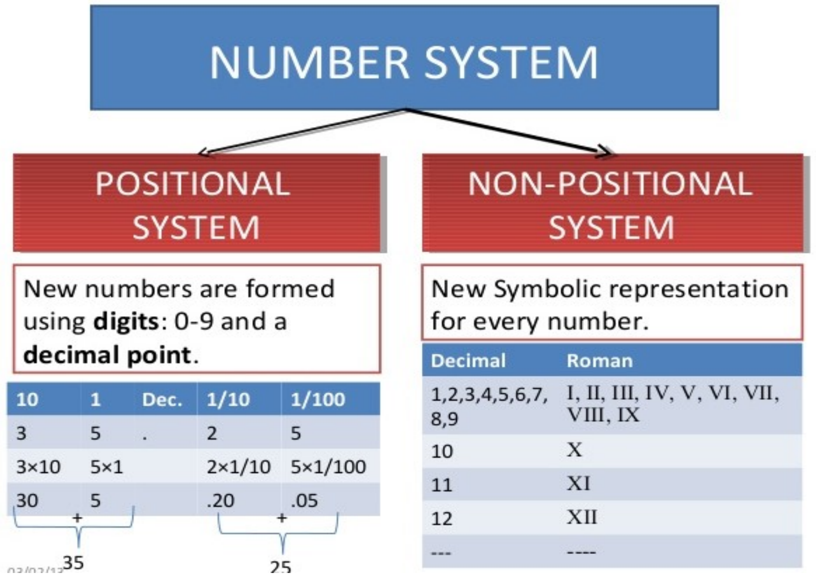
NUMBER SYSTEMS

There are several number systems which we normally use, such as decimal, binary, octal, hexadecimal, etc. The Decimal Number System: The Decimal number system is a number system of base or radix equal to 10, called Arabic numerals. The symbols used to represent number : 0, 1, 2, 3,4,5,6,7,8, 9 which are used for counting.
This is expressed mathematically of the first five positions as 10^4 10^3 10^2 10^1 10^0

For example the value of the combination of symbols 435 is determined by adding the weight of each position as $4 * 10^2 + 3 * 10^1 + 5 * 10^o$which can be written as
$4 * 100 + 3 * 10 + 5 * 1$ or $400 + 30 + 5 = 435$

What is positional and non-positional number system?



NUMBER SYSTEM

POSITIONAL SYSTEM

New numbers are formed using **digits**: 0-9 and a **decimal point**.

10	1	Dec.	1/10	1/100
3	5	.	2	5
3×10	5×1		2×1/10	5×1/100
30	5		.20	.05

$\underbrace{30 + 5}_{35}$ $\underbrace{.20 + .05}_{25}$

NON-POSITIONAL SYSTEM

New Symbolic representation for every number.

Decimal	Roman
1,2,3,4,5,6,7, 8,9	I, II, III, IV, V, VI, VII, VIII, IX
10	X
11	XI
12	XII
---	----

Different Types of Conversions

- Decimal System

Binary to Decimal Decimal to Binary

- Octal System

Octal to decimal Decimal to Octal
Octal to Binary Binary to Octal

- Hexa-Decimal System

Hexa-Decimal to Decimal
Decimal to Hexa-Decimal
Hexa-Decimal toBinary
Binary to Hexa-Decimal

……

Binary to Decimal

Binary weights for whole numbers

Position	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Weight	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Value	**1024**	**512**	**256**	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**

Binary weights for Fractional numbers

Position	**-1**	**-2**	**-3**	**-4**	**-5**	**-6**
Weight	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
Value	**0.5**	**0.25**	**0.125**	**0.0625**	**0.03125**	**0.015625**

Example Problems

1. Convert binary number 111001_2 to decimal.
   $111001_2 = 1\times2^5+1\times2^4+1\times2^3+0\times2^2+0\times2^1+1\times2^0 = \mathbf{(57)_{10}}$

2. Convert binary number 1110.011_2 to decimal.
   1110.001_2
   $= 1\times2^3+1\times2^2+1\times2^1+0\times2^0+0\times2^{-1} +1\times2^{-2} +1\times2^{-3} = \mathbf{(15.375)_{10}}$

Solve the following (Binary to Decimal)

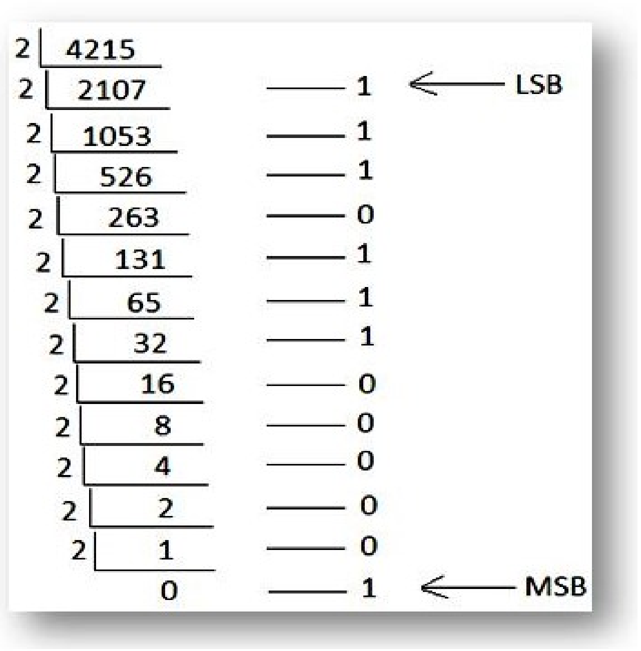
1. $1101101_2 = 109_{10}$
2. $0.011_2 = 0.375_{10}$
3. $1111.1011_2 = 15.6875_{10}$

