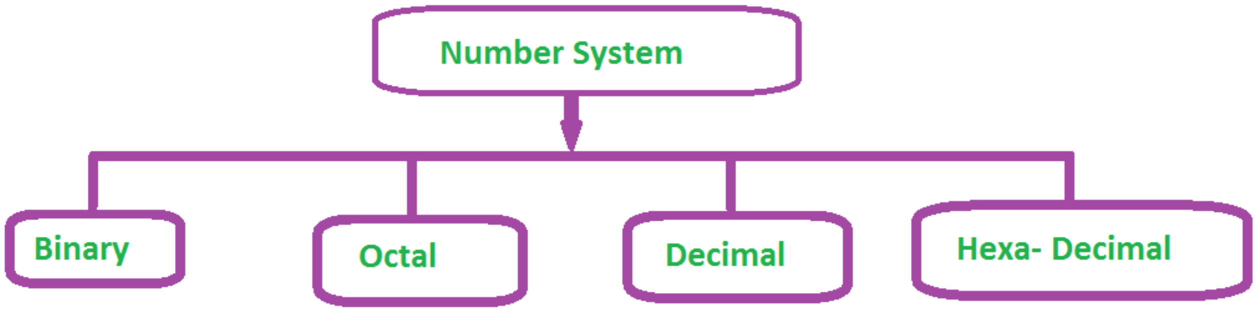
Data Representation

Introduction:

In digital systems like computers, the quantities represented by some symbols of digits. There are many number systems used in digital technology to represent the digits.

Types of Number System:

There are 4 basic number system to represent the digits. Binary, Octal, Decimal and Hexa-Decimal.



Binary Number:

- Binary Numbers contains only Two digits
- Combination of 0 and 1 digits make Binary numbers
- Base of Binary numbers is 2
- Ex. $(1010110)_2$ $(11011001)_2$

Octal Number:

- Octal Numbers contains Eight digits
- Combination of 0,1,2,3,4,5,6,7 digits make Octal numbers
- Base of Octal numbers is 8
- Ex. $(10637)_8$ $(11001)_8$

Decimal Number:

- Decimal Numbers contains Ten digits
- Combination of 0,1,2,3,4,5,6,7,8,9 digits make Decimal numbers
- Base of Decimal numbers is 10
- Ex. $(80695)_{10}$ $(110101)_{10}$ $(51063)_{10}$

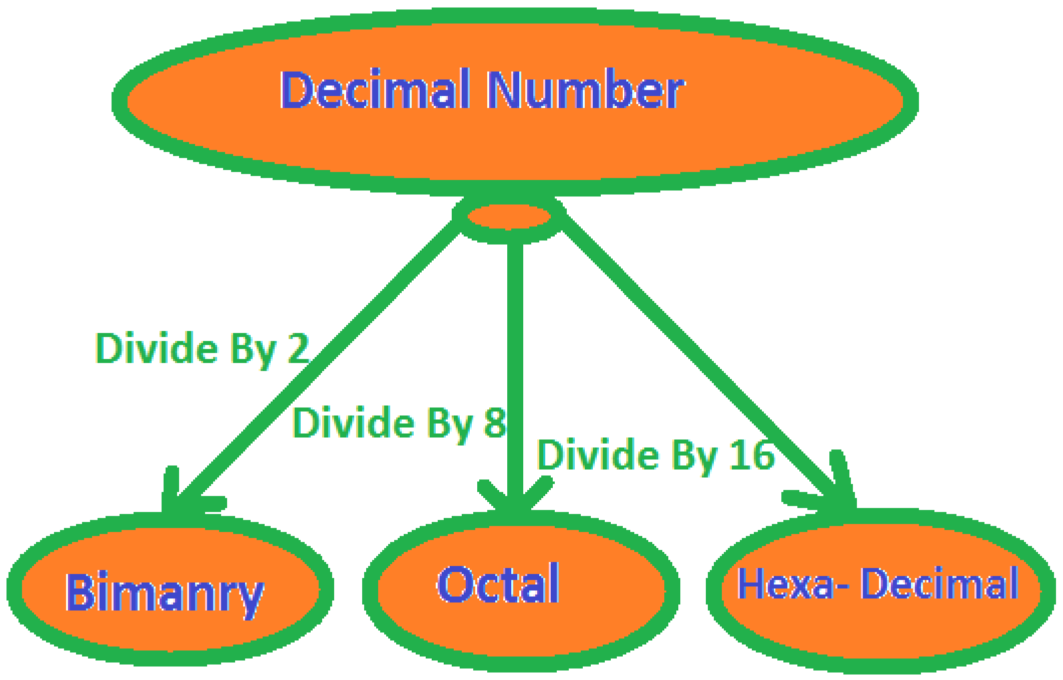
Hexa-Decimal Number:

