UNIT-I CSO MCA – 1 Sem Introduction 8 hours Introduction:

Digital Computers

and Number System,

Logic Gates,

Boolean Algebra,

Map Simplification up to five variables,

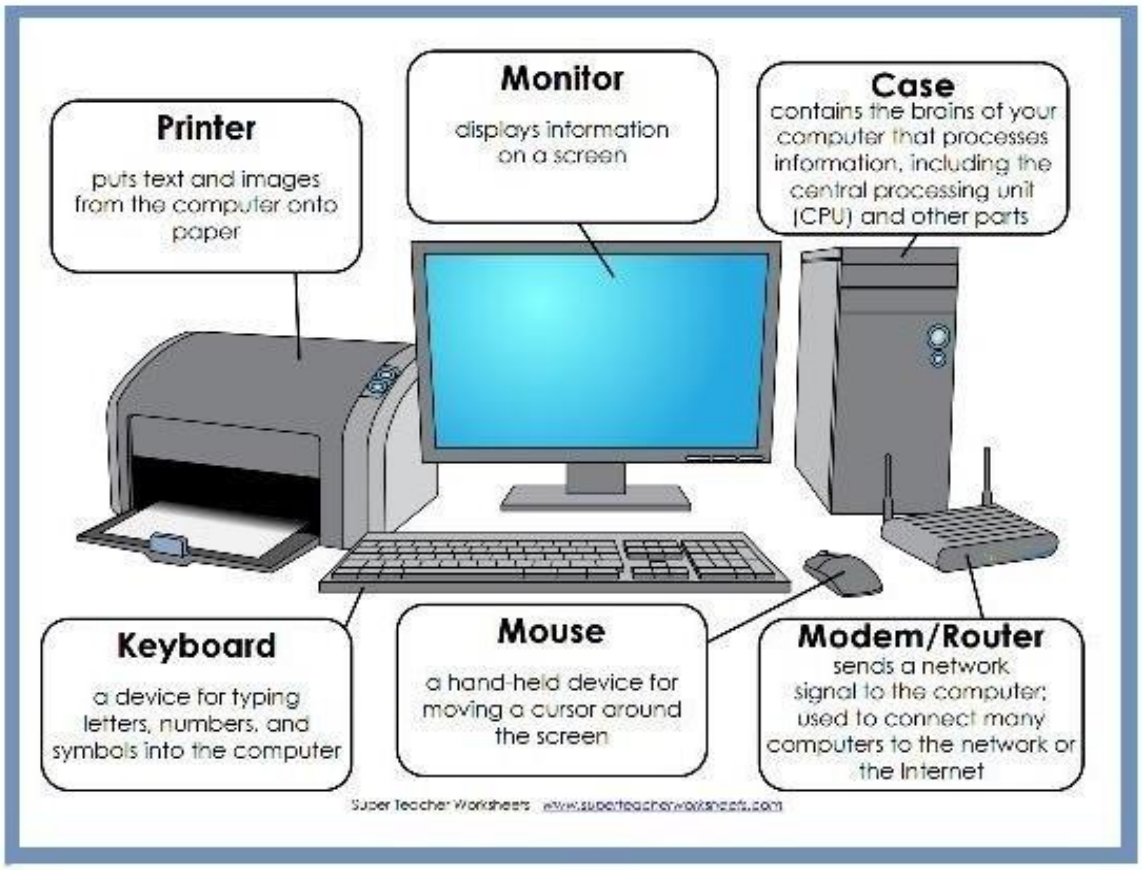
Combinational Circuits, Sequential Circuits, Look ahead carry adders, Data types, Complements, Fixed point representation, Fixed Point Addition & Subtraction, floating point Representation, Booth's Multiplication, IEEE754 Floating point standards.

Computer is an electronic device or machine that uses performs some, calculations and operations Like Arithmetic Operations instructions to store, retrieve, and process data by using **hardware and software**. and then produce outputs

It uses computer programming to perform arithmetical or logical operations sequences.

The **Full Form of Computer** is as follows: -

- **C** – Common
- **O** – Operated
- **M** -Machine
- **P** – Particularly
- **U** – Used for
- **T** – Technical &
- **E** – Educational
- **R** – Research



Full Forms of Computer Brands

- HCL – Hindustan Computer Limited
- HP – Hewlett Packard
- HTC – High Tech Computer
- IBM – International Business Machines
- MS – Microsoft
- Intel – Integrated Electronics
- Dell – Digital Electronic Link Library
- Apple – Asian Passenger Payload Experiment
- Lenovo – Le (Legend) & novo (New)
- Acer – Agency for the Cooperation of Energy Regulators.

Full Forms of Computer language

- **ALGOL** – **ALGO**rithmic **L**anguage
- **ASP** – **A**ctive **S**erver **P**ages

- **ADO** – **A**ctive **D**ata **O**bject
- **AJAX** – **A**synchronous **J**avaScript **A**nd **X**ML
- **BCPL** – **B**asic **C**ombined **P**rogramming **L**anguage
- **BASIC** – **B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode
- **COBOL** – **Common** Business-Oriented Language
- **CFML** – **C**old **F**usion **M**arkup **L**anguage
- CSS – Cascading Style Sheet
- FORTRAN – FORmula TRANslation
- HTML – Hyper Text Markup Language
- iMAC – iNTERNET MACintosh
- JS – Java Script
- JSP – Java Server Pages
- J2EE – Java 2 Platform Enterprise Edition
- LISP – LISt Processing
- MS WINDOWS – Microsoft **W**ide **I**nteractive **N**etworking **D**evel opment for **O**ffice **W**ork **S**olutions
- **ORACLE** – **O**ak **R**idge **A**nalytical and **C**ommon **L**ogical **E**ngine
- PHP – Hypertext Pre Processor
- **PROLOG** – **PRO**gramming in **LOG**ic
- **PERL** – **P**ractical **E**xtraction and **R**eporting **L**anguage
- SQL – Structured Query Language
- SAP – System Analysis and Programming
- **VFP** – **V**isual **F**ox**P**ro
- **VBS** – **V**isual **B**asic **S**cripting
- XSL – e**X**tensible Stylesheet Language
- **XML** – e**X**tensible **M**arkup **L**anguage

Full forms of Computer Memory

- KB- Kilobyte (this is the smallest storage unit)
- MB- MegaByte
- GB- GigaByte
- TB- TeraByte
- PB- PentaByte

Full Forms of Computer Hardware

- BIOS- Basic Input Output System
- CPU– Central Processing Unit
- CD- Compact Disk

- DVD- Digital Video Disk
- FDD- Floppy Disk Drive
- HDMI- High Definition Multimedia Interface
- HDD- Hard Disk Drive
- LCD – Liquid Crystal Display
- LED- Light Emitting Diode
- MMC- Multi-Media Card
- NTFS- New Technology File System
- PDF- Portable Document Format
- RAM– Random Access Memory
- Prom- Programmable Read-Only Memory
- ROM- Read-only Memory
- SMPS- Switch Mode Power Supply
- SSD- Solid State Drive
- UPS- Uninterrupted Power Supply
- USB- Universal Serial Bus
- VDU- Visual Display Unit
- VGA- Video Graphics Array
- Computer Softwares
- ALU- Arithmetic Logic Unit
- DVI- Digital Visual Interface
- OS- Operating System
- VIRUS – Vital Information Resources Under Seige

Full forms of Computer Networks

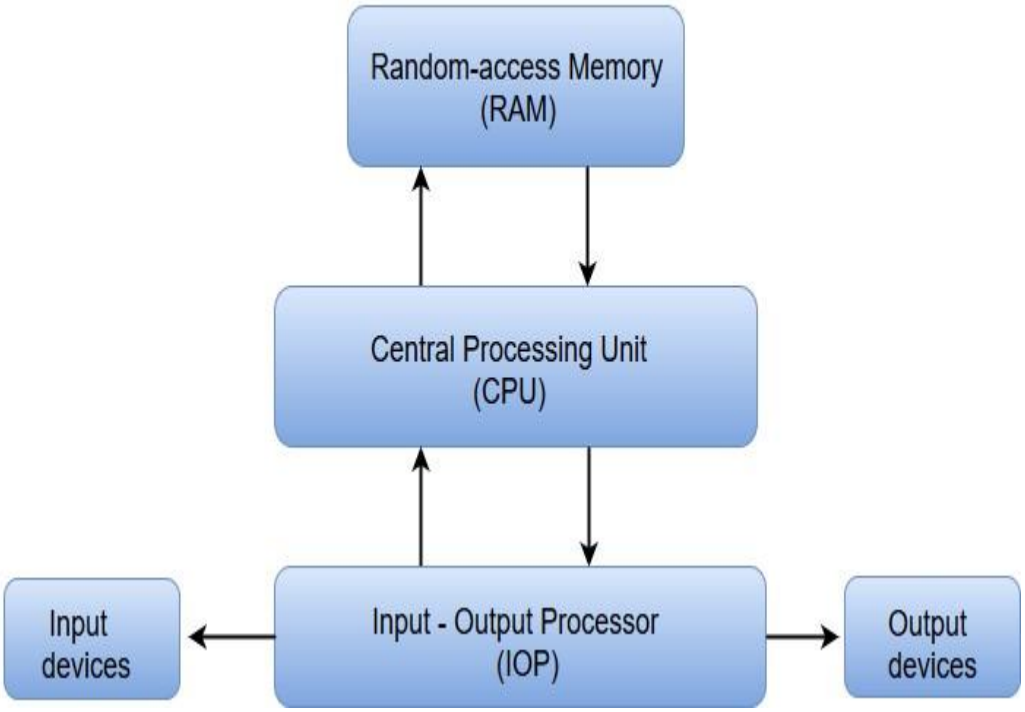
- CDMA full form- Code Division Multiple Access
- GSM- Global System for Mobile Communication
- GPRS- General Packet Radio Service
- SIM- Subscriber Identity Module
- 2G- 2nd Generation
- 3G- 3rd Generation
- 4G- 4th Generation
- 5G- 5th Generation
- DNS- Domain Name System
- HTML- HyperText Markup language
- IP- Internet Protocol
- ISP- Internet Service Provider
- URL- Uniform Resource Locator
- VPS- Virtual Private Server
- WAN- Wide Area Network

- WI-FI- Wireless Fidelity
- WLAN- Wireless Local Area Network
- WWW- World Wide Web

Digital Computers

A Digital computer can be considered as a digital system that performs various computational tasks.
The first electronic digital computer was developed in the late 1940s and was used primarily for numerical computations.
By convention, the digital computers use the binary number system, which has two digits: 0 and 1. A binary digit is called a bit.
A computer system is subdivided into two functional entities: Hardware and Software.
The hardware consists of all the electronic components and electromechanical devices that comprise the physical entity of the device.
The software of the computer consists of the instructions and data that the computer manipulates to perform various data-processing tasks.