Decimal to Binary

Example Problems

1. Convert 4215_{10} to its binary equivalent

Solution:



Therefore, $4215_{10} = 1000001110111_2$

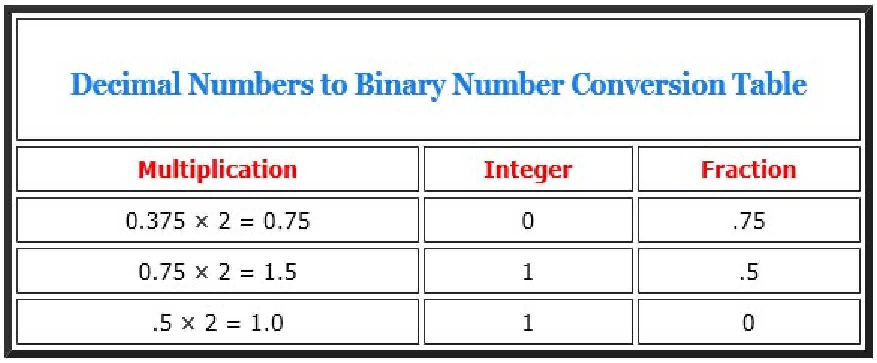
Solve the following (Decimal to Binary)

1. $41_{10} = 101001_2$
2. $0.6_{10} = 0.10011_2$
3. $4096.3125_{10} =$
   1000000000000.0101_2

2. Convert the following decimal numbers to their binary equivalents:

(a) 0.375

Solution:



Decimal Numbers to Binary Number Conversion Table

Multiplication	Integer	Fraction
0.375 × 2 = 0.75	0	.75
0.75 × 2 = 1.5	1	.5
.5 × 2 = 1.0	1	0

Therefore, $0.375_{10} = 0.011_2$

(b) 0.435

Solution:

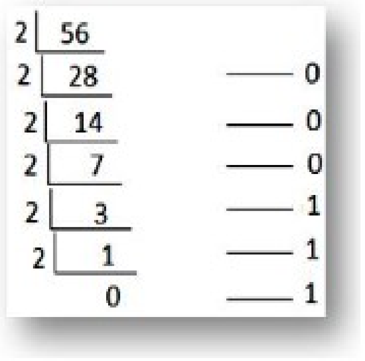
Decimal Numbers to Binary Number Conversion Table		
Multiplication	**Integer**	**Fraction**
0.435 × 2 = 0.87	0	.87
0.87 × 2 = 1.74	1	.74
.74 × 2 = 1.48	1	.48
.48 × 2 = 0.96	0	.96
.96 × 2 = 1.92	1	.92

Therefore, 0.435_{10} = $(0.01101\ldots)_2$

3. Convert $(56.75)_{10}$ to its binary equivalent.

Solution:

At first we find the binary equivalent of 56.



Therefore, 56_{10} = 111000_2

The binary equivalent of 0.75 is obtained below:

Decimal Numbers to Binary Number Conversion Table		
Multiplication	**Integer**	**Fraction**
0.75 × 2 = 1.5	1	.5
0.5 × 2 = 1.0	1	0

Therefore, 0.75_{10} = 0.11_{10}

Hence 56.75_{10} = 111000.11_{10}

Octal to Decimal

Example Problems

Convert the following octal numbers to their decimal, equivalent.

(a) 35_8 (b) 100_8 (c) 0.24_8

$35_8 = (3 \times 8^1) + (5 \times 8^o)$
$= 24 + 5 = 29$

$100_8 = (1 \times 8^2) + (0 \times 8^1) + (0 \times 8^o)$
$= 64 + 0 + 0 = 64_{10}$

$0.24_8 = (2 \times 8^{-1}) + (4 \times 8^{-2})$
$= \frac{2}{8} + \frac{4}{64}$
$= 0.3125_{10}$

Solve the following (Octal to Decimal)

1. $630.4_8 = 408.5_{10}$
2. $5473_8 = 2875_{10}$
3. $763.375_8 = 499.49395_{10}$

Convert Decimal to Octal

Example Problems

Convert the following decimal numbers into Octal Equivalent

a) 245 **b) 175**

Solution:

8	245		8	175
8	30 – 5 (LSD)		8	21 – 7
8	3 – 6		8	2 – 5
	0 – 6 (MSD)			0 – 2

Therefore, $245_{10} = 365_8$, therefore , $175_{10} = 257_8$

Octal to Binary

Convert the following octal numbers to their binary equivalent.

(a) 247_8 (b) 501_8

2	4	7	Octal
010	100	111	binary

$247_8 = 010100111_2$

5	0	1	octal
101	000	001	binary

$501_8 = 101000001_2$

Binary to Octal

Convert 11010101 . 01101 to an octal-number.

