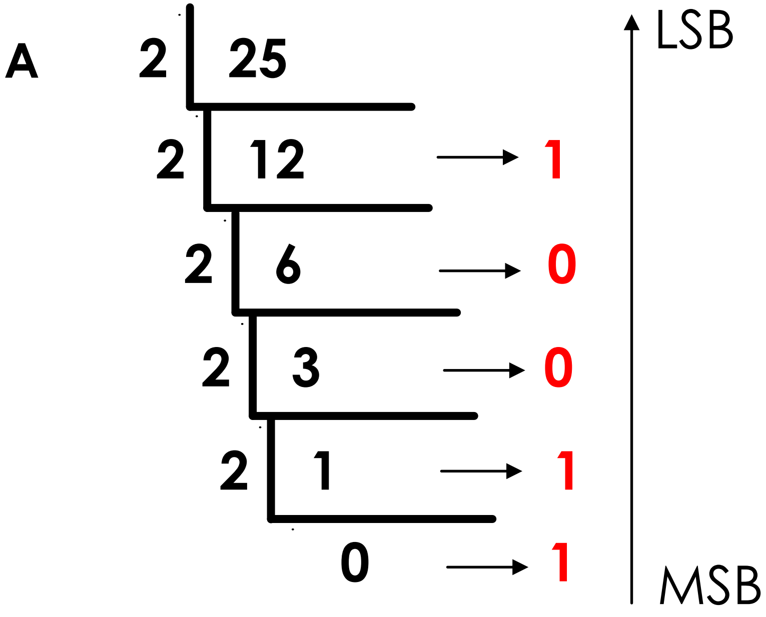
- **Repeatedly dividing it by 2, until the quotient is zero**
- **Reminder generate each division form binary number**
- **Eg : convert $(25\ .\ 28)_{10}$ to Binary.**

- **Position Representation**

2	5	.	2	8
10^{1}	10^{0}	Decimal Point	10^{-1}	10^{-2}

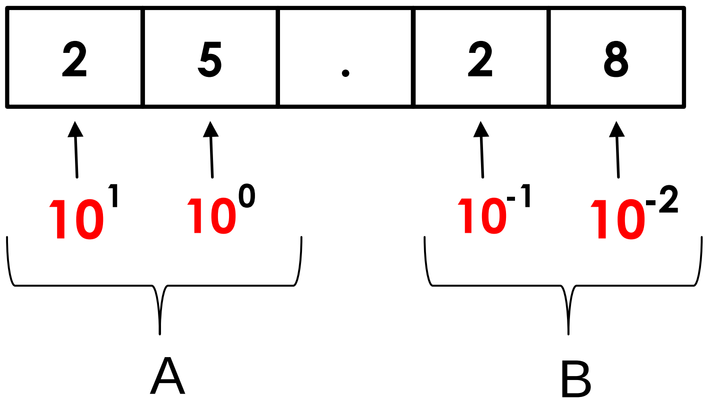
- Decimal to Binary Conversion





A = 11001

- Decimal to Binary Conversion



- Fractional part conversion by multiply with 2

B

$$0.28 \times 2 = 0.56 \rightarrow 0 \quad \text{MSB}$$
$$0.56 \times 2 = 1.12 \rightarrow 1$$
$$0.12 \times 2 = 0.24 \rightarrow 0$$
$$0.24 \times 2 = 0.48 \rightarrow 0$$
$$0.48 \times 2 = 0.96 \rightarrow 0 \quad \text{LSB}$$

B = 01000

$$(25.28)_{10} = (11001.01000)_2$$

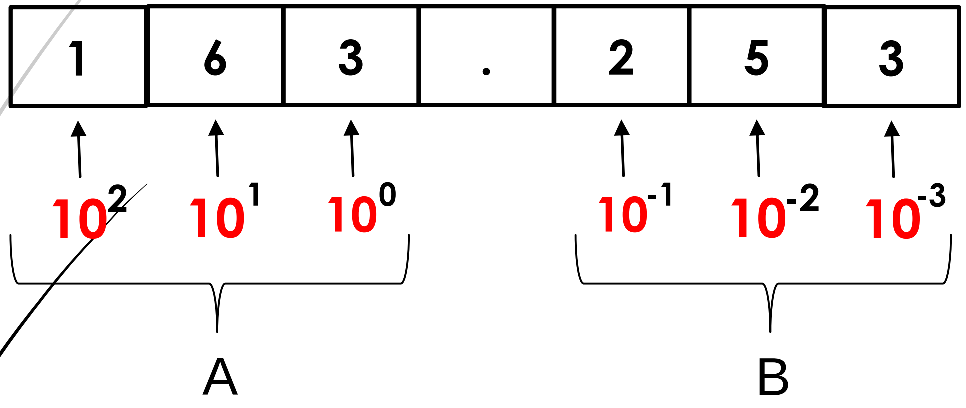
- Decimal to Octal Conversion

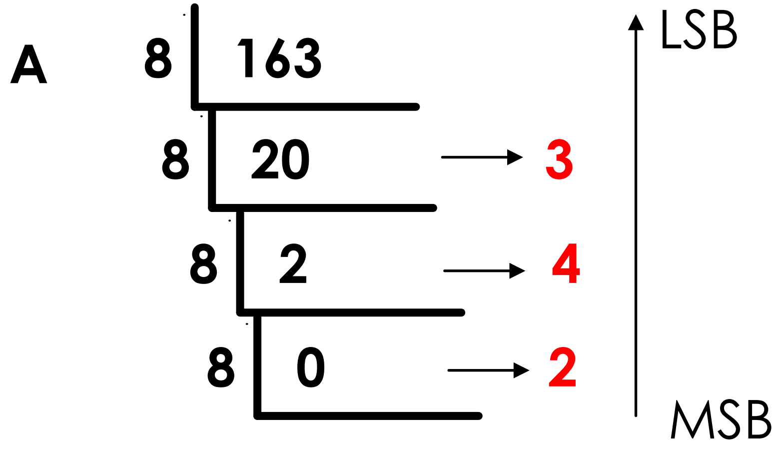
- **Repeatedly dividing it by 8, until the quotient is zero**
- **Reminder generate each division form Octal number**
- **Eg : convert $(163 \,.\, 253)_{10}$ to Octal.**