Block diagram of a digital computer:



- The Central Processing Unit (CPU) contains an arithmetic and logic unit for manipulating data, a number of registers for storing data, and a control circuit for fetching and executing instructions.
- The memory unit of a digital computer contains storage for instructions and data.
- The Random Access Memory (RAM) for real-time processing of the data.
- The Input-Output devices for generating inputs from the user and displaying the final results to the user.
- The Input-Output devices connected to the computer include the keyboard, mouse, terminals, magnetic disk drives, and other communication devices

Real Life examples of digital computer

Some of the basic real-life examples of digital devices are as follows: -

- Personal computers/ Laptops/Notebooks
- Smartphones/tablets
- Digital Weighing Machine
- Check-in Kishok at Airport
- Automated Teller Machine – ATM

Components of Digital Computer

A digital computer has the following basic components –

- Input device
- CPU
- Output device

Input Devices

The Input device is basically the devices that are attached to the system such as a **mouse**, **keyboard**, and **scanner**.

Central Processing Unit (CPU)

CPU is the Central Processing Unit which is known as the brain of the computer as it controls the entire computer system.

- **Arithmetic Logic Unit** (ALU) – The main function of ALU is that it performs all the arithmetic and mathematical calculations which include addition, subtraction, multiplication, and division.
- **Control Unit** – The task of the Control unit is that it mainly allows the data to move from and to the CPU and manages the operations performed by ALU. All the instructions that are sent are picked, decoded, and analyzed. It then sends the instruction to input/output devices accordingly.
- **Memory** – This part is mainly used to store the data and is named "Internal memory".

Number systems

Number systems provide a way of conveying and quantifying information. The study of number systems is one of the important topics in digital electronics so as to understand how information is represented. Many of the applications of digital electronics like computers do not process the information as either alphabets or decimal numbers but are designed using circuits that process binary digits- only 0 and 1.

Number System

In a digital system, the system can understand only the optional number system. In these systems, digits symbols are used to represent different values, depending on the index from which it settled in the number system.

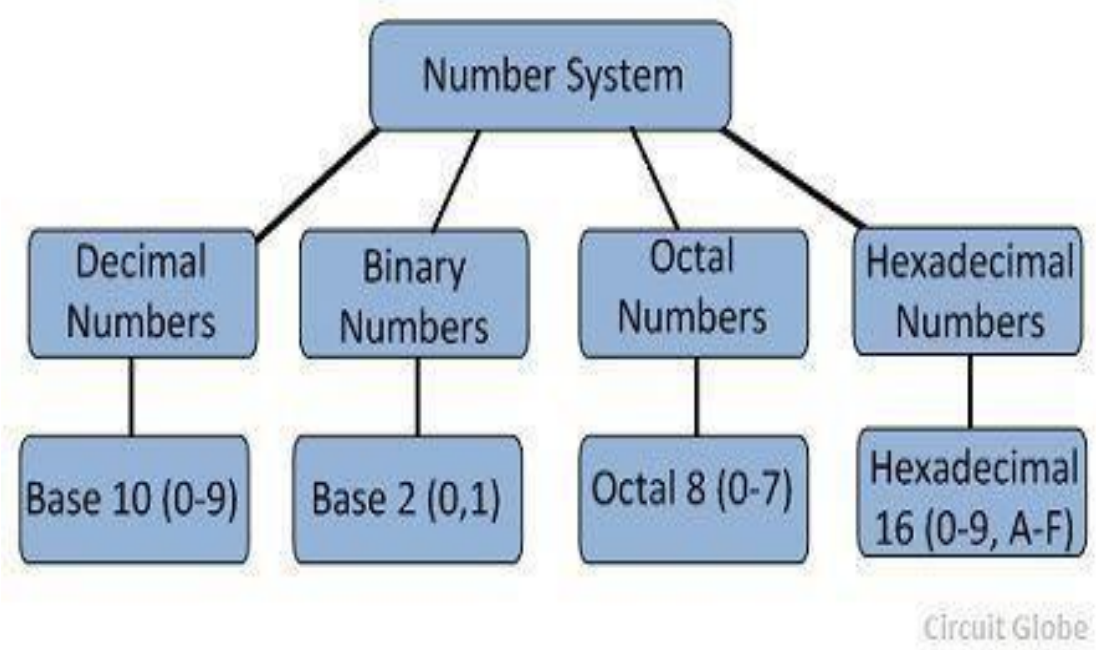
In simple terms, for representing the information, we use the number system in the digital system.

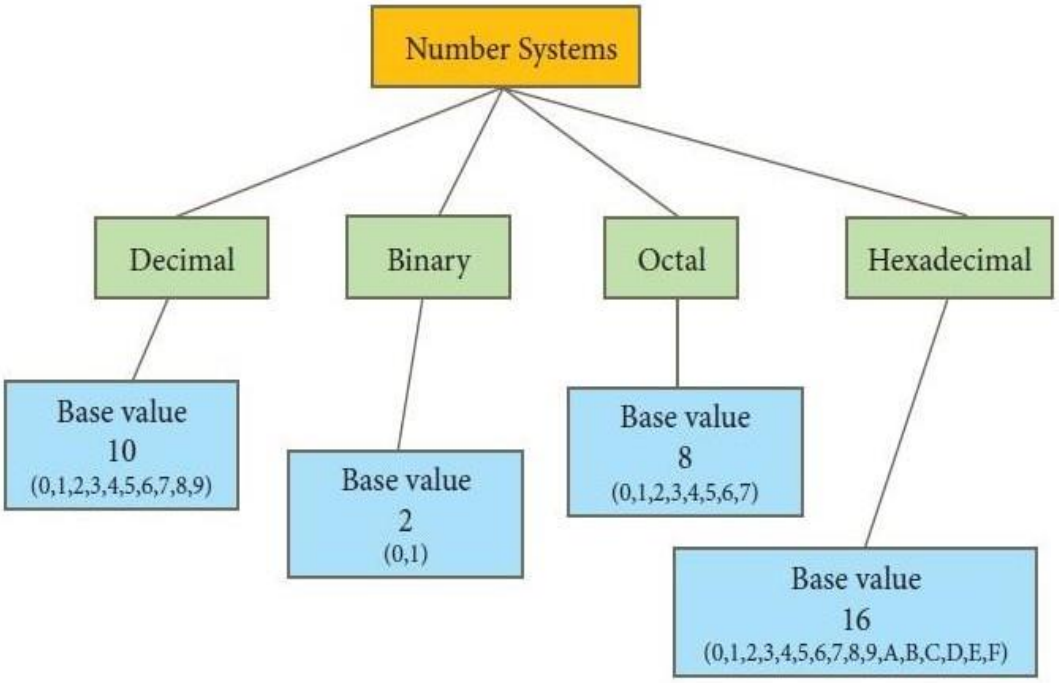
The radix or base of a number system is the number of digits or basic symbols used in that

Types of Number Systems

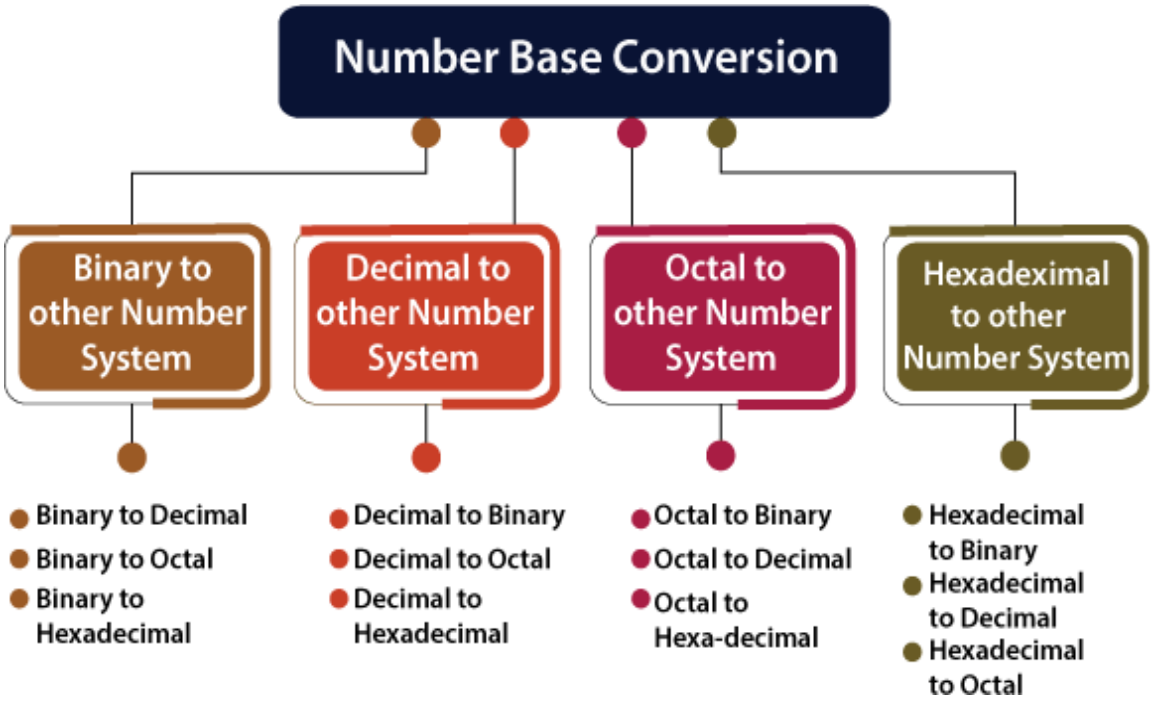
Some of the important types of number system are

1. Decimal Number System
2. Binary Number System
3. Octal Number System
4. Hexadecimal Number System





Number Systems



Number System	Base / Radix	Distinct Symbols
Binary	2	0, 1
Octal	8	0 – 7
Decimal	10	0 – 9
Hexadecimal	16	0 – 9, A, B, C, D, E, F

- In hexadecimal format, A=>10,B=>11,C=>12,D=>13,E=>14,F=>15.