- Hexa-Decimal Numbers contains Sixteen digits
- Combination of 0,1,2,3,4,5,6,7,8,9 and A,B,C,D,E,F digits make Hexa-Decimal numbers

  $$\left.\begin{array}{l}\text{Where } 0\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 9 \Rightarrow 10 \text{ Digits} \\ A=10,\ B=11,\ C=12,\ D=13,\ E=14,\ F=15 \Rightarrow 06 \text{ Digits}\end{array}\right\}\ \text{Total=16 Digits}$$
- Base of Hexa-Decimal numbers is 16
- Ex. $(10749)_{16}$ $(110101)_{16}$ $(51063)_{16}$ $(A5B21CF)_{16}$

Conversion of Number System

(By Using Repeated Division Method)



Decimal to Binary Conversion:

A decimal number can be convert in its equivalent Binary number by using following steps.

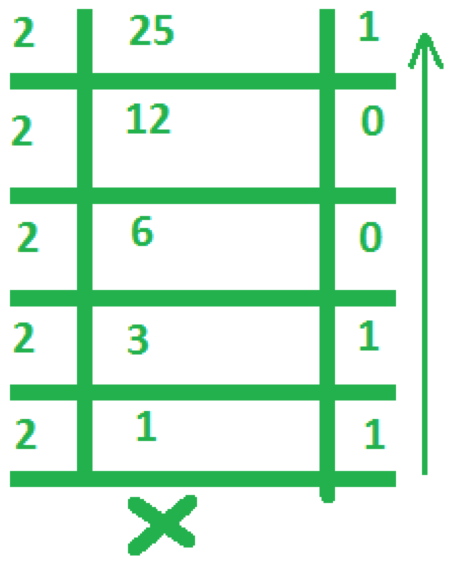
Step-1: Divide the decimal number by 2 and get quotient.

Step-2: Write remainder in front of division.

Step-3: Repeat step-1 and step-2 until you get quotient 0 (Zero).

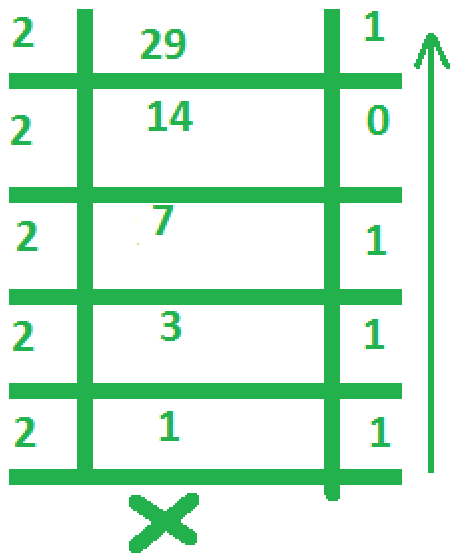
Step-4: Write the remainders in Last to First Order (MSB to LSB) and this number will be equivalent binary number of your decimal number.

Convert $(25)_{10} = (.......)_2$



$(25)_{10} = (11001)_2$

Convert $(29)_{10} = (......)_2$



$(29)_{10} = (11101)_2$

(MSB: Most Significant Bit, LSB: Least Significant Bit)

Convert the following numbers

1. $(51)_{10} = (..................)_2$
2. $(64)_{10} = (..................)_2$
3. $(15)_{10} = (..................)_2$

Decimal to Octal Conversion:

A decimal number can be convert in its equivalent Octal number by using following steps.

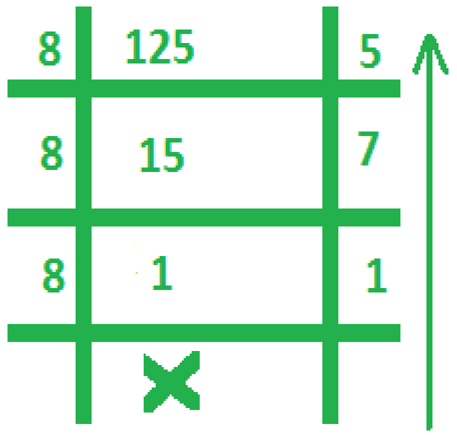
Step-1: Divide the decimal number by 8 and get quotient.

Step-2: Write remainder in front of division.