011 010 101 . 011 10

$\underline{011}$ $\underline{010}$ $\underline{101}$ $\underline{011}$ $\underline{010}$

3 2 5 3 2

$11010101.0110_1 = 325.32_8$

Binary Addition

Add (a) 111 and 101 (b) 1010, 1001 and 1101.

$$\begin{array}{r} \scriptstyle(1)(1) \\ 1\,1\,1 \\ 1\,0\,1 \\ \hline 1\,1\,0\,0 \end{array} \qquad \text{(b)} \quad \begin{array}{r} \scriptstyle(2)(1)(1)(1) \\ 1\;0\;1\;0 \\ 1\;0\;0\;1 \\ 1\;1\;0\;1 \\ \hline 1\,0\,0\,0\,0 \end{array}$$

Perform the following subtraction 101 – 111.

Subtract the smaller number from the larger.

$$\begin{array}{r} 111 \\ -\,101 \\ \hline 010 \end{array}$$

Thus 1 0 1 – 1 1 1 = - 010 = - 10

Perform the following subtractions.

(a) 11 - 01 , (b) 11-10 (c) 100 - 011

$$\text{(a)} \quad \begin{array}{r} 11 \\ -\,01 \\ \hline 10 \end{array} \qquad \text{(b)} \quad \begin{array}{r} 11 \\ -\,10 \\ \hline 01 \end{array} \quad \text{(c)} \quad \begin{array}{r} 100 \\ -\,011 \\ \hline 001 \end{array}$$

Multiply the following binary numbers.

(a) **101 × 1 1** **(b)** **1101 × 10**

(c) **1010 × 1 0 1** **(d)** **1 0 1 1 × 1 0 1 0**

Solution:

$$\text{(a)} \quad \begin{array}{r} 101 \\ \underline{11} \times \\ 101 \\ \underline{101} \\ 1111 \end{array} \qquad \text{(b)} \quad \begin{array}{r} 11101 \\ \underline{1\,0} \times \\ 0000 \\ \underline{1101} \\ 11010 \end{array}$$

$$\text{(c)} \quad \begin{array}{r} 1010 \\ \underline{101} \times \\ 1010 \\ 0000 \\ \underline{1010} \\ \underline{110010} \end{array} \qquad \text{(d)} \quad \begin{array}{r} 1011 \\ \underline{1010} \times \\ 0000 \\ 1011\times \\ 0000\times \\ \underline{1011\times} \\ 1101110 \end{array}$$

COMBINATIONAL CIRCUITS

1. What is a combinational circuit?
   - The output of combinational circuit at any instant of time depends only on the levels present at input terminals.
   - The combinational circuit does not use any memory. The previous state of input does not have any effect on the present state of the circuit.
   - A combinational circuit can have an n number of inputs and m number of outputs.

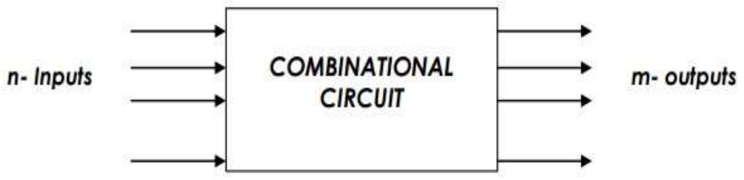


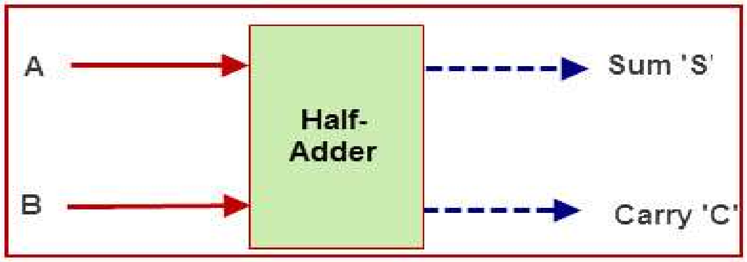
Fig: Generalized Combinational Circuit

…..………

HALF ADDER

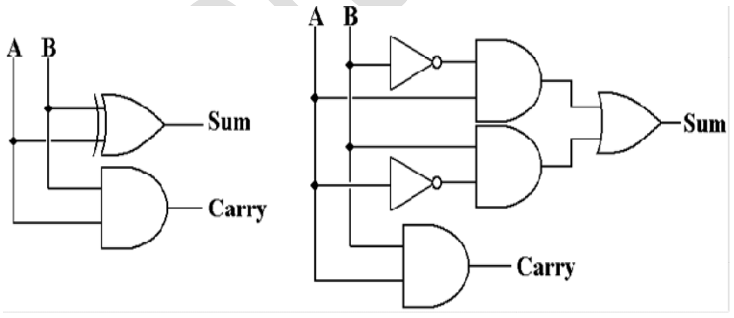
A half adder takes two single bit binary numbers and produces a sum and a carry–out, called “carry”.