- **Position Representation**

1	6	3	.	2	5	3
10^2	10^1	10^0	Decimal Point	10^{-1}	10^{-2}	10^{-3}

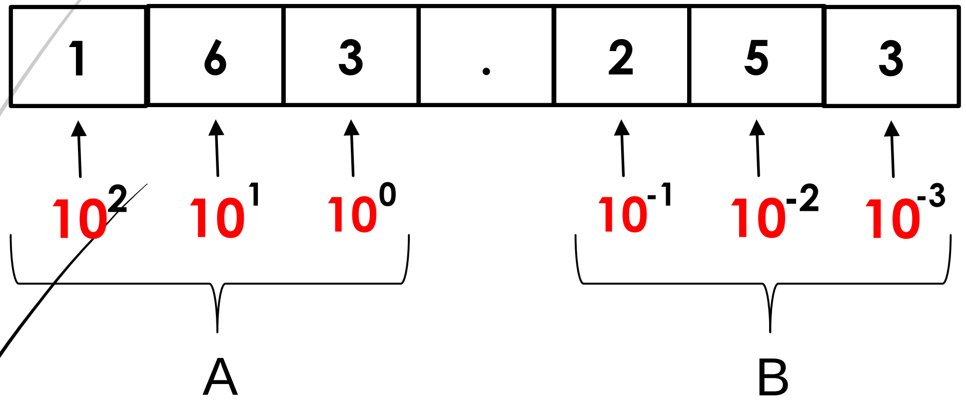
- Decimal to Octal Conversion





A = 243

- Decimal to Octal Conversion



- Fractional part conversion by multiply with 8

B

0.253	**x**	**8**	**=**	**2.024**	→	**2**
0.024	**x**	**8**	**=**	**0.192**	→	**0**
0.192	**x**	**8**	**=**	**1.536**	→	**1**
0.536	**x**	**8**	**=**	**4.288**	→	**4**
0.288	**x**	**8**	**=**	**2.304**	→	**2**

MSB ↓ LSB

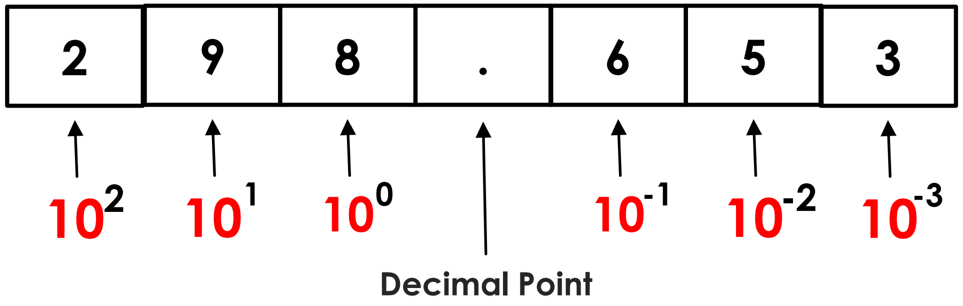
B = 20142

$$(163.253)_{10} = (243\ .\ 20142\ )_8$$

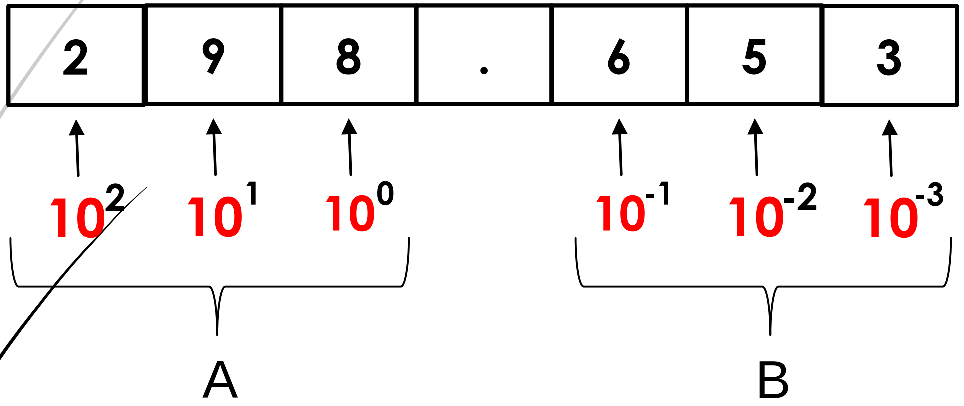
- Decimal to Hexadecimal Conversion

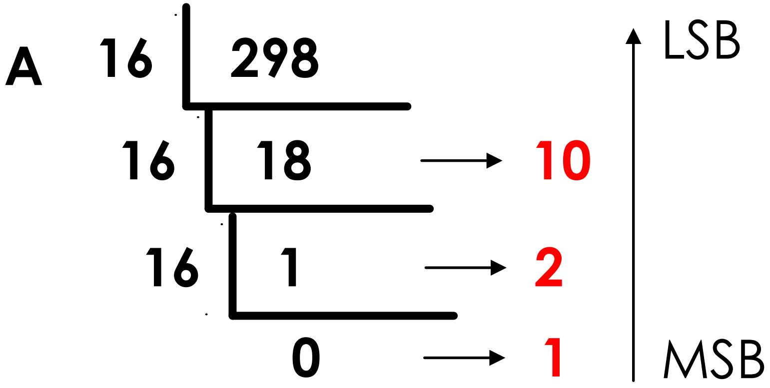
- **Repeatedly dividing it by 16, until the quotient is zero**
- **Reminder generate each division form Hexadecimal number**
- **Eg : convert $(298\ .\ 653)_{10}$ to Octal.**