Step-3: Repeat step-1 and step-2 until you get quotient 0 (Zero).

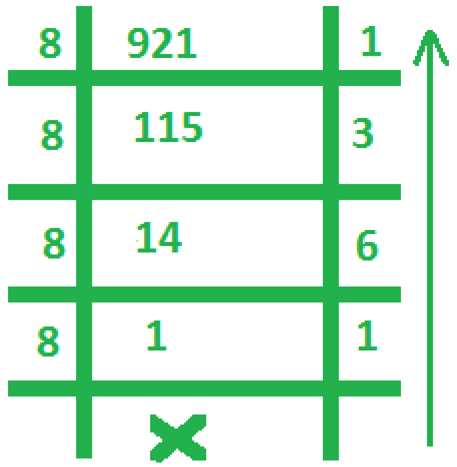
Step-4: Write the remainders in Last to First Order and this number will be equivalent octal number of your decimal number.

Convert $(125)_{10} = (.........)_8$



$(125)_{10} = (175)_8$

Convert $(921)_{10} = (.........)_8$



$(921)_{10} = (1631)_8$

Convert the following numbers

1. $(251)_{10} = (..................)_8$
2. $(694)_{10} = (..................)_8$
3. $(159)_{10} = (..................)_8$

Decimal to Hexa- Decimal Conversion:

A decimal number can be convert in its equivalent Hexa-Decimal number by using following steps.

Step-1: Divide the decimal number by 16 and get quotient.

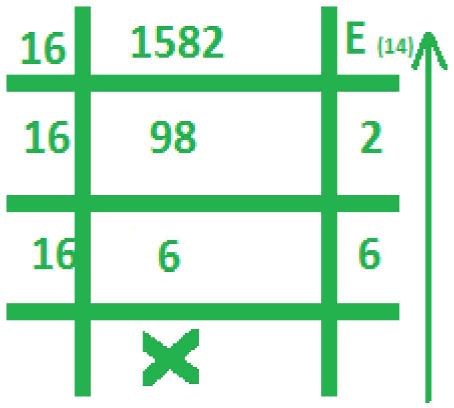
Step-2: Write remainder in front of division.

Step-3: Repeat step-1 and step-2 until you get quotient 0 (Zero).

Step-4: Write the remainders in Last to First Order and this number will be equivalent Hexa- Decimal number of your decimal number.

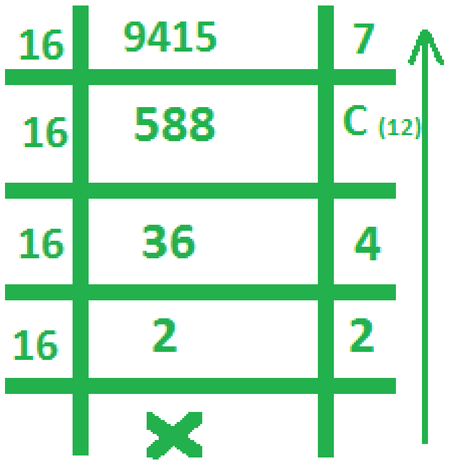
(Note: if Remainder is 10 then write A, 11 then write B and so on.)