Logic Symbol Truth Table



A in	B in	S out	C out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Logic Diagram



Logical Expression

The sum can be given in two equivalent expressions. The simplest expression uses the exclusive OR function

Sum = $A \oplus B$

Carry is the logical AND of the two inputs:

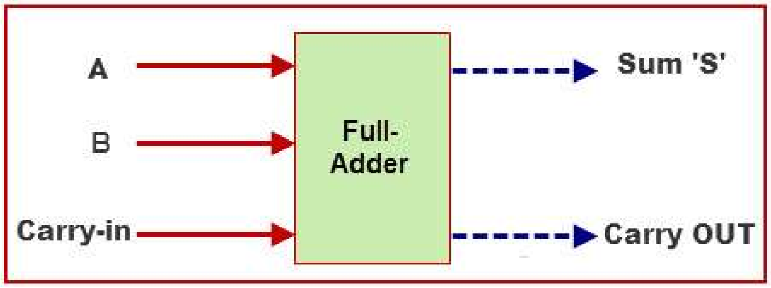
Carry = $A \cdot B$

Half adder is a combinational logic circuit with two inputs and two outputs. The half adder circuit is designed to add two single bit binary numbers A and B. It is the basic building block for addition of two **single** bit numbers. This circuit has two outputs **carry** and **sum**.

FULL ADDER

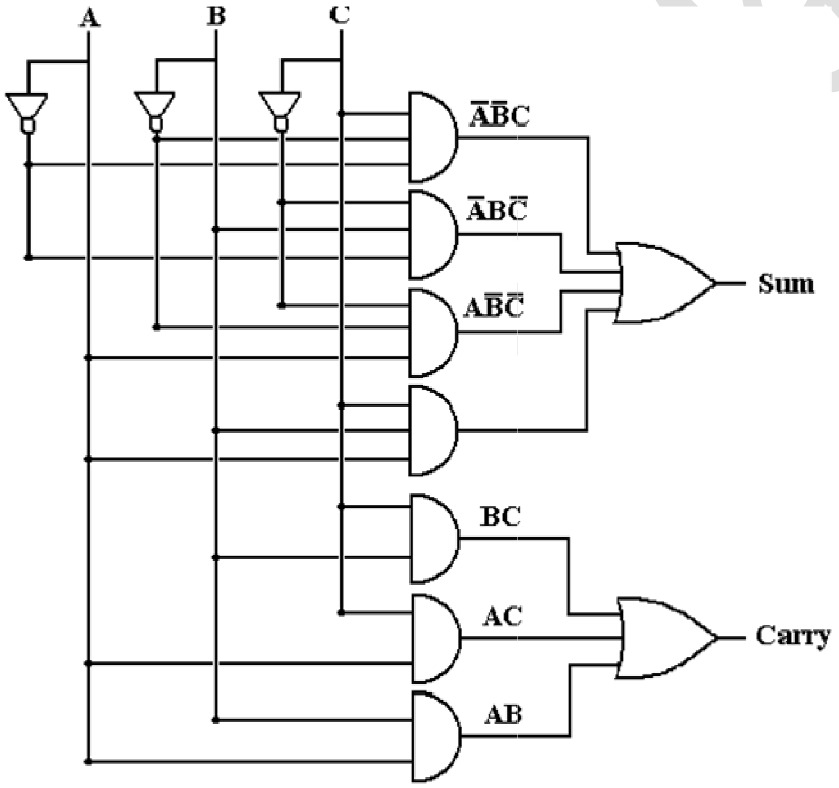
Full adder is developed to overcome the drawback of Half Adder circuit. The full adder is a three input and two output combinational circuit. The first two inputs are A and B and the third input is an input carry as C-in and produces a sum 'S' and a carry–out.

Logic SymbolTruth Table



A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Logical Diagram



Logical Expression

The circuit implements the following two expressions, where C is the carry–in to the full adder.

Sum= $\bar{A}\cdot\bar{B}\cdot C + \bar{A}\cdot B\cdot\bar{C} + A\cdot\bar{B}\cdot\bar{C} + A\cdot B\cdot C$

Carry = $A\cdot B + A\cdot C + B\cdot C$

Suppose we let the carry–in C = 0. Then $\bar{C}$ = 1.

What we have then is as follows.

Sum $= \bar{A}\cdot\bar{B}\cdot 0 + \bar{A}\cdot B\cdot 1 + A\cdot\bar{B}\cdot 1 + A\cdot B\cdot 0$

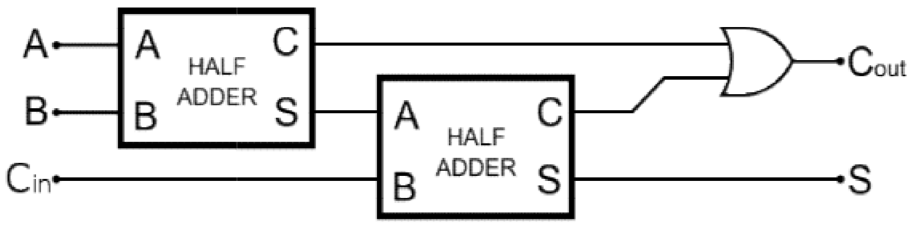
$= \bar{A}\cdot B + A\cdot\bar{B}$

Carry $= A\cdot B + A\cdot 0 + B\cdot 0$

$= A\cdot B$

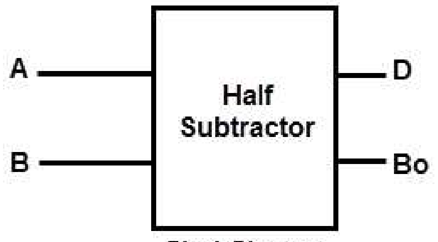
As expected, a full adder with carry–in set to zero acts likes a half adder.