- **Position Representation**



- Decimal to Hexadecimal Conversion





$$A = 12\ (10)$$
$$= 12A$$

- Decimal to Hexadecimal Conversion

2	9	8	.	6	5	3
10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}

$$\underbrace{10^2 \quad 10^1 \quad 10^0}_{A} \qquad \underbrace{10^{-1} \quad 10^{-2} \quad 10^{-3}}_{B}$$

- Fractional part conversion by multiply with 16

B

MSB

$$0.653 \times 16 = 10.448 \rightarrow 10$$
$$0.448 \times 16 = 7.168 \rightarrow 7$$
$$0.168 \times 16 = 2.668 \rightarrow 2$$
$$0.668 \times 16 = 11.008 \rightarrow 11$$

LSB

B = (10) 7 2 (11)

= A72B

$$(298\ .\ 653)_{10} = (12A\ .\ A72B)_{16}$$

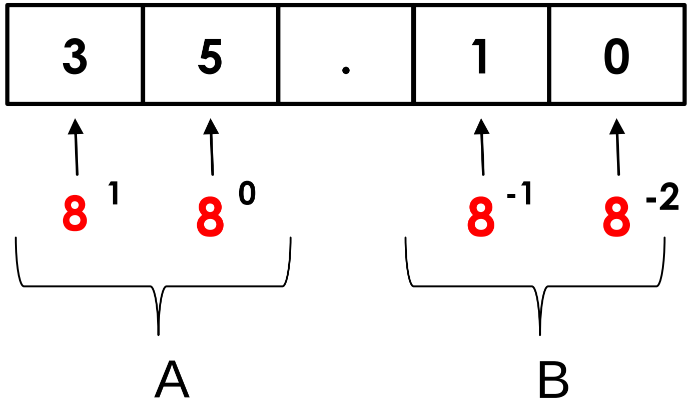
- Octal to Binary Conversion

- Covert each Octal digit into its 3 bit binary equivalent

Octal	Binary Equivalent
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

- Octal to Binary Conversion

- **Eg : Convert $(35 \,.\, 10)_8$ to Binary**



A

3 5 . 1 0

B

011 101 . 001 000

11 101 . 001

$$(35.10)_{10} = (11101.\,001)_2$$

- Hexadecimal to Binary Conversion

- **Covert each Hexadecimal digit into its 4 bit binary equivalent**

Hexadecimal	Binary Equivalent
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

- Hexadecimal to Binary Conversion

- **Eg : Convert $(6A\ .\ 21)_{16}$ to Binary**

6	**A**	**.**	**2**	**1**
16^{1}	16^{0}		16^{-1}	16^{-2}

$$\underbrace{16^{1}\quad 16^{0}}_{A}\qquad\qquad \underbrace{16^{-1}\quad 16^{-2}}_{B}$$

A

6 A . 2 1

B

0110 1010 . 0010 0001

110 1010 . 0010 0001

$$(6A.21)_{16} = (1101010\ .\ 00100001)_2$$

- Binary to Octal Conversion

- **Covert 3 bit Binary number into corresponding Octal equivalent**

Binary	Octal Equivalent
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

- Binary to Octal Conversion

- **Eg : Convert $(1100110\ .01)_2$ to Octal**

1	100	110	.	01
001	100	110	.	010
1	4	6	.	2

$$(1100110\ .01)_2 = (146.2)_8$$

- Binary to Hexadecimal Conversion

- Covert 3 bit Binary number into corresponding Octal equivalent

Binary	Hexadecimal Equivalent
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

- Binary to Hexadecimal Conversion

- **Eg : Convert $(1100110\ .01)_2$ to Hexadecimal**

110	0110	.	01
←	←		→
0110	0110	.	0100
←	←		→
6	6	.	4

$$(1100110\ .01)_2 = (66\ .\ 4)_{16}$$

- Octal to Hexadecimal Conversion

- Convert octal number into equivalent binary,
- then convert binary to hexadecimal
- Eg : Convert $(67.4)_8$ to Hexadecimal

- Octal to Binary

6	7	.	4
110	111	.	100

- Binary to Hexadecimal

11	0111	.	100
0011	0111	.	1000
3	7	.	8

$$(67.4)_8 = (37.8)_{16}$$

- Hexadecimal to Octal Conversion

- Convert Hexadecimal number into equivalent binary,
- then convert binary to Octal
- Eg : Convert $(B6.F)_{16}$ to Octal

- Hexadecimal to Binary

B	6	.	F
1011	0110	.	1111

- Binary to Octal

10	110	110	.	111	1
010	110	110	.	111	100
2	6	6	.	7	4

$(B6.F)_{16} = (266.74)_8$

Binary Arithmetic

- **Binary Addition**
- **Binary Subtraction**

- Binary Addition

- Carry bit 1 is created only two 1s are added
- Three 1s are added ,then sum is 1 and carry is 1

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$\begin{array}{r} 0\ + \\ 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0\ + \\ 1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1\ + \\ 1 \\ 1 \\ \hline 11 \end{array}$$

$$\begin{array}{r} 1\ + \\ 0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1\ + \\ 1 \\ \hline 10 \end{array}$$

- Eg : Find sum of binary numbers 1011 and 1001

$$
\begin{array}{r}
1\,1 \\
1\,0\,1\,1\;+ \\
1\,0\,0\,1 \\
\hline
1\,0\,1\,0\,0
\end{array}
$$