Convert $(1582)_{10} = (.......)_{16}$



$(1582)_{10} = (62E)_{16}$

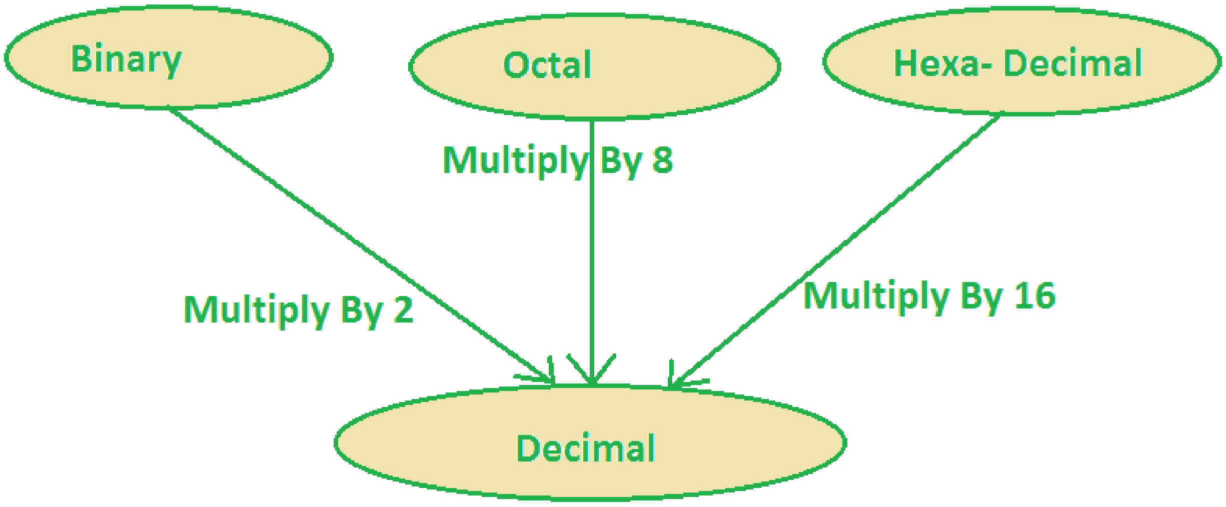
Convert $(9415)_{10} = (.......)_{16}$



$(9415)_{10} = (24C7)_{16}$

Convert the following numbers

1. $(6950)_{10} = (..................)_{16}$
2. $(6004)_{10} = (..................)_{16}$
3. $(1899)_{10} = (..................)_{16}$



Binary to Decimal Conversion

Example: Convert $(10011)_2 = (\ldots\ldots\ldots.)_{10}$

4	3	2	1	0	<-- index number (n)
1	0	0	1	1	<-- Digit of index

$\sum$ nth Digit $*2^n$

Where

$\sum$ Denotes the Sum,

n Denotes to index and

nth Digit denotes to Binary Digit at nth index

$(1x2^4) + (0x2^3) + (0x2^2) + (1x2^1) + (1x2^0)$

16 + 0 + 0 + 2 + 1

$(19)_{10}$

Convert $(11101)_2 = (\ldots\ldots\ldots.)_{10}$

4	3	2	1	0	< - - Index
1	1	1	0	1	

$(1x2^4) + (1x2^3) + (1x2^2) + (0x2^1) + (1x2^0)$

16 + 8 + 4 + 0 + 1

$(29)_{10}$

Octal to Decimal Conversion

Example: Convert $(12005)_8 = (\dots\dots\dots)_{10}$

FORMULA: $\sum {}_{n}{}^{th}$ Digit * 8^n

4 3 2 1 0 < - - Index

1	2	0	0	5

$(1x8^4) + (2x8^3) + (0x8^2) + (0x8^1) + (5x8^0)$

4096 + 1024 + 0 + 0 + 5

$(5125)_{10}$

Convert $(2170)_8 = (\dots\dots\dots)_{10}$

3 2 1 0 < - - Index

2	1	7	0

$(2x8^3) + (1x8^2) + (7x8^1) + (0x8^0)$

1024 + 64 + 56 + 0

$(1144)_{10}$

Hexa-Decimal to Decimal Conversion

Example: Convert $(1A0B)_{16} = (\dots\dots\dots)_{10}$

FORMULA: $\sum {}_{n}{}^{th}$ Digit * 16^n

3 2 1 0 < - - Index

1	A	0	D

$(1x16^3) + (Ax16^2) + (0x16^1) + (Dx16^0)$

4096 + (10*256) + 0 + (13*1) (Here A=10, D=13)

4096 + 2560 +0 + 13

$(6669)_{10}$

Convert $(FACE)_{16}$= $(……….)_{10}$

3	2	1	0	< - - Index
F	A	C	E	

$(F\text{x}16^3) + (A\text{x}16^2) + (C\text{x}16^1) + (E\text{x}16^0)$

$(15\text{x}16^3) + (10\text{x}16^2) + (12\text{x}16^1) + (14\text{x}16^0)$

$(15\text{x}4096) + (10\text{x}256) + (12\text{x}16) + (14\text{x}1)$

$(61440) + (2560) + (192) + (14)$

$(64206)_{10}$

Octal to Binary Conversion

Octal Number (0 to 7)	Equivalent Binary Number (421)
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Convert $(257)_8 = (\ldots\ldots\ldots)_2$

Find the equivalent binary number from table-1 for each given octal number.

Each octal number must to equal to 3 binary digits.

2->010 5->101 7->111

$(257)_8 = (010101111)_2$

Convert $(6050301)_8 = (\ldots\ldots\ldots)_2$

6-> 110 0-> 000 5-> 101 3-> 011 1-> 001

$(6050301)_8 = (110000101000011000001)_2$

Binary to Octal Conversion

Step-1: Make group of 3 binary digits from right hand side

Step-2: If last group have not sufficient digits for group then add 0 (ZEROs) before the incomplete group and make it 3 digit group.