A full adder can be constructed from two half adders by connecting A and B to the input of one half adder, connecting the sum from that to an input to the second adder, connecting the carry in, C_{in}, to the other input and ORing the two half adder carry outputs to give the final carry output, C_{out}.



HALF SUBTRACTOR

A Half Subtractor is a logic circuit that performs Binary subtraction. It accepts two input variables and produces two output variables called the difference bit and borrow bit.
The binary subtraction is also performed by the Ex-OR gate with additional circuitry to perform the borrow operation. Thus, a half subtractor is designed by an Ex-OR gate including AND gate with A input complemented before fed to the gate.



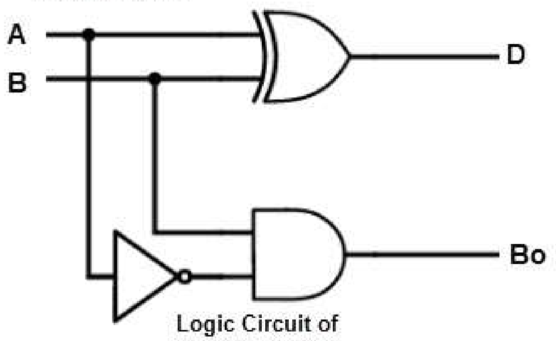
Block Diagram

A	B	D	B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Truth Table

Logical Expression
DIFFERENCE bit:
D =A⊕B

BORROW bit
$\mathbf{B = \overline{X}.Y}$

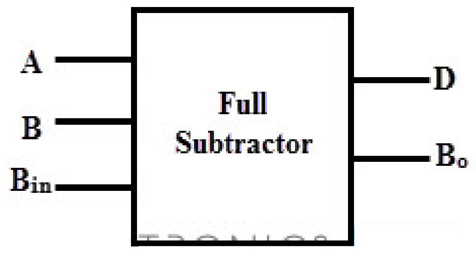


Logic Circuit of Half Subtractor

…..………

FULL SUBTRACTOR

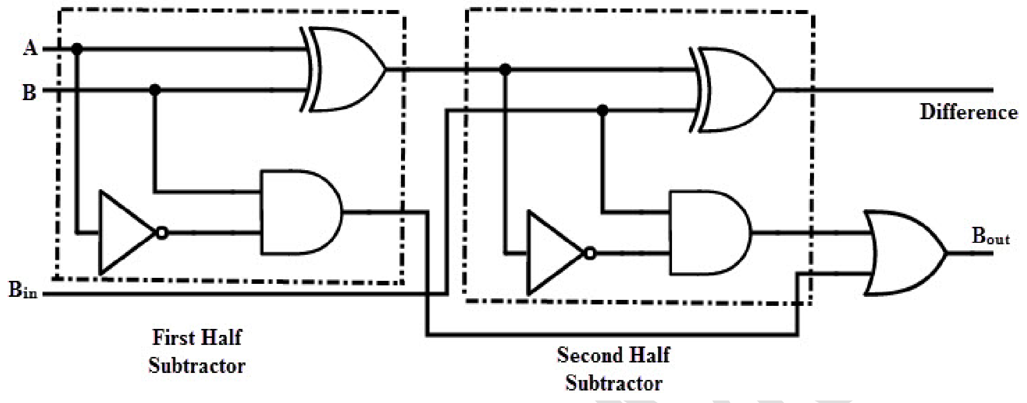
The Half Subtractor does not handle Borrow from the previous stage. The Full Subtractor accepts Borrow from the previous stage, so it has three inputs Bin, A and B. It has two outputs D (Difference) and Bo (Borrow Output).



Block Diagram of Full Subtractor

A	B	B_{in}	D	B_o
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Truth Table



$$D = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$$

$$= B_{in}(\bar{A}\bar{B} + AB) + \bar{B}_{in}(\bar{A}B + A\bar{B})$$

$$= B_{in}(A \odot B) + \bar{B}_{in}(A \oplus B)$$

$$= B_{in}(\overline{A \oplus B}) + \bar{B}_{in}(A \oplus B)$$

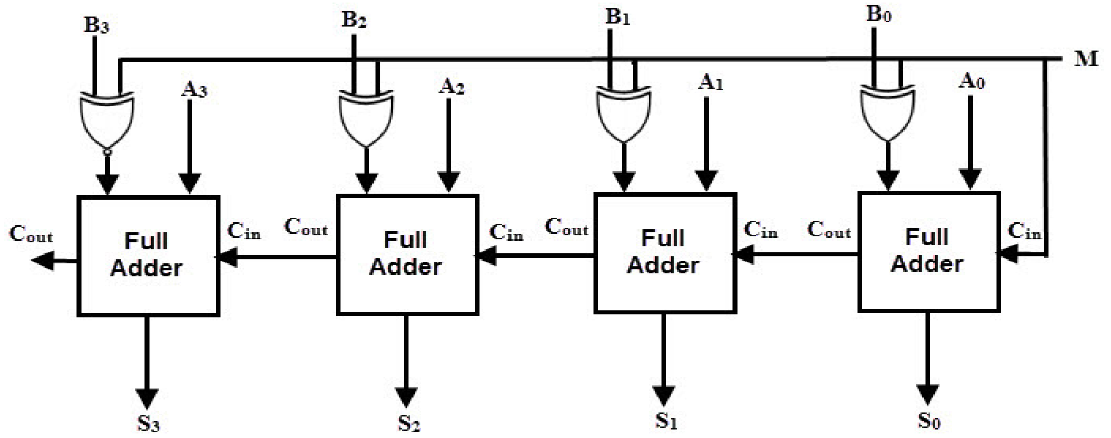
$$= B_{in} \oplus (A \oplus B)$$

………

Working of Adder/Subtractor with a neat diagram

- It is a logic circuit that can be used to perform both addition and subtraction. The circuit is laid from right to left.
- The circuit can be designed by adding an Ex-OR gate with each full adder as shown in below figure.
- The adder/subtractor which has two 4 bit inputs as A3, A2, A1, A0 and B3, B2, B1, B0.