Q2. Write the number system with their base and distinct symbols.

Ans.

Number System	Base / Radix	Distinct Symbols
Binary	2	0, 1
Octal	8	0 – 7
Decimal	10	0 – 9
Hexadecimal	16	0 – 9, A, B, C, D, E, F

Decimal (Radix 10)	Binary (Radix 2)	Octal (Radix 8)	Hexadecimal (Radix 16)
0	0	0	0
1	1	1	1
2		2	2
3		3	3
4		4	4
5		5	5
6		6	6
7		7	7
8			8
9			9
			A
			B
			C
			D
			E
			F

Number system conversion table from 1 to 15

Decimal Number Base-10	Binary Number Base-2	Octal Number Base-8	Hexadecimal Number Base-16
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

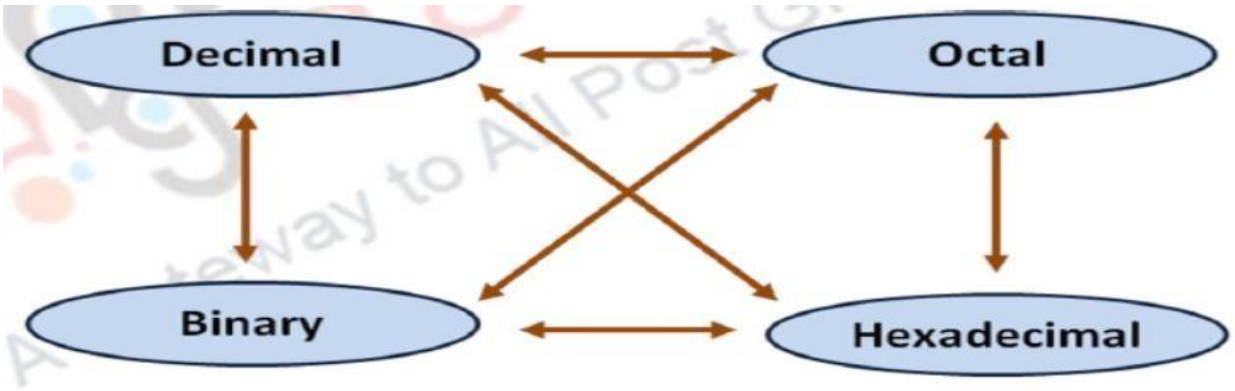


Figure-3: Possible number base interconversions

We will learn following interconversions

(1) DEC to BIN	(2) BIN to DEC
(3) DEC to OCT	(4) OCT to DEC
(5) OCT to BIN	(6) BIN to OCT
(7) DEC to HEX	(8) HEX to DEC
(9) HEX to BIN	(10) BIN to HEX

This program can perform the following conversion:

- Binary to Decimal
- Binary to Octal
- Binary to Hexa-Decimal
- Decimal to Binary
- Decimal to Octal
- Decimal to Hexa-Decimal
- Octal to Binary
- Octal to Decimal
- Octal to Hexa-Decimal
- Hexa-Decimal to Binary
- Hexa-Decimal to Decimal
- Hexa-Decimal to Octal

Total number of Digit = Base-1;

Decimal number systems :- The maximum value of single digit = **base (10)-1 = 9**. And the all 10 digits are 0, 1, 2,3, 4, 5, 6, 7, 8, 9.
Hexadecimal number systems :- The maximum value of single digit = **base (16)-1 = 15**. And the all 16 digits are 0, 1, 2,3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Where the value of A, B, C, D, E, F are 10, 11, 12, 13, 14, 15 correspondingly.
Octal number systems :- The maximum value of single digit = **base (8)-1 = 7**. And the all 8 digits are 0, 1, 2,3, 4, 5, 6, 7, 8.
Binary number systems :- The maximum value of single digit = **base (2)-1 = 1**. And the all 2 digits are 0, 1.

1) Binary Number System (Base 2)
The number system with base two is called as the binary number system. Only 0 & 1 are the symbols used in this system. 0 & 1are called as binary digits or bits. The left most bit is called the Most Significant Bit (MSB) The right most bit is called the least significant Bit (LSB).

A Binary number system has only two digits that are **0 and 1**.

Example

Decimal to Binary Conversion

The binary sequence $(1101)_2$ has the decimal equivalent:

Decimal Numbers 0-9 are represented in binary as: 0, 1, 10, 11, 100, 101, 110, 111, 1000, and 1001

$$(1101)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$
$$= 8 + 4 + 0 + 1$$
$$= (13)_{10}$$

14 can be written as 1110

19 can be written as 10011

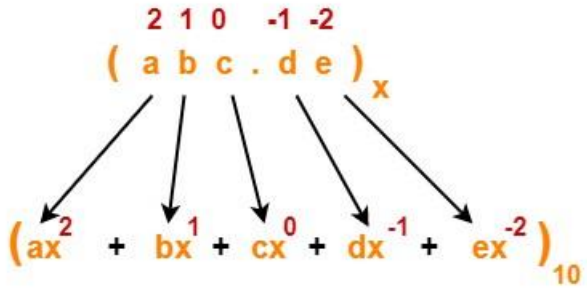
50 can be written as 110010

Converting to Base 10-

A given number can be converted from any base to base 10 using **Expansion Method**.

According to expansion method, if abc.de is any given number in base x, then its value in base 10 is given

$$(abc.de)_x = (ax^2 + bx + c + dx^{-1} + ex^{-2})_{10}$$



Expansion Method

PRACTICE PROBLEMS BASED ON CONVERSION TO BASE 10-

Convert the following numbers to base 10-

1. $(10010)_2$

2. $(254)_8$
3. $(AC)_{16}$
4. $(10010.101)_2$
5. $(254.7014)_8$
6. $(AC.FBA5)_{16}$
7. $(0.1402)_8$
8. $(0.ABDF)_{16}$

Solutions-

1. $(10010)_2$

$$(10010)_2 \rightarrow (\ ?\ )_{10}$$

9. $= (\ 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0\ )_{10}$
10. $= (\ 16 + 0 + 0 + 2 + 0\ )_{10}$
11. $= (\ 18\ )_{10}$

2. $(254)_8$

$$(254)_8 \longrightarrow (\ ?\ )_{10}$$

Using expansion method, we have-

$(254)_8$

$= (\ 2 \times 8^2 + 5 \times 8^1 + 4 \times 8^0\ )_{10}$

$= (\ 128 + 40 + 4\ )_{10}$

$= (\ 172\ )_{10}$

3. $(AC)_{16}$

$$(AC)_{16} \rightarrow (\ ?\ )_{10}$$

Using expansion method, we have-

$(AC)_{16}$

$= (A \times 16^1 + C \times 16^0)_{10}$

$= (10 \times 16 + 12 \times 1)_{10}$

$= (160 + 12)_{10}$

$= (172)_{10}$

4. $(10010.101)_2$

$$(10010.101)_2 \rightarrow (\ ?\ )_{10}$$

Using expansion method, we have-

$(10010.101)_2$

$= (1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3})_{10}$

$= (16 + 0 + 0 + 2 + 0 + 0.5 + 0.125)_{10}$

$= (18.625)_{10}$

5. $(254.7014)_8$

$$(254.7014)_8 \rightarrow (\ ?\ )_{10}$$

Using expansion method, we have-

$(254.7014)_8$

$= (2 \times 8^2 + 5 \times 8^1 + 4 \times 8^0 + 7 \times 8^{-1} + 0 \times 8^{-2} + 1 \times 8^{-3} + 4 \times 8^{-4})_{10}$

$= (128 + 40 + 4 + 0.875 + 0.0019 + 0.0009)_{10}$

$= (172.8778)_{10}$

6. $(AC.FBA5)_{16}$

$$(AC.FBA5)_{16} \rightarrow (\ ?\ )_{10}$$

Using expansion method, we have-

$(AC.FBA5)_{16}$

$= (A \times 16^{1} + C \times 16^{0} + F \times 16^{-1} + B \times 16^{-2} + A \times 16^{-3} + 5 \times 16^{-4})_{10}$

$= (10 \times 16 + 12 \times 1 + 15 \times 16^{-1} + 11 \times 16^{-2} + 10 \times 16^{-3} + 5 \times 16^{-4})_{10}$

$= (160 + 12 + 0.9375 + 0.0429 + 0.0024 + 0.0001)_{10}$

$= (172.9829)_{10}$

8. $(0.ABDF)_{16}$

$$(0.ABDF)_{16} \rightarrow (\ ?\ )_{10}$$

Using expansion method, we have-

$(0.ABDF)_{16}$

$= (0 \times 16^{0} + A \times 16^{-1} + B \times 16^{-2} + D \times 16^{-3} + F \times 16^{-4})_{10}$

$= (0 \times 1 + 10 \times 16^{-1} + 11 \times 16^{-2} + 13 \times 16^{-3} + 15 \times 16^{-4})_{10}$

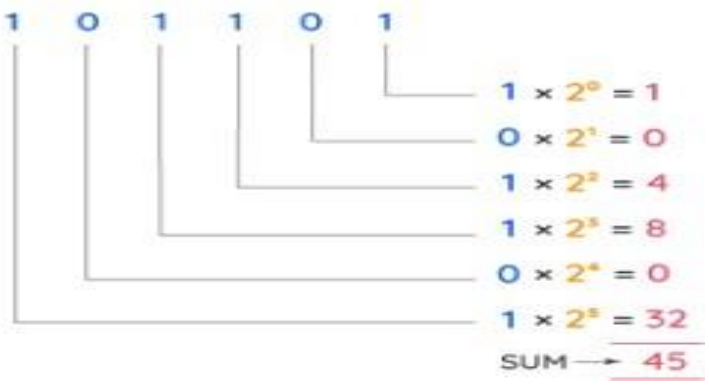
$= (0 + 0.625 + 0.0429 + 0.0032 + 0.0002)_{10}$

$= (0.6713)_{10}$

For example, $(101101)_2$ in decimal is

$= 1 \times 2^{5} + 0 \times 2^{4} + 1 \times 2^{3} + 1 \times 2^{2} + 0 \times 2^{1} + 1 \times 2^{0}$
$= 1 \times 32 + 0 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$
$= 32 + 8 + 4 + 1$
$= (45)_{10}$