Step-3: Find equivalent octal number for each group from table-1

Example: $(10101000011000001)_2 = (\ldots\ldots\ldots\ldots\ldots)_8$

$(10\ \underline{101}\ \underline{000}\ \underline{011}\ \underline{000}\ \underline{001})_2$

Here 10 is incomplete group so make it by adding 0 before given digits. Now it will be as shown below

$(\underline{010}\ \underline{101}\ \underline{000}\ \underline{011}\ \underline{000}\ \underline{001})_2$

$(\underline{2}\quad \underline{5}\quad \underline{0}\quad \underline{3}\quad \underline{0}\quad 1)_8$

Example: $(1010110110111)_2 = (\ldots\ldots\ldots\ldots\ldots)_8$

Example: $(\underline{001}\ \underline{010}\ \underline{110}\ \underline{110}\ \underline{111})_2 = (1\ 2\ 6\ 6\ 7)_8$

Hexa-Decimal to Binary Conversion

Hexa-Decimal Number (0 to 9 and A to F)	Equivalent Binary Number (8421)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A (10)	1010
B (11)	1011
C (12)	1100
D (13)	1101
E (14)	1110
F (15)	1111

Convert $(FACE)_{16}$= $(\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots)_2$

Find the equivalent binary number from table-2 for each given Hexa-Decimal number. Each Hexa-Decimal number must to equal to 4 binary digits.

F->1111 A->1010 C->1100 E->1110

$(FACE)_8$= $(1111101011001110)_2$

Convert $(A5B2C6D9)_{16}$= $(\ldots\ldots\ldots\ldots\ldots\ldots\ldots)_2$

$(A5B2C6D9)_{16}$= $(10100101101100101100011011011001)_2$

Binary to Hexa-Decimal Conversion

Step-1: Make group of 4 binary digits from right hand side

Step-2: If last group have not sufficient digits for group then add 0 (ZEROs) before the incomplete group and make it 4 digit group.

Step-3: Find equivalent Hexa-Decimal number for each group from table-2

Example: $(10101000011000001)_2$ = $(\ldots\ldots\ldots\ldots)_{16}$

$(1\ \underline{0101}\ \underline{0000}\ \underline{1100}\ \underline{0001})_2$

Here 1 is incomplete group so make it by adding 0 before given digits. Now it will be as shown below

$(\underline{0001}\ \underline{0101}\ \underline{0000}\ \underline{1100}\ \underline{0001})_2$

$(\underline{1}\quad \underline{5}\quad \underline{0}\quad \underline{C}\quad 1)_{16}$

Example: $(110100111011001101)_2$ = $(\ldots\ldots\ldots\ldots)_{16}$

$(\underline{0011}\ \underline{0100}\ \underline{1110}\ \underline{1100}\ \underline{1101})_2$ = $(2\ 8\ E\ C\ D)_{16}$

Example: $(\underline{0001}\ \underline{0001}\ \underline{0001}\ \underline{0001}\ \underline{0001})_2$ = $(11111)_{16}$

Floating Point Conversion

Binary to Decimal Floating Point Conversion

Floating Point numbers contains decimal point values. Integer numbers does not contain decimal point values. Binary to decimal, floating point conversion is similar to integer conversion. We use the same formula as given below

FORMULA: $\sum_{n}{}^{th}\ \text{Digit} * 2^{n}$

2	1	0		-1	-2	< - - Index
1	0	1	.	1	1	

$$(1x2^{2}) + (0x2^{1}) + (1x2^{0}) + (1x2^{-1}) + (1x2^{-2})$$

$$(1x4) + (0x2) + (1x1) + \left(\frac{1}{2^{1}}\right) + \left(\frac{1}{2^{2}}\right)$$

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

$$\frac{5}{1} + \frac{1}{2} + \frac{1}{4}$$

$$\frac{20+2+1}{4} \Rightarrow \frac{23}{4} \Rightarrow 5.75$$

$$(5.75)_{10}$$

Decimal to Binary Floating Point Conversion

When the floating point decimal to Binary conversion take place then the Integral part of decimal number convert as repetitive division method. But the Fractional part of decimal number convert with different method.