- The mode input control line M is connected with carry input of the least significant bit of the full adder. This control line decides the type of operation, whether addition or subtract



When M= 1, the circuit is a subtractor and when M=0, the circuit becomes adder.
The Ex-OR gate consists of two inputs to which one is connected to the B and other to input M.
When M = 0, B Ex-OR of 0 produce B. Then full adders add the B with A with carry input zero and hence an addition operation is performed.
When M = 1, B Ex-OR of 0 produce B complement and also carry input is 1. Hence the complemented B inputs are added to A and 1 is added through the input carry, nothing but a 2's complement operation. Therefore, the subtraction operation is performed.

INTRODUCTION TO COMPUTER ARCHITECTURE

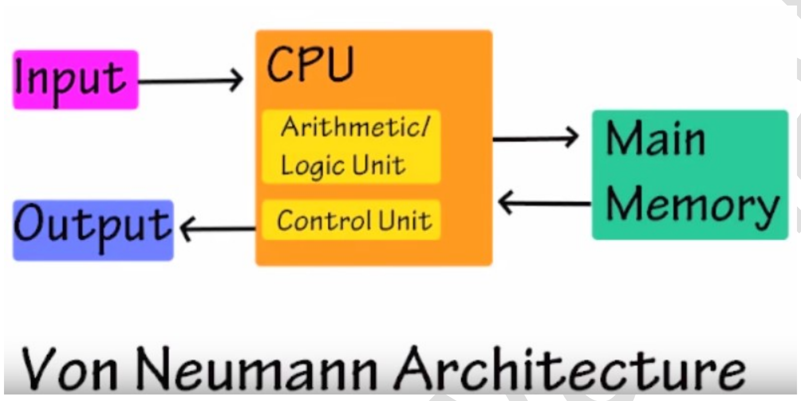
What is Computer Architecture?
Computer Architecture or digital computer organization describes the design and operational structure of a computer system.
It can also be defined as the art and science of selecting and interconnecting hardware components to create computers that meet functional, performance and cost goals.

There are two types of Computer Architecture

1. Von Neumann Architecture
2. Harvard Architecture

Von Neumann Architecture

A mathematician called John Von Neumann described the basic arrangement (or architecture) of a computer. Most computers today follow the concept that he described although there are other types of architecture.



Arithmetic logic unit - This is the part of the CPU performs arithmetic and logic operations. The ALU has 3 sections, the Register, the ALU circuitry and the pathways in between.
Register is basically a storage cell which holds information such as the address of the instructions and results of the calculations.
ALU circuitry It actually performs calculations and is designed from AND, OR and NOT gates just as a chip.
Pathways in between are for electric current within ALU.

Main Memory (RAM)–Main memory is volatile which means the information will be lost without constant flow of electricity; hence it is called a temporary storage device. Main memory can be seen as a sequence of cells. Each cell has its own unique address so that the data can be fetched.
Input/ Output -Computers can only interact with the world using input and output devices. Inputs receive data for the computer and outputs send information from the computer.
Control Unit - The control unit is in charge of 'fetching' each instruction that needs to be executed in a program by issuing control signals to the hardware. It then decodes the instruction and finally issues more control signals to the hardware to actually execute it.

Harvard Architecture

Harvard architecture is computer architecture with physically separate storage and signal pathways for instructions and data. The term originated from the Harvard Mark I relay-based computer, which stored instructions on punched tape (24 bits wide) and data in electro-mechanical counters (23 digits wide). These early machines had limited data storage, entirely contained within the CPU, and provided no access to the instruction storage as data, making loading and modifying programs tedious.

Comparison between Von Neumann and Harvard Architecture

Von Neumann Architecture	Harvard Architecture
Programs and data are stored in the same memory and managed by same information handling system.	Programs and data are stored and handled in different systems.
Increased efficiency in designing and operating one memory system.	Inefficient systems, as data and program handling tasks are different.
Can be slow because data and programs are stored in same memory.	Can be much faster because data and programs are stored in different memory.
Desktop personal Computer	Microcontroller based systems.

LOGIC GATES

1. What are Logic gates?
Logic gates are electronic circuits that perform logical operations. Logic gates are circuit made of Transistor, Diode and Resistor.A logic gate can have more than one input but only one output.

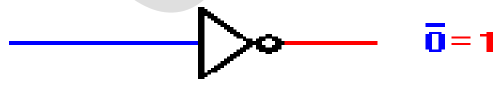
2. Derive basic/simple logic gates with its definition, logic symbol, truth table and expression?
The three basic logic operations are **NOT**, **AND**and**OR**

NOT Gate (Inverter)

Definition: NOT gate is a logic gate that changes a 0 input to 1 and a 1 input to 0. NOT gate also called the Inverter performs the operation called Inversion or Complement or Negation.