Binary to Decimal Conversion Using Positional Notation Method



Binary to Decimal Conversion

$(11001)_2$

Decimal	256	128	64	32	16	8	4	2	1
25					1	1	0	0	1

16+8+1 = 25

Binary to Decimal Conversion

$(11001)_2$

4 3 2 1 0

1 1 0 0 1

$2^4 + 2^3 + \quad + 2^0$

$= 16 + 8 + 1$

$= 25$

Binary to Decimal Conversion

$(0.0101)_2$

$0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}$

$= 0 \times 0.5 + 1 \times 0.25 + 0 \times 0.125 + 1 \times 0.0625$

$= 0 + 0.25 + 0 + 0.0625$

$= 0.3125$

$2^{-1} = \frac{1}{2^1} = 2)\,10(.5$

$2^{-2} = \frac{1}{2^2} = \frac{1}{4}$

$0.0625 + 0.2500 = 0.3125$

Binary to Octal Conversion

$(100011)_2$

$100\ 011$

$(43)_8$

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

Binary to Octal Conversion

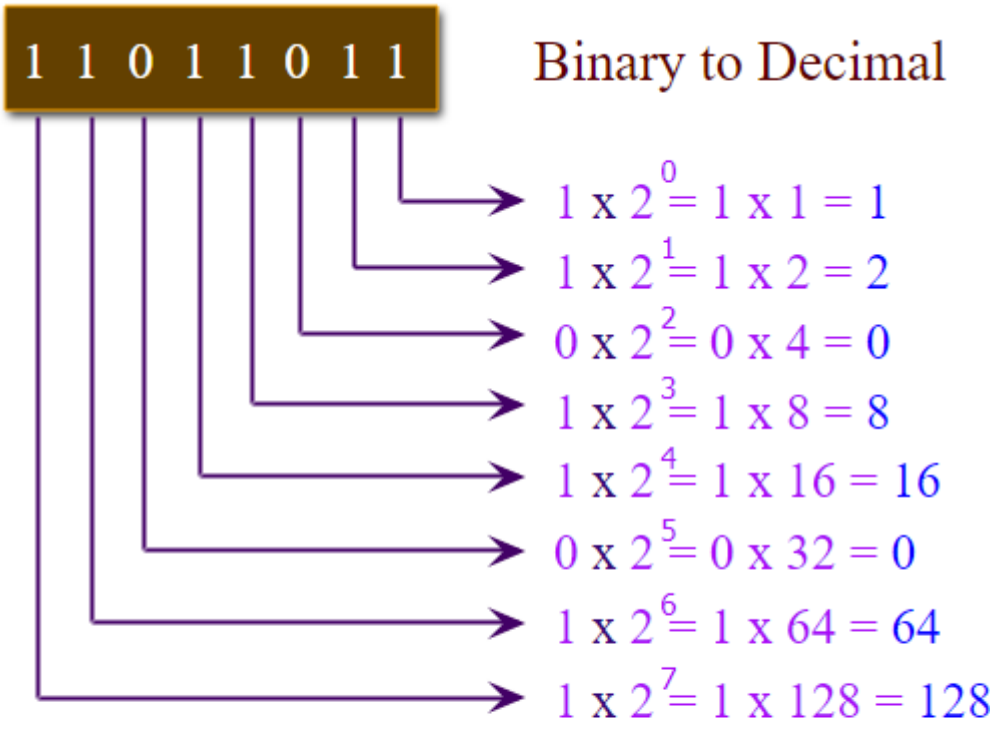
$(1101011.00101)_2$

1101011.00101

001101011.001010

$(153.12)_8$

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

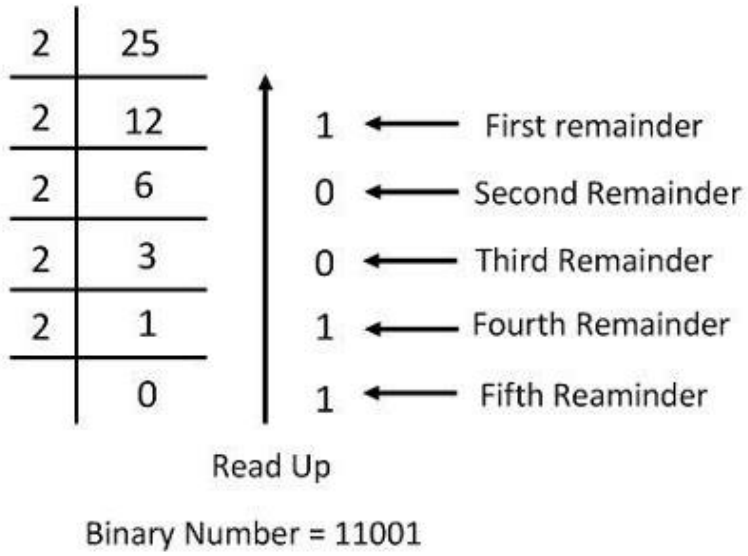


$1 + 2 + 8 + 16 + 64 + 128 = 219$

$(\ 11011011\ )_2 = (\ 219\ )_{10}$

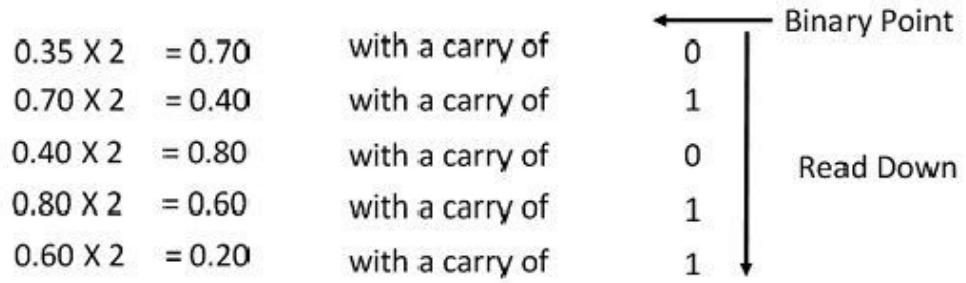
Decimal to Binary Conversion

For example – Consider the conversion of the decimal number 25 into its equivalent binary.



Binary Number = 11001

Consider the fractional binary number 0.35



Thus the fractional binary number is .01011, i.e., 0.01011.

1. Examples

1. $(42)_{10}$ = (____?____)$_2$

Solution:

$(42)_{10} = (\quad)_2$

2	42		
2	21	0	↑
2	10	1	↑
2	5	0	↑
2	2	1	↑
2	1	0	↑
	0	1	↑

$\therefore (42)_{10} = (101010)_2$

2. $(89.625)_{10}$ = (___ ? ___)$_2$

Solution:

$(89.625)_{10} = (\quad)_2$

2	89		
2	44	1	↑
2	22	0	↑
2	11	0	↑
2	5	1	↑
2	2	1	↑
2	1	0	↑
	0	1	↑

$0.625 \times 2 = 1.250 \Rightarrow 1 \downarrow$
$0.250 \times 2 = 0.500 \Rightarrow 0 \downarrow$
$0.500 \times 2 = 1.0 \Rightarrow 1 \downarrow$

$\therefore (89.625)_{10} = (1011001.101)_2$

represent 4 as (100) 2.

(4.25)

0.25 * 2 =0.50 //take 0 and move 0.50 to next step

0.50 * 2 =1.00 //take 1 and stop the process

0.25 = (01) $_2$

Combining both integral and fractional,
4.25 = (100.01) 2

Example

10.75

Integral Part

$10 = (1010)_2$

Fractional Part

0.75 * 2 =>1.50 // take 1 and move .50 to next step

0.50 * 2 =>1.00 // take 1 and stop the process because no remainder

$0.75 = (11)_2$

Combining both integral and fractional,

$10.75 = (1010.11)_2$

Example **2.33**

Integral Part
2 = (10) 2

Fractional Part

0.33 * 2 =>0.66 // take 0 and move .66 to next step

0.66 * 2 =>1.32 // take 1 and move .32 to next step

0.32 * 2 =>0.64 // take 0 and move .64 to next step

0.64 * 2 =>1.28 // take 1 and move .28 to next step

0.28 * 2 =>0.56 // take 0 and move .56 to next step

0.56 * 2 =>1.12 // take 1 and move .12 to next step

0.12 * 2 =>0.24 // take 0 and move .24 to next step

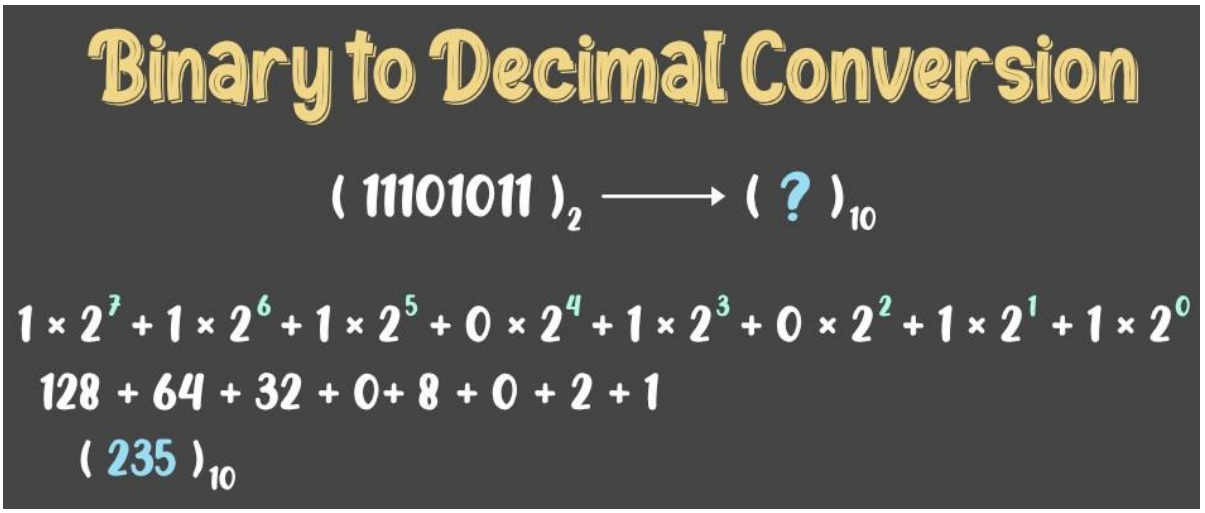
0.24 * 2 =>0.48 // take 0 and move .48 to next step