Example: Convert $(105.75)_{10} = (.........)_2$

$(105.75)_{10} = (.........)_2$

2	105	1	LSB
2	52	0	↑
2	26	0	↑
2	13	1	↑
2	6	0	↑
2	3	1	↑
2	1	1	MSB
	×		

Fractional Value	Multiply by 2	Integral Value
.75	.75 x 2 = 1.50	1 ↓
.50	.50 x 2 = 1.0	1 ↓

$(105)_{10} = (1101001)_2$
$(.75)_{10} = (11)$

Final Result
$(105.75)_{10} = (1101001.11)_2$

Example: Convert $(93.175)_{10} = (.........)_2$

2	93	1	LSB
2	46	0	↑
2	23	1	↑
2	11	1	↑
2	5	1	↑
2	2	0	↑
2	1	1	MSB
2	x		

Fractional Part	Multiply by 2	Integral Part
0.175	.175 x 2 = 0.350	0 ↓
0.35	.350 x 2 = 0.700	0 ↓
0.7	.7 x 2 = 1.4	1 ↓
0.4	.4 x 2 = 0.8	0 ↓
0.8	.8 x 2 = 1.6	1 ↓
0.6	.6 x 2 = 1.2	1

$(93)_{10} = (1011101)_2$
$(.175)_{10} = (001011)_2$

$(93.175)_{10} = (1011101.001011)_2$

Character Representation

1. ASCII Code:

ASCII stands for American Standard Code for Information Interchange and used to describe each character in Specific code. ASCII can describe all the characters of all languages.

Some of ASCII Values for some of characters are as given below

Character	ASCII Value	Binary Value
A	65	1000001
B	66	1000010
Z	90	1011010
a	97	1100001
b	98	1100010
z	122	1111010
0 (Zero)	48	110000
1	49	110001
2	50	110010
9	57	111001
क	2325	100100010101

In Python ord(character Value) will return the equivalent ASCII Value of given Character value.

Ex. ord('65') => output: A

ord('क') => output: 2325

In Python chr(ASCII Value) will return the equivalent Character value of given ASCII.

Ex. chr(42) => Output: *

chr(38) => Output: &

chr(2326) => Output: ख

2. ISCII Code:

The ISCII stands for Indian Standard Code for Information Interchange. This code is used to represent the Indian scripts in computer code. Therefore ISCII also known as Indian Script Code for Information Interchange. The standard code can represent the scripts like Devanagari script, Gujarati, Oriya, Bengali, Assamese, Telugu, Kannada, Malayalam and Tamil Script. It can also represent the Sanskrit Script.

3. Unicode:

The main objective of Unicode is to provide the worldwide one encoding translation scheme for all language. The Unicode is developed as a universal character set. The aim of Unicode are...

A. To define the all characters used in all languages which can write on computers.

B. To support of all other character sets that have been encoded.

There are multiple Unicode Translation Schemes which used to represent characters. Following are some of Unicode Translation formats

(I) UTF-8 (Unicode Transformation Format)

It is 8-bit code unit format that is also called Octet. It is type of multi byte encoding. So it is variable width encoding format to represent every character in Unicode. Variable width mean sometime 16 bit, 32 bit or more bit can use to store the character.

(II) UTF-32

It is a fixed length encoding scheme that uses exactly 4 byte (32 bit) space to represent all characters.

Binary Arithmetic

The ALU of computer system performs all the arithmetical operations inside it. But ALU does not recognize the decimal numbers so it convert the decimal numbers into binary numbers and after it ALU works on binary numbers for all arithmetic calculations. The following are the binary arithmetic operations.

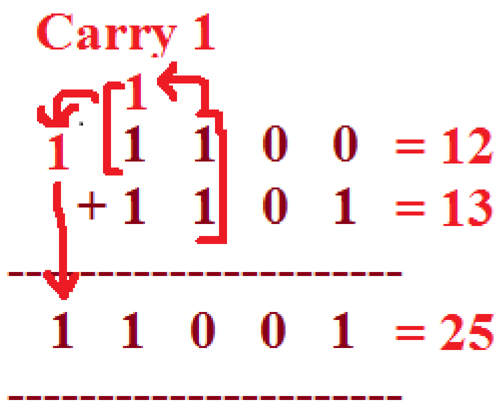
1. **Binary Addition**
2. **Binary Subtraction**
3. **Binary Multiplication**
4. **Binary Division**

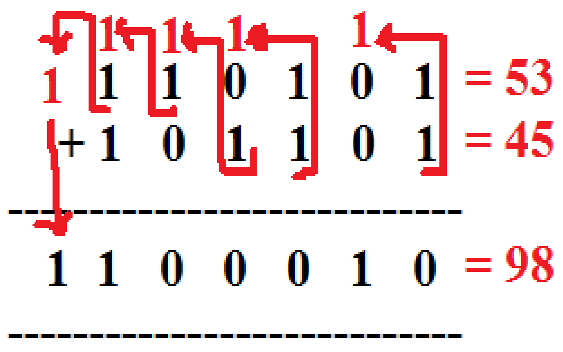
Binary Addition:

Basic rules for Binary Addition are:

(i) 0 + 0 = 0 (ii) 0 + 1 = 1 (iii) 1 + 0 = 1

(iv) 1 + 1 = 0 with carry 1





Boolean Logic Gate

A logic gate is a device that acts as a building block for digital circuits. Logic gates can be used in technologies such as Laptops, smartphones, tablets. Logic gates make decisions based on a combination of digital signals coming from its inputs.