Truth Table

Logic Symbol:



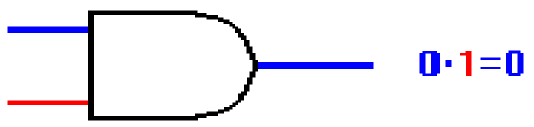
Y	$\bar{A}$
0	1
1	0

Expression: **Y= $\bar{A}$**

Application:
NOT gate can be used to find 1's complement of an 8-bit binary number (byte of data)

AND Gate (Multiplication)

Definition: AND gate is a logic gate which performs logical multiplication. The output is HIGH (1) only if both inputs are HIGH (1). Otherwise, output is LOW (0).



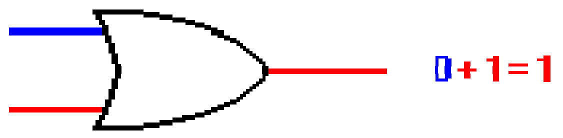
A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

Expression: **Y= A.B**

Application: AND gate is commonly used to enable Passage of signal (pulse waveform) from one point toanother.

OR Gate (Addition)

Definition: OR gate is a logic gate which performs logical addition. The output is HIGH (1) if any one of its inputs is HIGH (1). The output will be low only if both the inputs are low.



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

Expression: **Y= A+B**

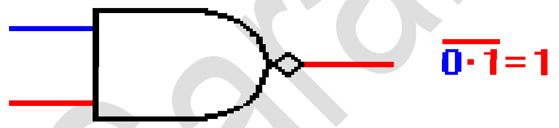
Application: OR gate is used in intrusion detection and alarm system.

3. Derive compound logic gates with its definition, logic symbol, truth table and expression?

NAND and NOR are compound logic gates.

NAND Gate

Definition: NAND is a combination of both AND and NOT gate. It operates the same as an AND gate but the output will be opposite. The output is HIGH (1) only if both inputs are LOW (0).

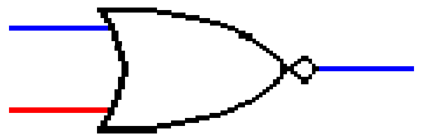


A	B	**Y=$\overline{A.B}$**
0	0	1
0	1	1
1	0	1
1	1	0

Expression: **Y= $\overline{A.B}$**

NOR Gate

Definition: NOR is a combination of both OR and NOT gate. It operates the same as an OR gate but the output will be opposite. The output is HIGH (1) if all its inputs are LOW (0), the output is LOW (0) if either of the inputs or all its inputs are HIGH (1).



$\overline{0+1}=0$

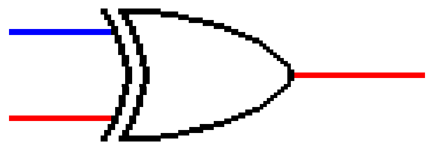
A	B	$Y=\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Expression: $Y=\overline{A+B}$

4. Explain XOR and XNOR logic gates with its definition, logic symbol, truth table and expression?

XOR Gate

Definition: The exclusive OR gate is a modified OR gate. The XOR gate produces a high outputwhen both its inputs are *different*. If the inputs are the same, the output is a low.



$0\oplus1=1$

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

Expression:

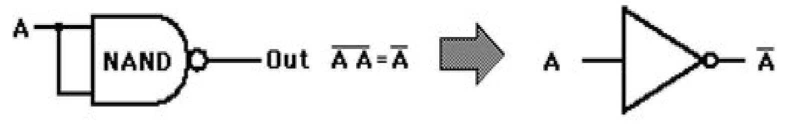
$Y=A.\overline{B}+\overline{A}.B$

Universal property of NAND and NOR Gates

Universal property of NAND Gate

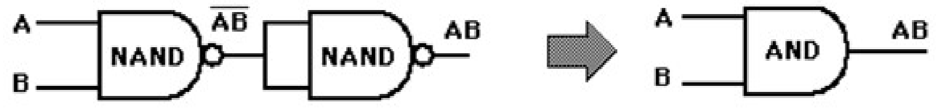
NAND as NOT Gate

All NAND input pins connect to the input signal A which gives an output $\overline{A}$



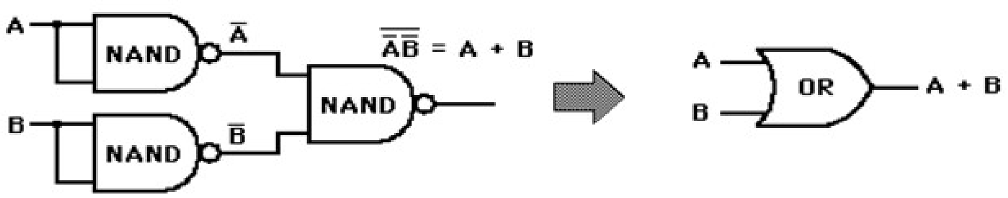
NAND as AND Gate

The AND is replaced by a NAND gate with its output complemented by a NAND gate inverter