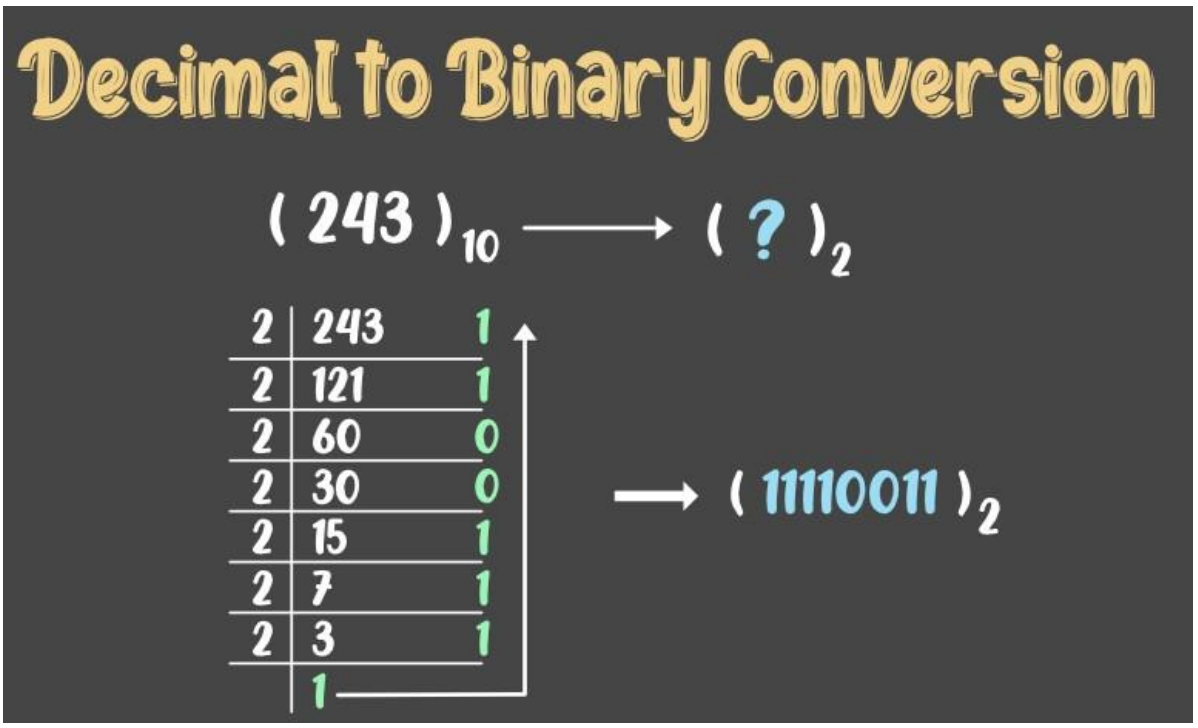
0.48 * 2 =>0.96 // take 0 and move .96 to next step

0.96 * 2 =>1.92 // take 1 and move .92 to next step

0.92 * 2 =>1.84 // take 1 and move .84 to next step

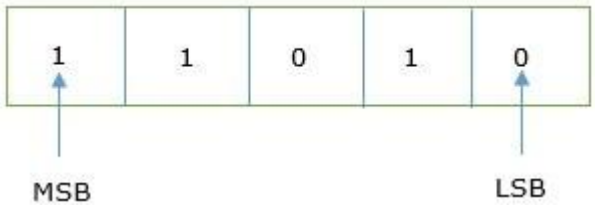
2.33 = (10.01010100011100101 1) 2


Binary to Decimal Conversion
(11101011)2 ⟶ (?)10
1 × 2^7 + 1 × 2^6 + 1 × 2^5 + 0 × 2^4 + 1 × 2^3 + 0 × 2^2 + 1 × 2^1 + 1 × 2^0
128 + 64 + 32 + 0+ 8 + 0 + 2 + 1
(235)10


Decimal to Binary Conversion
(243)10 ⟶ (?)2
2 | 243 | 1
2 | 121 | 1
2 | 60 | 0
2 | 30 | 0
2 | 15 | 1
2 | 7 | 1
2 | 3 | 1
1
⟶ (11110011)2

Example

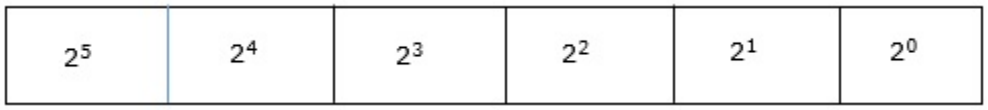
Binary TO Decimal Conversion



$11010_2 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 16 + 8 + 0 + 2 + 0$

$= 26_{10}$



2) Octal number system (Base 8)

The value of base is 8 so it is called octal number system.

Octal number system has only eight (8) digits from **0 to 7**. Every number (value) represents with 0,1,2,3,4,5,6 and 7 in this number system. The base of octal number system is 8, because it has only 8 digits.

This system uses digits 0 to 7 (i.e. 8 digits) to represent a number and the numbers are as a base of 8.
0, 1, 2, 3, 4, 5, 6, 7.

Exempla $(273)_8$, $(5644)_8$, $(0.5365)_8$, $(1123)_8$, $(1223)_8$.

For example, $(24)_8$ in decimal is
$= 2\times8^1+4\times8^0$
$= (20)_{10}$

Example

The Octal sequence $(547)_8$ has the decimal equivalent:

$$(547)_8 = 5\times8^2 + 4\times8^1 + 7\times8^0$$
$$= 5\times64 + 4\times8 + 7\times1$$
$$= 320 + 32 + 7$$
$$= (359)_{10}$$

$726_8 = 7\times8^2 + 2\times8^1 + 6\times8^0$

$= 448 + 16 + 6$

$= 470_{10}$

Binary to Octal Conversion

$(100011)_2$

$\overleftarrow{100}\ \overleftarrow{011}$

$(43)_8$

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

Binary to Octal Conversion

$(1101011.00101)_2$

1101011.00101

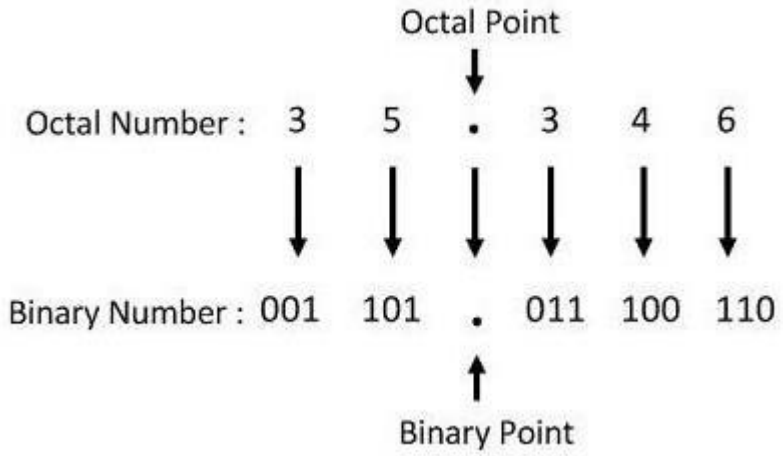
001101011.001010

$(153.12)_8$

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

Octal to Binary Conversion

octal number 35.346 to its equivalent binary number



Binary to Octal Conversion

binary number 010100111.100011

Octal to Binary Conversion

$(25)_8$

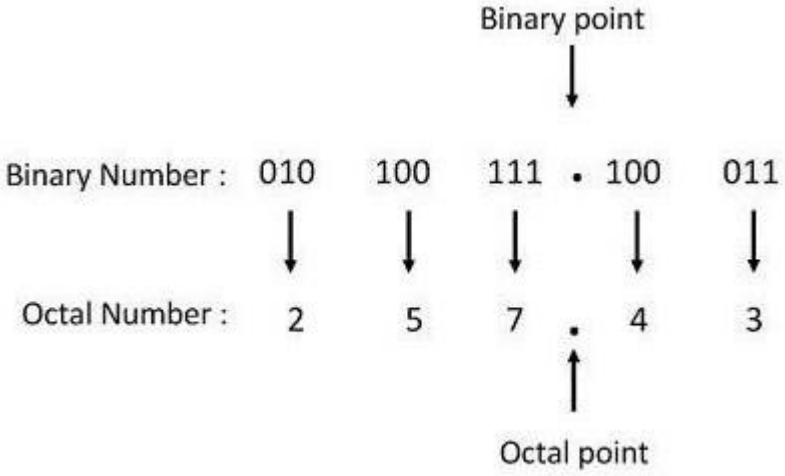
$(010101)_2 = (10101)_2$

$(123)_8$

$(001010011)_2 = (1010011)_2$

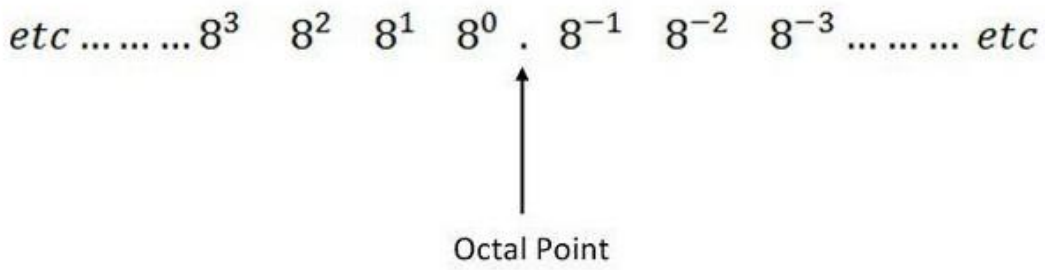
$(345.12)_8$ $(011100101.001010)_2$

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



Octal to Decimal Conversion

In octal number system, each digit position has the **weight eight** regarding **power eight** shown in the figure below.



Example – Consider the octal number 354.42 into its equivalent decimal number. The integer part 354 converts to octal shown below.