- Eg : Find sum of binary numbers 110111 and 111

$$
\begin{array}{r}
\color{red}{1\,1\,1} \\
1\,1\,0\,1\,1\,1\,+ \\
1\,1\,1 \\
\hline
1\,1\,1\,1\,1\,0
\end{array}
$$

- Binary Subtraction

- **Borrow bit 1 is created only 1 subtracted from 0**

A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

- Eg : Subtract 10101 from 11111

$$
\begin{array}{r}
1\ 1\ 1\ 1\ 1\ - \\
1\ 0\ 1\ 0\ 1 \\
\hline
0\ 1\ 0\ 1\ 0
\end{array}
$$

- Eg : Subtract 011 from 101

$$\begin{array}{rrrr} 0 & 2 & & \\ \cancel{1} & 0 & 1 & - \\ 0 & 1 & 1 & \\ \hline 0 & 1 & 0 & \end{array} \quad \begin{array}{c} \longrightarrow \\ \longrightarrow \\ \\ \end{array} \quad \begin{array}{rr} 5 & - \\ 3 & \\ \hline 2 & \end{array}$$

- Eg : Subtract 10111 from 101000

$$
\begin{array}{cccccc}
 & & & & 1 & 2 \\
 & & & 1 & \cancel{2} & \\
0\,2 & & 0 & \cancel{2} & & \\
\cancel{1} & 0 & \cancel{1} & 0 & 0 & 0 \quad - \\
 & 1 & 0 & 1 & 1 & 1 \\
\hline
0 & 1 & 0 & 0 & 0 & 1
\end{array}
$$

Representation of Numbers

- **Numbers or integers represented in computer are**

  - **Sign and Magnitude Representation (SM)**
  - **1’s Compliment Representation**
  - **2’s Compliment Representation**

- Sign and Magnitude Representation

- **It is used to represent signed numbers**
- **It consist of sign part and magnitude part**
- **Example : - 32 +51 - 7**
- **A Computer word size is 1 byte (8 bit)**
- **The 7th bit (MSD) is used for representing sign of a number**

Sign Part	Magnitude Part

MSD LSD

0	0	0	0	0	0	0	0

Sign

Magnitude

- If 7^{th} bit is **Zero** , it indicate the number is **+ve**
- If 7^{th} bit is **One** , it indicate the number is **-ve**

0	0	0	0	0	1	0	1

+ **5**

+5

1	0	0	0	0	1	0	1

- **5**

-5

- 1's Compliment Representation

- **Another method of representing Negative numbers**
- **-ve number indicate its compliment**

0	0	0	0	0	1	0	1	**+5**

↓ 1's Compliment

1	1	1	1	1	0	1	0	**-5**

- 2's Compliment Representation

- **Another method of representing Negative numbers**
- **Just add 1 with 1's compliment**

	0	0	0	0	0	1	0	1	**+5**
1's Compliment	1	1	1	1	1	0	1	0	
								1	
2's Compliment	1	1	1	1	1	0	1	1	**-5**

Representation of Characters

- **Symbolic way to represent characters in computers are called character representation**
  - **ASCII**
  - **EBCDIC**
  - **ISCII**
  - **Unicode**

ASCII

- **ASCII**
- **American Standard Code for Information interchange**
- **Use 7 bit to represent a character**
- **Example : A $\longrightarrow$ 97 (1 1 0 0 0 0 1)**
- **It can represent 128 characters**
- **$2^7 = 128$**

EBCDIC

- EBCDIC
- **Extended Binary Coded Decimal Interchange Code**
- **Similar to ASCII**
- **Use 8 bit to represent a character**
- **It can represent 256 characters**
- $2^8 = 256$

ISCII

- **ISCII**
- **Indian Standard Code for Information Interchange**
- **Or Indian Script Code for Information Interchange**
- **Use 8 bit to represent a character**
- **It can be used to represent various writing system of india**

Unicode

- Unicode
- 16 bit Code represent up to 65,536 characters
- Used to represent almost all written languages of the world
- Now a days Unicode represent more than 16 bit codes

Boolean Algebra

Boolean Algebra

- Boolean algebra is a form of mathematics that deals with statements and their values.
- It can have only two values: true or false. (0 or 1)
- The operation performed by Boolean values are called Boolean operations / logical operations
- Logical gate : physical device that perform logical operations
- It is implemented using diodes and transistors

Boolean Operations and logical gates

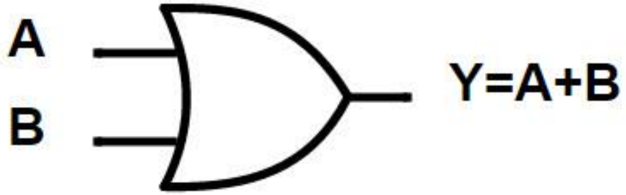
- 3 basic Boolean operations in Boolean algebra
  1. OR → Logical Addition
  2. AND → Logical Multiplication
  3. NOT → Logical Negation