Truth Table:

A truth table is a mathematical table used in logic gates, Boolean algebra and Boolean functions. It is used to prove the truthiness of any Boolean equation / expression / function.

Types of Logic Gates:

There are Three Basic (Fundamental) gates in Boolean Algebra

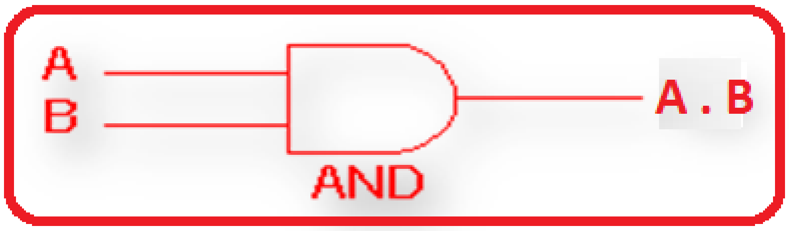
AND, OR, NOT,

Other Gates are…

NAND, NOR, XOR, and XNOR

AND GATE

- The AND gate is an electronic circuit that gives a True output (1) only if all its inputs are True (1).
- AND gate takes two or more input signals and produce only one output signal
- It represent with dot (.) in expression.

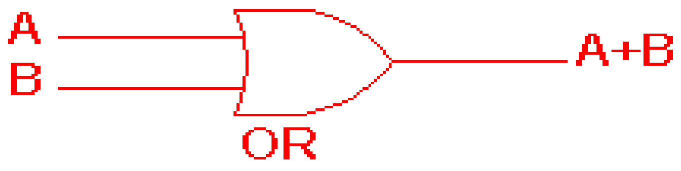


Truth Table of AND Gate:

Input A	Input B	Output A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR GATE

- The OR gate is an electronic circuit that gives a True output (1) if one or more of its inputs are True.
- OR gate also takes two or more input signals and produce only one output signal.

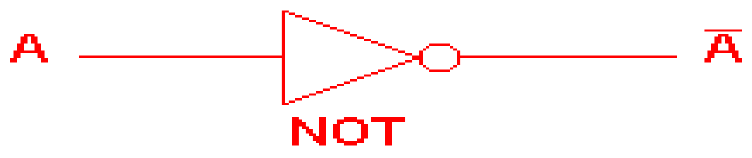


Truth Table of OR Gate:

Input A	Input B	Output A+B
0	0	1
0	1	1
1	0	1
1	1	0

NOT GATE

- The NOT gate is an electronic circuit that gives a True output (1) if its input is False.
- NOT gate takes only one input signal and produce only one output signal.
- The output of NOT gate is complement of its input.
- It is also called inverter.



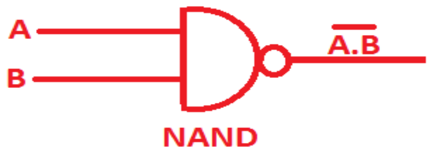
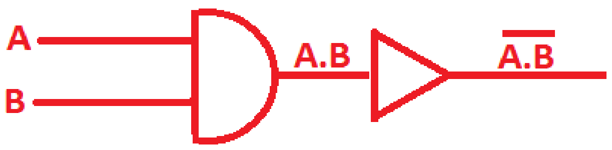
Truth Table of NOT Gate:

Input A	Output $\overline{A}$
0	1
0	0

NAND GATE, NOR GATE XOR AND XNOR GATE

NAND GATE

- It is known as a “Universal” Gate because ANY digital circuit can be implemented with NAND gates alone.
- It takes at least two inputs and produce one Boolean output (True / False)
- It is combination AND gate followed by a NOT Gate.