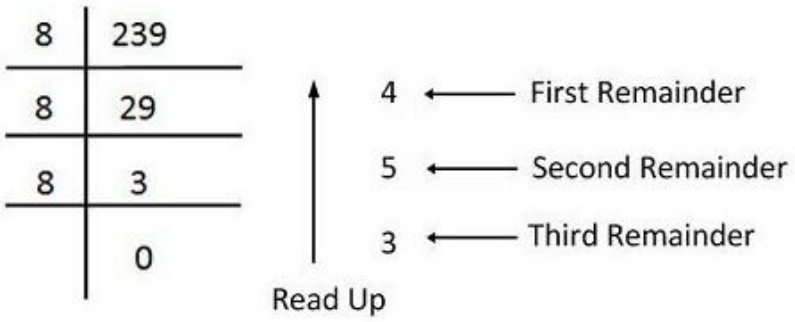
$$3(8^2) + 5(8^1) + 4(8^0) = 236$$

And the fractional parts 0.42 converts to octal

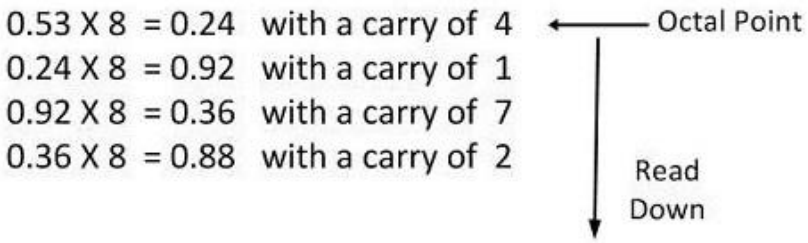
$$0.[4(8^{-1}) + 2(8^{-2})] = 0.53125$$

The **decimal number system is 236.53125.**

Example: Consider the conversion of the decimal number 236.53. The conversion of integer part is shown below.



And the fraction part



Thus the **octal number is 354.4172.**

Q. What is the Octal Representation of Decimal Number 8?

A. The octal representation of Decimal Number 8 is 10.

Q. How do you convert from Decimal to Octal?

A. Divide the given Decimal number by 8 until the quotient is less than 8. Now note all the remainders from bottom to top.

Example – Convert decimal number 98 into octal number.

Answer- First convert it into binary or hexadecimal number,

$= (98)_{10}$

$= (1x2^6+1x2^5+0x2^4+0x2^3+0x2^2+1x2^1+0x2^0)_{10}$ or $(6x16^1+2x16^0)_{10}$

Because base of binary and hexadecimal are 2 and 16 respectively.

$= (110\ 0010)_2$

or $(62)_{16}$

Then convert each digit of hexadecimal number into 4 bit of binary number whereas convert each group of 3 bits from least significant in binary number.

$= (001\ 100\ 010)_2$

or $(0110\ 0010)_2$

$= (001\ 100\ 010)_2$

$= (1\ 4\ 2)_8$

$= (142)_8$

Decimal to Octal Conversion Solved Examples

Few solved examples of Decimal number to Octal numbers are given below

Q1. Convert 256 to Octal Number.

Answer:

1st Step: 256 ÷ 8 = Q(32) R(0)
2nd Step: 32 ÷ 8 = Q(4) R (0)
3rd Step: 4 ÷ 8 = Q(0) R (4)

256 Octal Equivalent number is $\mathbf{(400)_8}$

Decimal No	Quotient	Remainder
256 ÷ 8	32	0
32 ÷ 8	4	0
4 ÷ 8	0	4

Q2. Convert 100 to Octal Number.

Answer:
1st Step: 100 ÷ 8 = Q(12) R(4)
2nd Step: 12 ÷ 8 = Q(1) R (4)
3rd Step: 1 ÷ 8 = Q(0) R (1)

100 Octal Equivalent number is $\mathbf{(144)_8}$

Decimal No	Quotient	Reminder
100 ÷ 8	100	4
12 ÷ 8	12	4
1 ÷ 8	0	1

Q3. Convert 2980 to Octal Number

Answer:
1st Step: 2980 ÷ 8 = Q(372) R(4)
2nd Step: 372 ÷ 8 = Q(46) R (4)
3rd Step: 46 ÷ 8 = Q(5) R (6)
4th Step: 5 ÷ 8 = Q(0) R (5)

2980 Octal Equivalent number is $\mathbf{(5644)_8}$

Decimal No	Quotient	Reminder
290 ÷ 8	372	4
372 ÷ 8	46	4
46 ÷ 8	5	6
5 ÷ 8	0	5

Decimal Number: $(540)_{10}$

8	540	
8	67	4
8	8	3
8	1	0
	0	1

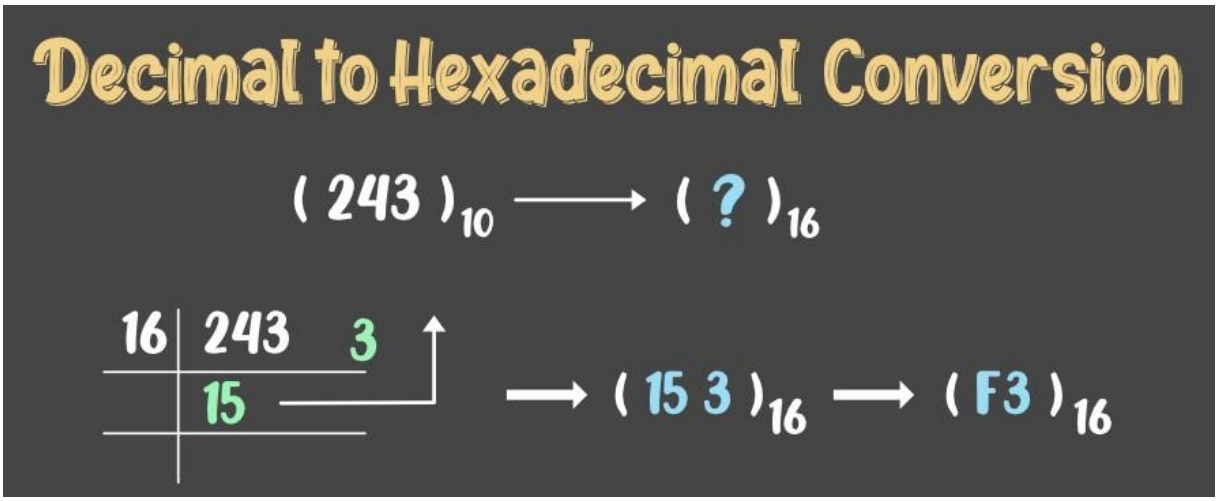
$(540)_{10} = (1034)_8$ Octal Number

Convert 440 to Octal Number

Decimal To Octal Number Conversion

8	440	
8	55	(0)
8	6	(7)
		(6)

The Octal Number of 440 is $(670)_8$



1. $(2020)_{10}$

$(2020)_{10} \rightarrow (?)_{16}$

16	2020	
16	126 ,	4
	7 ,	14 = E

From here, $(2020)_{10} = (7E4)_{16}$

1. $(1423)_{10}$ = (___?___)$_{16}$

Solution:

$(1423)_{10} = (\quad)_{16}$

16	1423	
16	88	F ↑
16	5	8 ↑
	0	5 ↑

$\therefore (1423)_{10} = (58F)_{16}$

2. $(93419)_{10} = (\underline{\quad ? \quad})_{16}$

Solution:

$(93419)_{10} = \left(\quad\right)_{16}$

16	93419	
16	5838	B ↑
16	364	E ↑
16	22	C ↑
16	1	6 ↑
	0	1 ↑

$\therefore (93419)_{10} = \left(16CEB\right)_{16}$

1. Examples

1. $(11011001)_2 = (\underline{\quad ? \quad})_8$

Solution:

$(11011001)_2 = \left(\quad\right)_8$

011 011 001
3 3 1

∴ (11011001)_2 = (underline{331})_8

2. $(10110011)_2 = (\underline{\quad ? \quad})_8$

Solution:

$(10110011)_2 = \left(\quad\right)_2$

010 110 011
2 6 3

$\therefore (10110011)_2 = \left(263\right)_2$

1. Examples

1. $(1111010100)_2 = (\underline{\quad ? \quad})_{16}$

Solution:

$(1111010100)_2 = \left(\quad\right)_{16}$

0011 1101 0100
3 D 4

∴ (1111010100)_2 = (underline{3D4})_16

2. $(1111010100111110)_2 = (\_\_\_?\_\_\_)_{16}$

Solution:

$$(1111010100111110)_2 = \left(\quad\right)_{16}$$

1111 0101 0011 1110
F 5 3 E

$$\therefore (1111010100111110)_2 = \left(F53E\right)_{16}$$

1. Examples

1. $(123)_8 = (\_\_\_?\_\_\_)_2$

Solution:
(123)_8 = (underline{" "})_2

1 2 3
001 010 011

:. (123)_8 = (underline{1010011})_2

2. $(4321)_8 = (\_\_\_?\_\_\_)_2$

Solution:

$$(4321)_8 = \left(\quad\right)_2$$

4 3 2 1
100 011 010 001

$$\therefore (4321)_8 = \left(100011010001\right)_2$$

1. Examples

1. $(567)_8$ = (___?___)$_{16}$

Solution:

$(567)_8 = (\quad)_{16}$

First convert octal to binary

$(567)_8 = (\quad)_2$

5 6 7
101 110 111

$\therefore (567)_8 = (101110111)_2$

Now convert binary to hexadecimal

$(101110111)_2 = (\quad)_{16}$

0001 0111 0111
1 7 7

$\therefore (101110111)_2 = (177)_{16}$

$\therefore (567)_8 = (177)_{16}$

2. $(4321)_8$ = (___?___)$_{16}$

Solution:

$(4321)_8 = \left(\quad\right)_{16}$

First convert octal to binary

$(4321)_8 = \left(\quad\right)_2$

4 3 2 1
100 011 010 001

$\therefore (4321)_8 = \left(100011010001\right)_2$

Now convert binary to hexadecimal

$(100011010001)_2 = \left(\quad\right)_{16}$

1000 1101 0001
8 D 1

$\therefore (100011010001)_2 = \left(8D1\right)_{16}$

$\therefore (4321)_8 = \left(8D1\right)_{16}$

1. Examples

1. $(A10)_{16}$ = (___?___)$_{10}$

Solution:

$(A10)_{16} = \left(\quad\right)_{10}$

$A10$

$= A \times 16^2 + 1 \times 16^1 + 0 \times 16^0$

$= 10 \times 256 + 1 \times 16 + 0 \times 1$

$= 2576$

:. (A10)_16 = (underline{2576})_10

2. $(BCA)_{16}$ = (___?___)$_{10}$

Solution:

$(BCA)_{16} = \left(\quad\right)_{10}$

BCA

$= B \times 16^2 + C \times 16^1 + A \times 16^0$

$= 11 \times 256 + 12 \times 16 + 10 \times 1$

$= 3018$

$\therefore (BCA)_{16} = \left(3018\right)_{10}$

1. Examples

1. $(283)_{16}$ = (___?___)$_2$

Solution:

$(283)_{16} = (\quad)_2$

2 8 3
0010 1000 0011

$\therefore (283)_{16} = (1010000011)_2$

2. $(A35)_{16}$ = (___?___)$_2$

Solution:

$(A35)_{16} = (\quad)_2$

A 3 5
1010 0011 0101

$\therefore (A35)_{16} = (101000110101)_2$

1. Examples

1. $(951)_{16}$ = (___?___)$_8$

Solution:

$(951)_{16} = (\quad)_8$

First convert hexadecimal to binary

$(951)_{16} = (\quad)_2$

9 5 1
1001 0101 0001

$\therefore (951)_{16} = (100101010001)_2$

Now convert binary to octal

$(100101010001)_2 = (\quad)_8$

100 101 010 001
4 5 2 1

$\therefore (100101010001)_2 = (4521)_8$

$\therefore (951)_{16} = (4521)_8$

2. $(FC3A)_{16}$ = (___ ? ___)$_8$

Solution:

$(FC3A)_{16} = (\quad)_8$

First convert hexadecimal to binary

$(FC3A)_{16} = (\quad)_2$

F C 3 A
1111 1100 0011 1010

$\therefore (FC3A)_{16} = (1111110000111010)_2$

Now convert binary to octal

$(1111110000111010)_2 = (\quad)_8$

001 111 110 000 111 010
1 7 6 0 7 2

$\therefore (1111110000111010)_2 = (176072)_8$

$\therefore (FC3A)_{16} = (176072)_8$

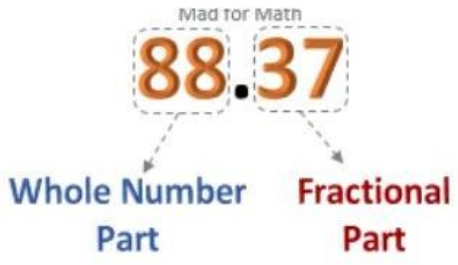
$(92)_{10} = (134)_8$

DESCRIPTIONS

Divide the number repeatedly by **8** until the quotient becomes **0**.

Remainders

8 | 92 — 4
8 | 11 — 3
8 | 1 — 1
0



$(88)_{10} = (130)_8$

Remainders

8 | 88		0
8 | 11		3
8 | 1		1
0		

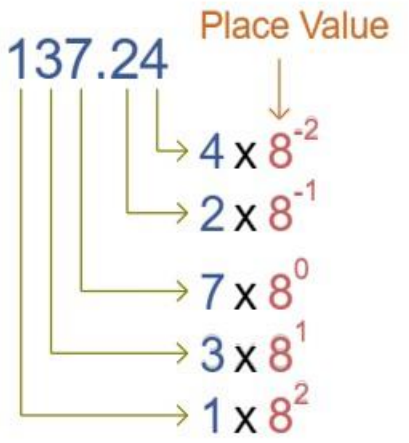
$(0.37)_{10} = (0.275...)_8$

- $0.37 \times 8 = 2.96$
- $0.96 \times 8 = 7.68$
- $0.68 \times 8 = 5.44$
- .....

$$(88.37)_{10} = (88)_{10} + (0.37)_{10}$$
$$= (130)_8 + (0.275...)_8$$
$$= (130.275...)_8$$

$(137.24)_8 = (95.3125)_{10}$

DESCRIPTIONS



$(137.24)_8 = (1 \times 8^2) + (3 \times 8^1) + (7 \times 8^0) + (2 \times 8^{-1}) + (4 \times 8^{-2})$

$= 64 + 24 + 7 + \frac{2}{8} + \frac{4}{64}$

$= (95.3125)_{10}$

3) Decimal number system (Base 10)

The value of base is 10 so it is called decimal number system.

Decimal number system has only ten (10) digits from **0 to 9**. Every number (value) represents with 0,1,2,3,4,5,6, 7,8 and 9 in this number system. The base of decimal number system is 10, because it has only 10 digits.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

DECIMAL TO DECIMAL

$(145)_{10}$

$= 1*(10^2) + 4*(10^1) + 5*(10^0)$

$= 100+40+5$

$= (145)_{10}$

1. Decimal Number System

$$(123)_{10} = 1x10^2 + 2x10^1 + 3x10^0$$
$$= 100 + 20 + 3$$
$$= (123)_{10}$$

For example: 10285 has place values as
$(1 \times 10^4) + (0 \times 10^3) + (2 \times 10^2) + (8 \times 10^1) + (5 \times 10^0)$
$1 \times 10000 + 0 \times 1000 + 2 \times 100 + 8 \times 10 + 5 \times 1$

$10000 + 0 + 200 + 80 + 5$

10285

For example, the value of 786 is
$= 7 \times 10^2 + 8 \times 10^1 + 6 \times 10^0$
$= 700 + 80 + 6$

10^5	10^4	10^3	10^2	10^1	10^0

4) Hexadecimal number system (Base 16)

The value of base is 16 so it is called hexadecimal number system.

A Hexadecimal number system has sixteen (16) alphanumeric values from **0 to 9** and **A to F**. Every number (value) represents with 0,1,2,3,4,5,6, 7,8,9,A,B,C,D,E and F in this number system.

The base of hexadecimal number system is 16, because it has 16 alphanumeric values. Here **A is 10**, **B is 11**, **C is 12**, **D is 13**, **E is 14** and **F is 15**.

Example $(FAC2)_{16}$, $(564)_{16}$, $(0ABD5)_{16}$, $(1123)_{16}$, $(11F3)_{16}$.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

16^5	16^4	16^3	16^2	16^1	16^0

$27FB_{16} = 2\times16^3 + 7\times16^2 + 15\times16^1 + 10\times16^0$

$= 8192 + 1792 + 240 + 10$

$= 10234_{10}$

Hexadecimal to Binary Conversion Method

hexadecimal number 9AF which is converted into a binary digit.

Hexadecimal number :	9	A	F
Binary Number :	1001	1010	1111

Binary to Hexadecimal Conversion Methods

Binary number :	1110	1000	1101	0110
Hexadecimal number:	E	8	D	6

Hexadecimal to Decimal Conversion Method

The base of the hexadecimal number system is 16, therefore the weights corresponding to various positions of the digits will be as shown below.

$$16^2 \quad 16^1 \quad 16^0 \bullet 16^{-1} \quad 16^{-2} \quad 16^{-3}$$

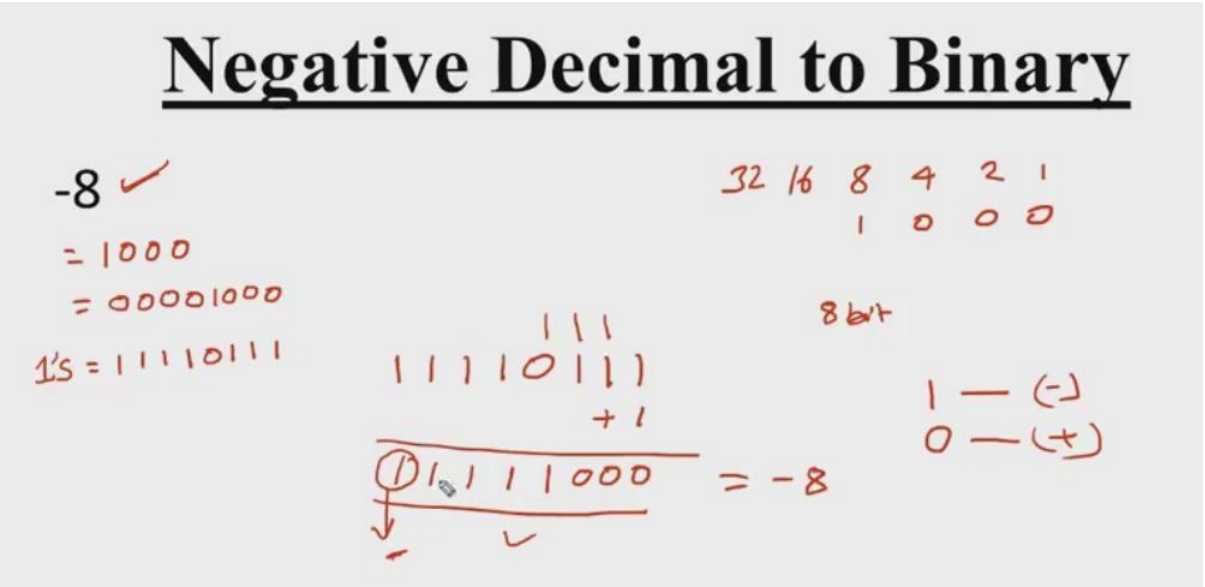
Hexadecimal Point

For instance, consider the conversion of hexadecimal number E8F6.27 into its equivalent binary number.

$$E8F6.27 = E(16^3) + 8(16^2) + F(16^1) + 6(16^0) + 2(16^{-1}) + 7(16^{-2})$$

$$E8F6.27 = 14(4096) + 8(256) + 15(16) + 6(1) + 2 \times \frac{1}{16} + 7 \times \frac{1}{16^2}$$

$$E8F6.27 = 59638.1523437$$



Binary to Hexadecimal Conversion

Binary	Hexadecimal
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

$(100100)_2$

16/2 = 8/2 = 4/2 = 2/2 = 1

100100

00100100

$(24)_{16}$

Binary to Hexadecimal Conversion

$(1101011.00101)_2$

1101011.00101

01101011.00101000

$(6B.28)_{16}$

Hexadecimal to Binary Conversion

$(23)_{16}$

$(00100011)_2 = (100011)_2$

$(C6)_{16}$

$(11000110)_2$

$(23.4)_{16}$

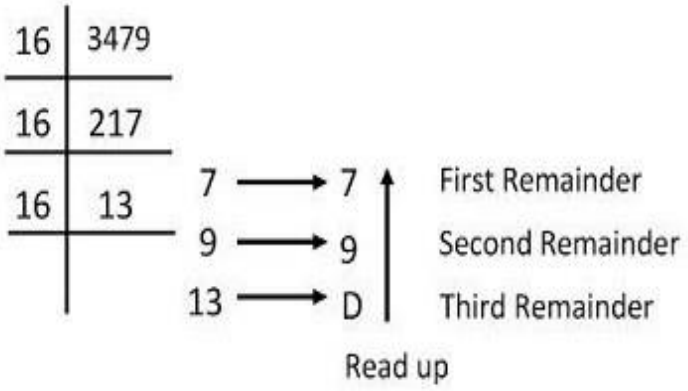
$(00100011.0100)_2$

2 3 . 4

Binary	Hexadecimal
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

Decimal to Hexadecimal Conversion Method

Consider the conversion of the decimal number 3749 into its hexadecimal equivalent number.