Logical OR

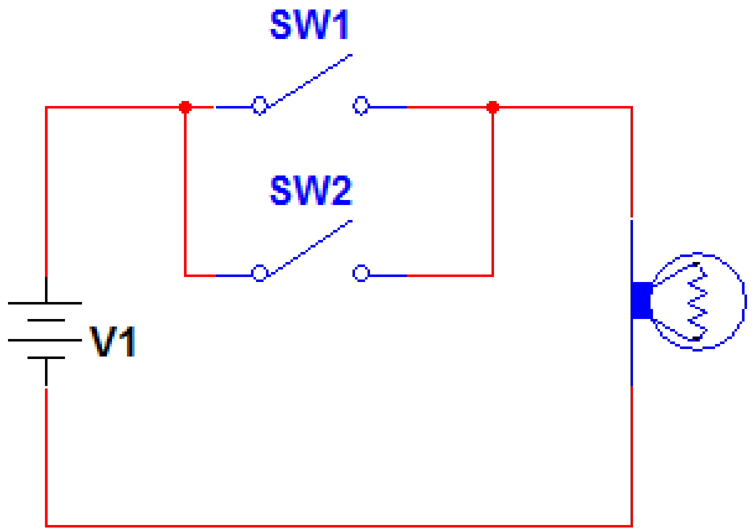
- OR operation performs logical addition
- + Plus symbol used for this operation
- A + B read as A OR B
- Truth table : (All possible operations and results)

Input A	Input B	Y = A + B (A OR B)
0	0	0
0	1	1
1	0	1
1	1	1

- OR Gate



- OR Gate Circuit



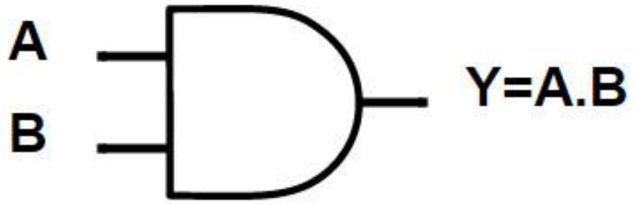
" If any one of the input is true, the total output is true "

Logical AND

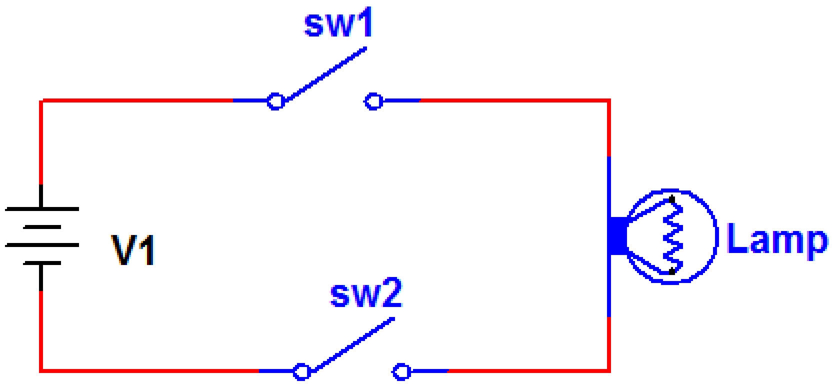
- AND operation performs logical Multiplication
- . Dot symbol used for this operation
- A . B read as A AND B
- Truth table : (All possible operations and results)

Input A	Input B	Y = A . B (A AND B)
0	0	0
0	1	0
1	0	0
1	1	1

- OR Gate



- OR Gate Circuit



“ If both of the input is true, the total output is true ”

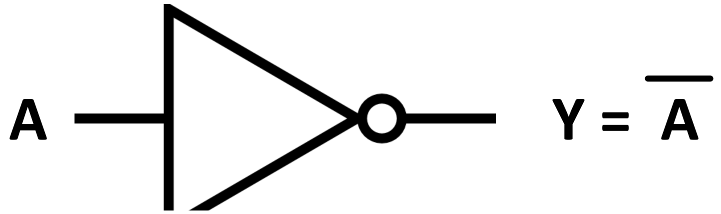
Logical NOT

- NOT operation performs logical Nagation
- $\overline{}$ over bar symbol used for this operation
- $\overline{A}$ read as A NOT
- Truth table : (All possible operations and results)

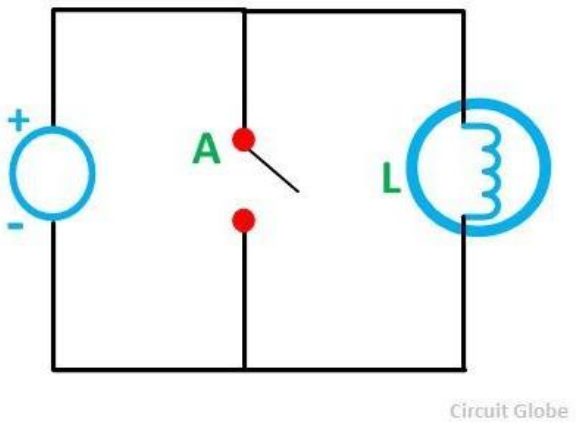
Input A	$Y = \overline{A}$
0	1
1	0

- Also called inverter

- OR Gate



- OR Gate Circuit



" output give the complement of input"

Principle of Duality

- A Boolean relation can be written to another Boolean relation by changing each OR operation to AND operation vice versa
- Example : A (B + C) = AB . AC

- If the statement is true for an expression, then it is also true for the dual of the expression

Operator / Variable	Dual
AND	OR
OR	AND
1	0
0	1
A	$\overline{A}$
$\overline{A}$	A

Basic theorems of Boolean Algebra

- Standard and accepted rules in Boolean algebra
- The rules are known as axioms

  - Identity law
  - Idempotent law
  - Involution law
  - Complimentary law
  - Commutative law
  - Associative law
  - Distributive law
  - Absorption law