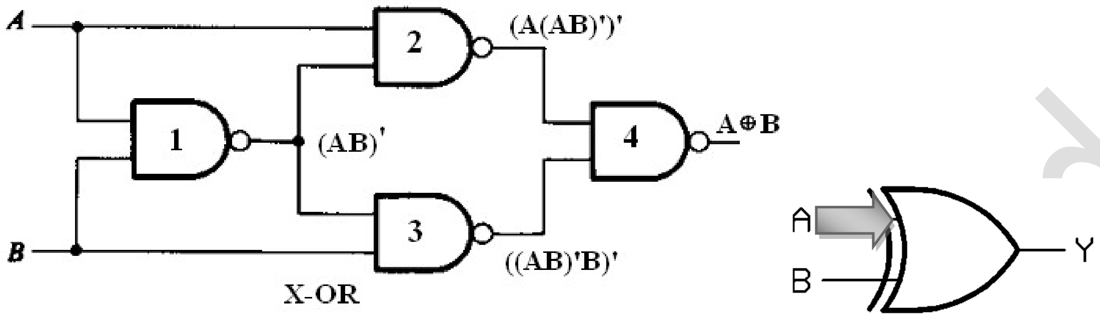
NAND as OR Gate

The OR gate is replaced by a NAND gate with all its inputs complemented by NAND gate inverter



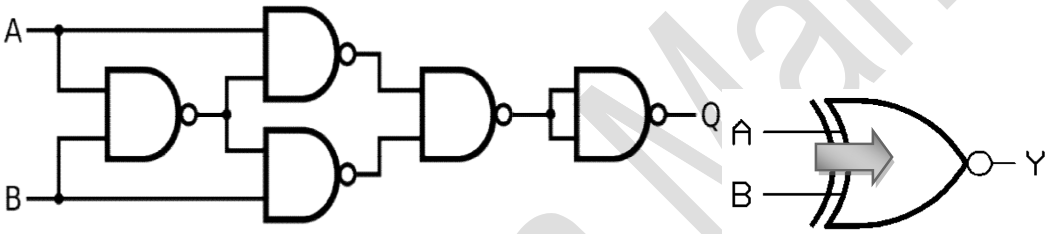
NAND as X-OR Gate

The X-OR gate is replaced by a NAND gate with all its inputs complemented by NAND gate inverter



NAND as X-NOR Gate

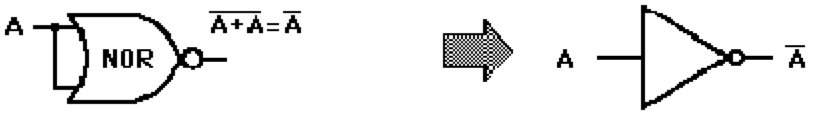
The X-NOR gate is replaced by a NAND gate with all its inputs complemented by NAND gate inverter



Universal property of NOR Gate

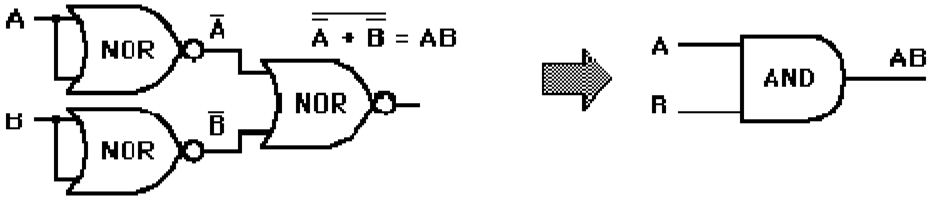
NOR as NOT Gate

A NOT gate is equivalent to an inverted-input NOR gate.



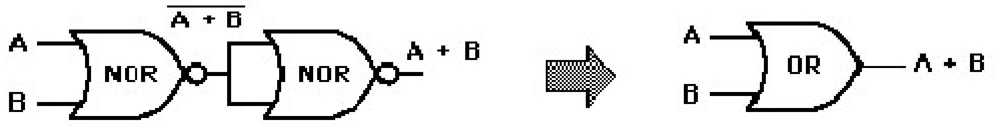
NOR as AND Gate

The AND is replaced by a NOR gate with its output complemented by a NOR gate inverter



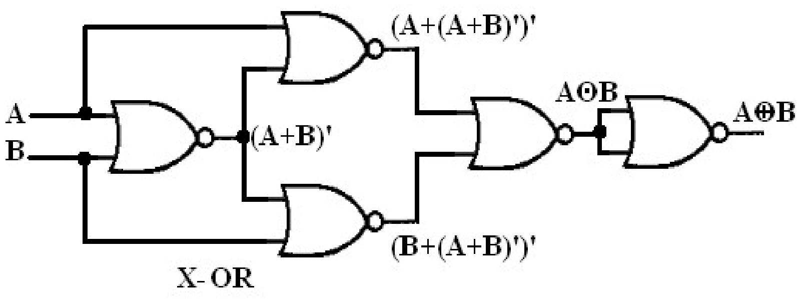
NOR as OR Gate

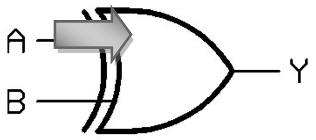
The OR is replaced by a NOR gate with its output complemented by a NOR gate inverter



NOR as X-OR Gate

The X-OR gate is replaced by a NOR gate with all its inputs complemented by NOR gate inverter





NOR as X-NOR Gate

The X-NOR gate is replaced by a NOR gate with all its inputs complemented by NOR gate inverter