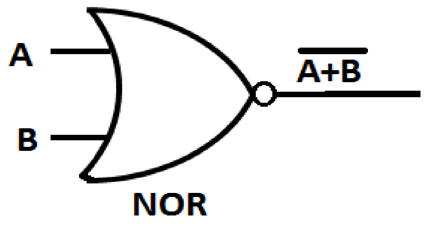
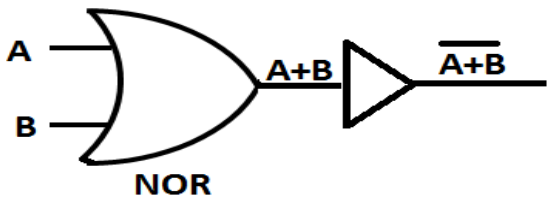
-  

Truth Table of NAND Gate:

Input A	Input B	A.B	Output $\overline{A.B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

NOR GATE

- It is known as a "Universal" Gate because ANY digital circuit can be implemented with NOR gates alone.
- It takes at least two inputs and produce one Boolean output (True / False)
- It is combination OR gate followed by a NOT Gate.

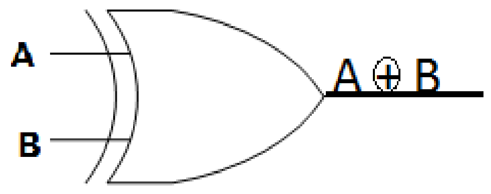
 

Truth Table of NOR Gate:

Input A	Input B	A + B	Output $\overline{A+B}$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Exclusive-OR Gate

- It takes at least two inputs and produce one Boolean output (True / False)
- It produce output as True, if all inputs are of different nature (Some are True and some are False)
- The output is "False" if both inputs are "False" or if both inputs are "True."

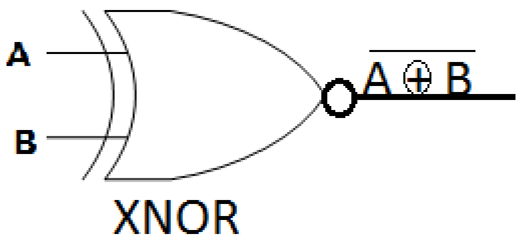


XOR

Input A	Input B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive-NOR Gate

- It takes at least two inputs and produce one Boolean output (True / False)
- It produce output as False, if all inputs are of different nature (Some are True and some are False)
- The output is "True" if both inputs are "False" or if both inputs are "True."
- It is combination XOR gate followed by a NOT Gate.



Input A	Input B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

Boolean Theorems

SN	THEOREM	BOOLEAN PROPERTY
1	A + 0 = A	COMMON PROPERTIES
2	A + 1 = 1	
3	A . 0 = 0	
4	A . 1 = A	
5	A + A = A	
6	A + A′ = 1	
7	A . A = A	
8	A . A′ = 0	
9	(A′)′ = A	
10	A + B = B + A	Cumulative
11	A + (B+C) = (A+B)+C	Associative
12	A.(B+C) = (A.B)+(A.C) A+(B.C) = (A+B).(A+C)	Distributive
13	A+B′+1=1 is True for A.B′.0=0	Principle of Duality

De- Morgan's Theorem

Pre-Requisite Knowledge

Product of terms	:	(A.B.C)
Sum of terms	:	(A+B+C)
Complement of term	:	(A.B.C)′ or (A+B+C)′
Sum of Product (SOP)	:	(A.B)+(A.C)+(B.C)
Product of Sum (POS)	:	(A+B).(A+C).(B+C)
Complement of Product	:	$\overline{(A.B).(A.C).(B.C)}$
Complement of Sum	:	$\overline{(A.B)+(A.C)+(B.C)}$

De-Morgan's Theorem is very powerful theorem that extremely useful to simplify the complex Boolean expression where product or sum of variables is inverted.

There are two theorems of De-Morgan.

1. The Complement of Sum is equal to Product of Complement or Vice Versa.

$$\overline{(A + B + C)} = \overline{A} \,.\, \overline{B} \,.\, \overline{C}$$

2. The Complement of Product is equal to Sum of Complement or Vice Versa.

$$\overline{(A \,.\, B \,.\, C)} = \overline{A} + \overline{B} + \overline{C}$$

Prove the De-Morgan Theorem:

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

A	B	C	A+B+C	$\overline{(A+B+C)}$	A′	B′	C′	A′.B′.C′
0	0	0	0	1	1	1	1	1
0	0	1	1	0	1	1	0	0
0	1	0	1	0	1	0	1	0
0	1	1	1	0	1	0	0	0
1	0	0	1	0	0	1	1	0
1	0	1	1	0	0	1	0	0
1	1	0	1	0	0	0	1	0
1	1	1	1	0	0	0	0	0

Finish*