Identity law

- Additive identity

$0 + x = x \qquad 1 + x = 1$

$0 + x = x$	$1 + x = 1$
$0 + 0 = 0$	$1 + 0 = 1$
$0 + 1 = 1$	$1 + 1 = 1$

- Multiplicative identity

$0 . x = 0 \qquad 1 . x = x$

$0 . x = 0$	$1 . x = x$
$0 . 0 = 0$	$1 . 0 = 0$
$0 . 1 = 0$	$1 . 1 = 1$

Idempotent law

$x + x = x$

$x \cdot x = x$

if $x = 0$,

$$\frac{x + x = x}{0 + 0 = 0}$$

if $x = 1$,

$$\frac{x + x = x}{1 + 1 = 1}$$

if $x = 0$,

$$\frac{x \cdot x = x}{0 \cdot 0 = 0}$$

if $x = 1$,

$$\frac{x \cdot x = x}{1 \cdot 1 = 1}$$

Involution law

$\overline{\overline{X}} = X$

if $x = 0$,

$\overline{\overline{X}} = X$

$\overline{0} = 1$

$\overline{\overline{0}} = 0$

if $x = 1$,

$\overline{\overline{X}} = X$

$\overline{1} = 0$

$\overline{\overline{1}} = 1$

Complimentary law

$\overline{X} + X = 1$

if x = 0 ,

$\overline{x} + x = 1$

$\overline{0} + 0$

$1 + 0 = 1$

if x = 1 ,

$\overline{x} + x = 1$

$\overline{1} + 1$

$0 + 1 = 1$

$\overline{X} . X = 0$

if x = 0 ,

$\overline{x} . x = 0$

$\overline{0} . 0$

$1 . 0 = 0$

if x = 1 ,

$\overline{x} . x = 0$

$\overline{1} . 1$

$0 . 1 = 0$

Commutative law

x + y = y + x

X	Y	X + Y	Y + X
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

x . y = y . x

X	Y	X . Y	Y . X
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Associative law

$$x + (y + z) = (x + y) + z$$

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

$$x + (y + z) = (x + y) + z$$

X	Y	Z	Y + Z	X + (Y + Z)	X + Y	(X + Y) + Z
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

X	Y	Z	Y . Z	X . (Y . Z)	X . Y	(X . Y) . Z
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	1	0
1	1	1	1	1	1	1

Distributive law

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$

$$x + (y \cdot z) = (x + y) \cdot (x + z)$$

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$

X	Y	Z	Y + Z	X . (Y + Z)	X . Y	X . Z	(X . Y) + (X . Z)
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

$$x + (y \cdot z) = (x + y) \cdot (x + z)$$

X	Y	Z	Y . Z	X + (Y . Z)	X + Y	X + Z	(X + Y) . (X + Z)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Absorption law

$x + (x \cdot y) = x$

X	Y	X . Y	X + (Y . X)
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

$x \cdot (x + y) = x$

X	Y	X + Y	X . (Y + X)
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

De Morgan’s theorems

De Morgan's theorems

- The Complement of sum of Boolean variable is equal to product of their individual complements.

$$\overline{X + Y} = \overline{X} \cdot \overline{Y}$$

- The Complement of product of Boolean variable is equal to sum of their individual complements.

$$\overline{X \cdot Y} = \overline{X} + \overline{Y}$$

Proof of 1st theorem

We have to prove that, $\overline{X+Y} = \overline{X} \cdot \overline{Y}$

Let us assume that, $Z = X + Y$ ______ (1)

Then, $\overline{Z} = \overline{X+Y}$ ______ (2)

We know that, by complimentary law, the equations (3) and (4) are true.

$Z + \overline{Z} = 1$ ______ (3)

$Z \,.\, \overline{Z} = 0$ ______ (4)

Substituting expressions (1) in (3) and (2) in (4), we will get equations (5) and (6).

$(X + Y) + (\overline{X+Y}) = 1$ ______ (5)

$(X + Y) \,.\, (\overline{X+Y}) = 0$ ______ (6)

$\boxed{\overline{X + Y} = \overline{X} \,.\, \overline{Y}}$

$(X+ Y) + (\overline{X} \,.\, \overline{Y}) = 1$ ______ (7) 7

$(X + Y) \,.\, (\overline{X} \,.\, \overline{Y}) = 0$ ______ (8) 8

Equation 7 :

$$
\begin{aligned}
(X + Y) + (\overline{X} . \overline{Y}) &= (X + Y + \overline{X}) . (X + Y + \overline{Y}) && \textit{(Distributive Law)} \\
&= (X + \overline{X} + Y) . (X + Y + \overline{Y}) && \textit{(Associative Law)} \\
&= (1 + Y) . (X + 1) && \textit{(Complimentary Law)} \\
&= 1 . 1 && \textit{(Additive Identity)} \\
&= 1
\end{aligned}
$$

$$\boxed{(X + Y) + (\overline{X} . \overline{Y}) = 1 \quad \_\_\_\_\_\_\_ (7)}$$

Equation 8 :

$$
\begin{aligned}
(X+Y).(\overline{X}.\overline{Y}) &= (X.\ \overline{X}.\overline{Y}) + (Y.\overline{X}.\overline{Y}) && \textit{(Distributive Law)}\\
&= (X.\ \overline{X}.\overline{Y}) + (Y.\overline{Y}.\overline{X}) && \textit{(Associative Law)}\\
&= (0.\overline{Y}) + (0.\overline{X}) && \textit{(Complimentary Law)}\\
&= 0 + 0 && \textit{(Multiplicative Identity)}\\
&= 0
\end{aligned}
$$

$$\boxed{(X+Y).(\overline{X}.\overline{Y}) = 0 \quad \_\_\_\_\_\_\_(8)}$$

Proof of 2nd theorem

We have to prove that, $\overline{X.Y} = \overline{X} + \overline{Y}$

Let us assume that, $Z = X \cdot Y$ ______ (1)

Then, $\overline{Z} = \overline{X.Y}$ ______ (2)

We know that, by complimentary laws the equations (3) and (4) are true.

$Z + \overline{Z} = 1$ ______ (3)

$Z \cdot \overline{Z} = 0$ ______ (4)

Substituting expressions (1) in (3) and (2) in (4), we will get the expressions (5) and (6).

$(X \cdot Y) + (\overline{X.Y}) = 1$ ______ (5)

$(X \cdot Y) \cdot (\overline{X.Y}) = 0$ ______ (6)

$$\boxed{\overline{X \cdot Y} = \overline{X} + \overline{Y}}$$

$(X \cdot Y) + (\overline{X} + \overline{Y}) = 1$ ______ (7) 7

$(X \cdot Y) \cdot (\overline{X} + \overline{Y}) = 0$ ______ (8) 8

Equation 7 :

$$
\begin{aligned}
(X \,.\, Y) + (\overline{X} + \overline{Y}) &= (\overline{X} + \overline{Y}) + (X \,.\, Y) && \textit{(Commutative Law)} \\
&= (\overline{X} + \overline{Y} + X) \,.\, (\overline{X} + \overline{Y} + Y) && \textit{(Distributive Law)} \\
&= (\overline{X} + X + \overline{Y}) \,.\, (\overline{X} + \overline{Y} + Y) && \textit{(Associative Law)} \\
&= (1 + \overline{Y}) \,.\, (\overline{X} + 1) && \textit{(Complimentary Law)} \\
&= 1 \,.\, 1 && \textit{(Additive Identity)} \\
&= 1
\end{aligned}
$$

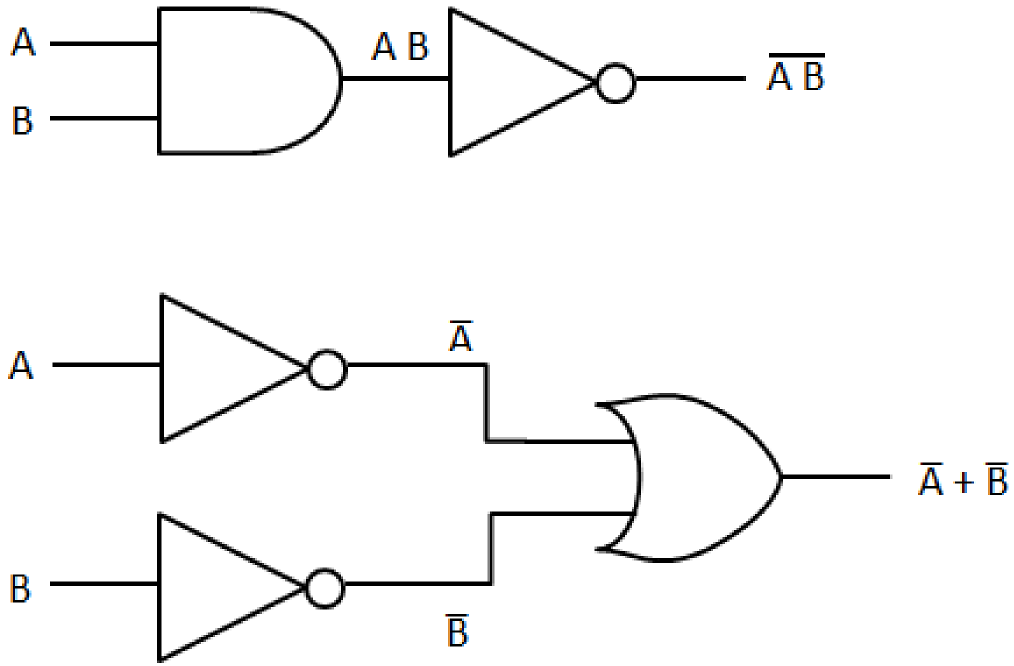
$$\boxed{(X \,.\, Y) + (\overline{X} + \overline{Y}) = 1 \quad \text{\_\_\_\_\_\_} \quad (7)}$$

Equation 8 :

$$
\begin{aligned}
(X.Y).(\overline{X}+\overline{Y}) &= (X.Y.\overline{X}) + (X.Y.\overline{Y}) && \textit{(Distributive Law)}\\
&= (X.\overline{X}.Y) + (X.Y.\overline{Y}) && \textit{(Associative Law)}\\
&= (0.Y) + (X.0) && \textit{(Complimentary Law)}\\
&= 0 + 0 && \textit{(Multiplicative Identity)}\\
&= 0
\end{aligned}
$$

$$\boxed{(X.Y).(\overline{X}+\overline{Y}) = 0 \qquad \text{\_\_\_\_\_\_\_\_} \quad (8)}$$

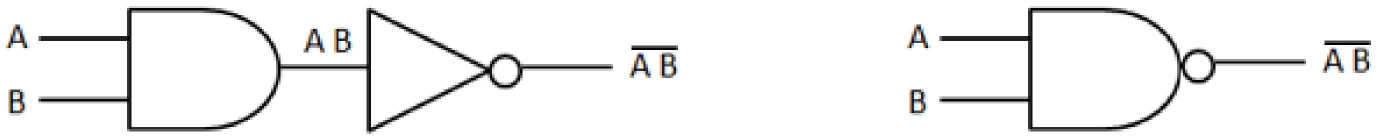
Logic Circuits



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

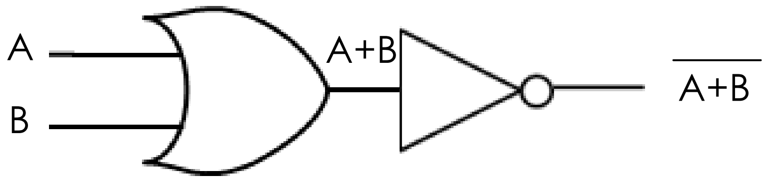
Universal gate

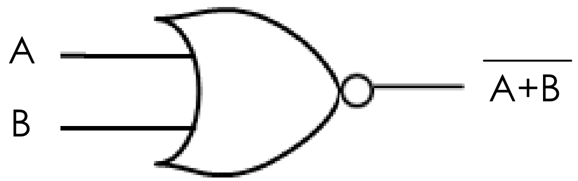
- Universal gate is a gate which can implement any other Boolean function without using any other gates
- NAND gate and NOR gate are called universal gate.
- NAND = Noted AND gate



Universal gate

- NOR = Noted OR gate





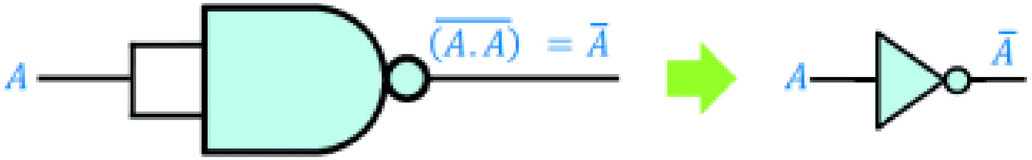


Fig. 2.20 : NOT gate using NAND gate

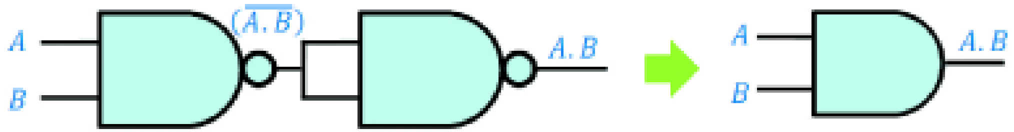


Fig. 2.21 : AND gate using NAND gate

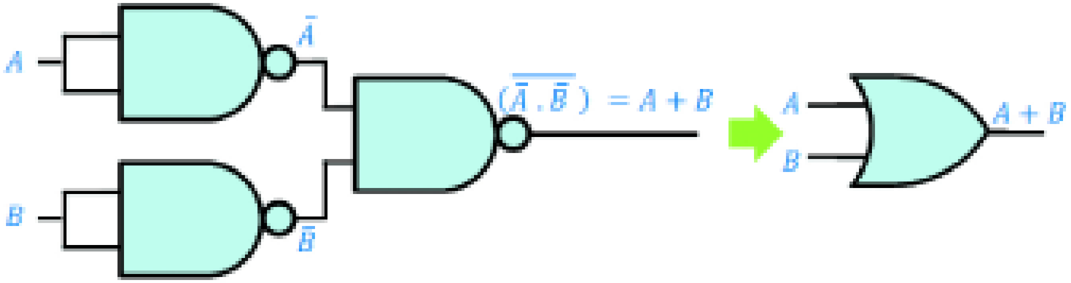


Fig. 2.22 : OR gate using NAND gate

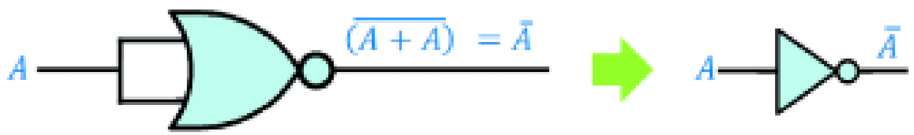


Fig 2.23 : NOT gate using NOR gate

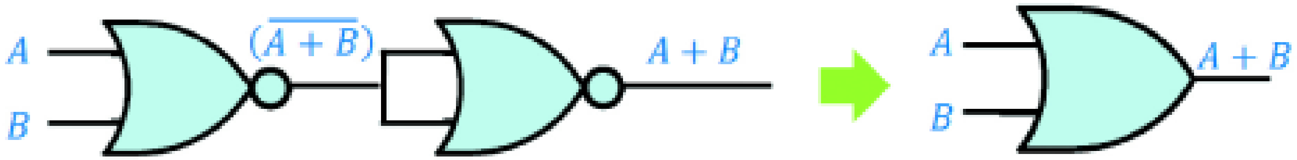


Fig 2.24: OR gate using NOR gate

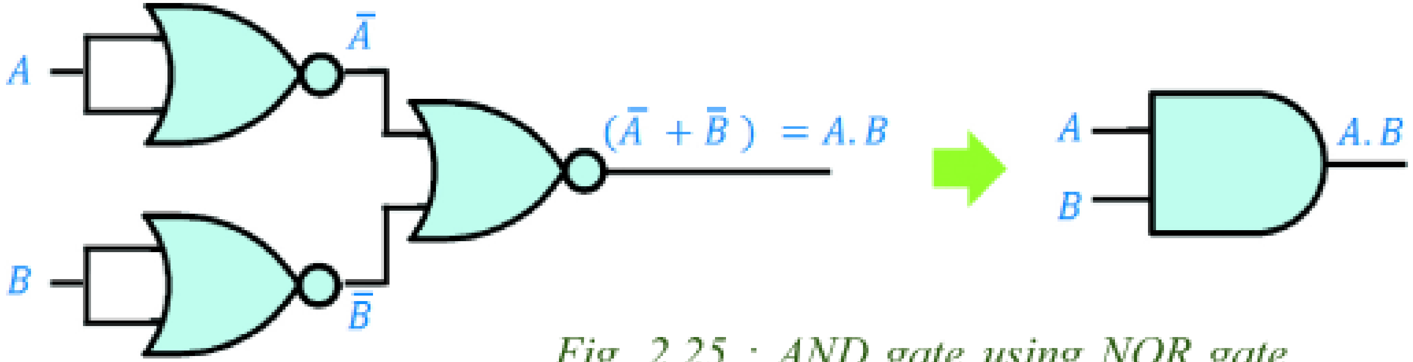


Fig. 2.25 : AND gate using NOR gate

Teaching materials Prepared by

RIDHUN DEV

Lecture IHRD College

+1 / +2 Computer Science

8089552581