The third reminder 13 is equivalent to D in a hexadecimal number system. Thus the equivalent hexadecimal number D97

Hexadecimal to Octal Conversion:

Given, $(FB2)_{16}$ is a hexadecimal number.

F → 1111, B → 1011, 2 → 0010

1111 1011 0010

Now group them from right to left, each having 3 digits.

111, 110, 110, 010

111 → 7, 110 → 6, 110 → 6, 010 → 2

Hence, **$FB2_{16} = 7662_8$**

Input Data :

Hexa Decimal = 48

Obejective :

Convert Hexa Decimal to Decimal Value

Solution with Steps :

$$48_{16} = (4 \times 16^1) + (8 \times 16^0)$$

$$48_{16} = 64 + 8$$

$$48_{16} = 72_{10}$$

Hex ⇄ Octal Conversions with Steps

Input Data :
Hexa Decimal = 48

Obejective :
Convert Hexa Decimal to Octal Value

Solution with Steps :
Convert Hexa Decimal number to binary

```
  4      8
0100  1000
```

$$48_{16} = 01001000_2$$

Convert Binary Number into Octal Number
Split the binary number from left to right each group 3 bits

```
 001   001   000
  1     1     0
```

$$01001000_2 = 110_8$$

Decimal	Binary	Octal	Hexadecimal
0	0000	000	0000
1	0001	001	0001
2	0010	002	0002
3	0011	003	0003
4	0100	004	0004
5	0101	005	0005
6	0110	006	0006
7	0111	007	0007
8	1000	010	0008
9	1001	011	0009
10	1010	012	A
11	1011	013	B
12	1100	014	C
13	1101	015	D
14	1110	016	E
15	1111	017	F

Q3. Explain the binary number system .

Ans. The number system with base two is known as binary number system.Only two symbols are used to represent numbers in this system i.e.,0 and 1,which are known as bits.

Q4. What is octal and hexadecimal number system?

Ans.Octal number system:The number system with base eight is known as octal number system.In this system,eight symbols i.e.,0,1,2,…..7 are used to represent the number.

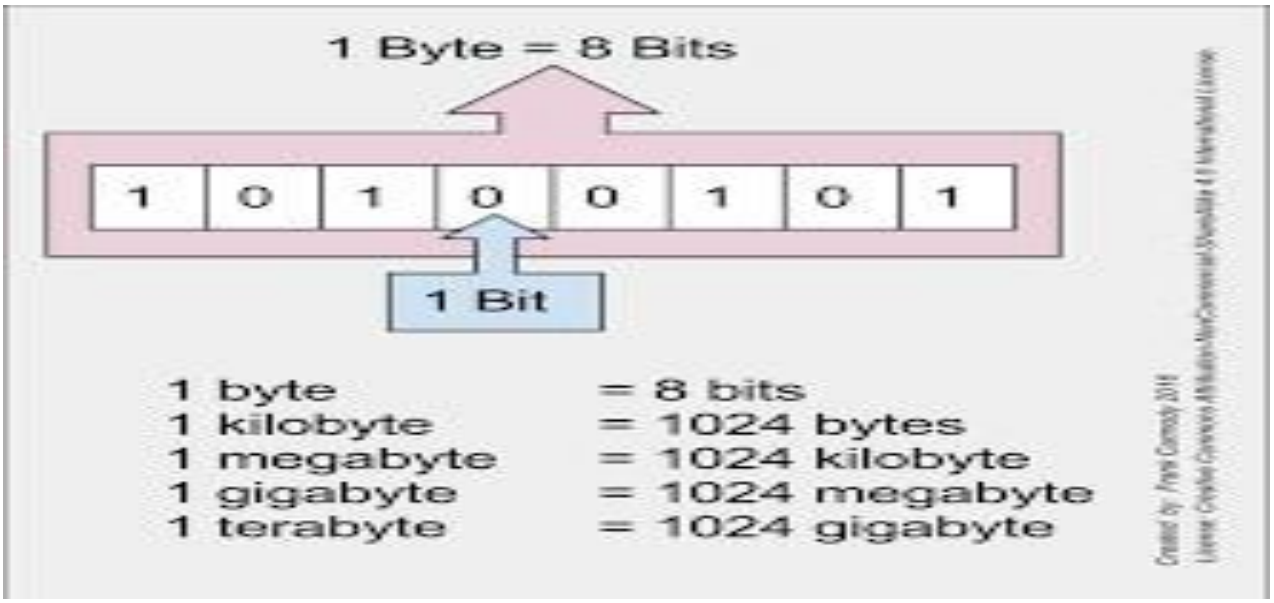
Hexadecimal number system: This system is very popular in computer cases.The base for hexadecimal number system is 16 which requires 16 distinct symbols to represent the number. There are numerals 0 to 9 and alphabets A to F.

Bit

The term 'bit' is a contraction of the words 'binary' and 'digit'. It is the smallest unit of memory or instruction that can be given or stored on a computer. A bit is either a 0 or a 1. The number in the above example is a 6-bit number as it has 6 binary digits (0s and 1s).

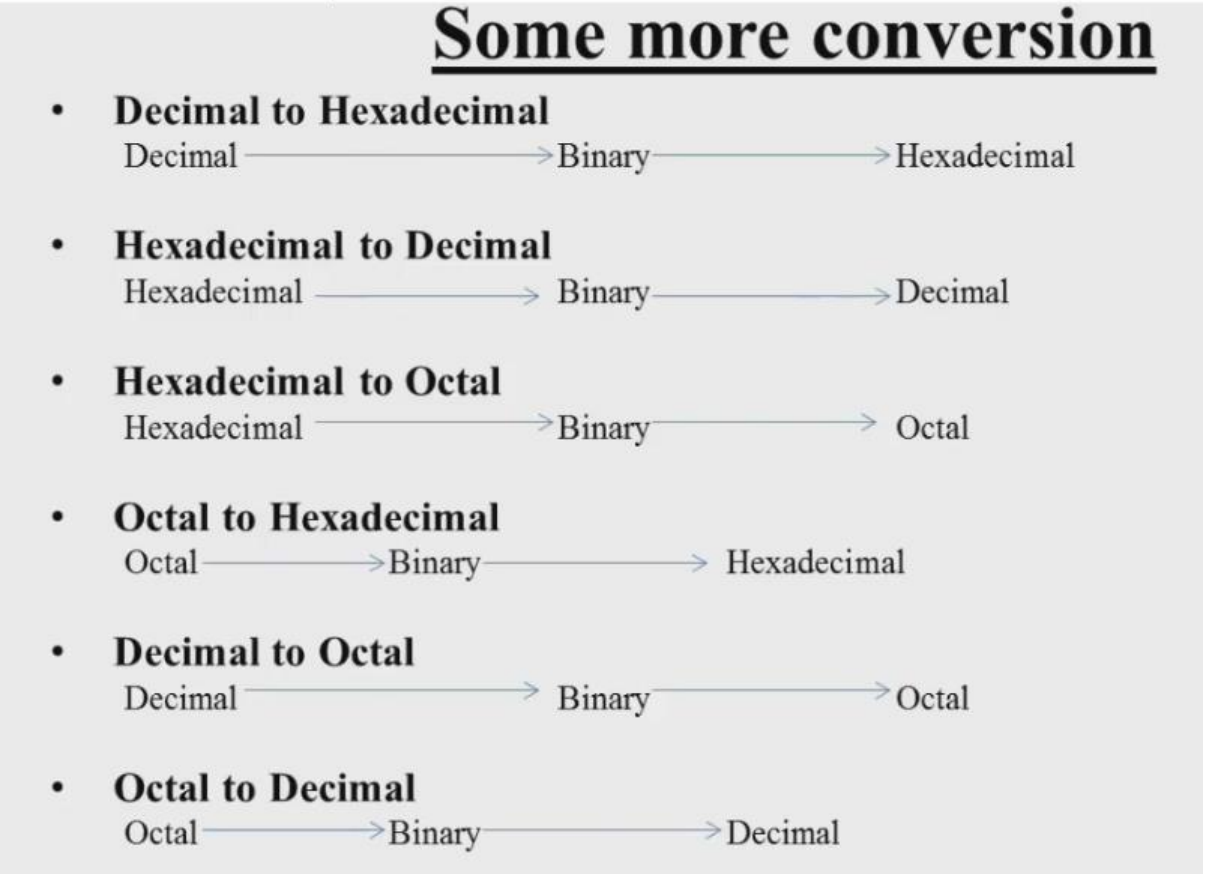
Byte

A group of 8 bits like 01100001 is a byte. Combination of bytes comes with various names like the kilobyte. One kilobyte is a collection of 1000 bytes. A word or letter like 'A' or 'G' is worth 8 bits or one byte. One thousand bytes make up a kilobyte (one thousand letters approximately). 1024 kilobytes form a Megabyte (Mb) and so on.



- 1 byte (B) = 8 bits
- 1 Kilobytes (KB) = 1024 bytes
- 1 Megabyte (MB) = 1024 KB
- 1 Gigabyte (GB) = 1024 MB
- 1 Terabyte (TB) = 1024 GB

- 1 Exabyte (EB) = 1024 PB
- 1 Zettabyte = 1024 EB
- 1 Yottabyte (YB) = 1024 ZB



The given hexadecimal number (1E.53)16 is equivalent to _____

a) (35.684)8
b) (36.246)8
c) (34.340)8
d) (35.599)8

.The octal number (651.124)8 is equivalent to _____

a) (1A9.2A)16
b) (1B0.10)16
c) (1A8.A3)16
d) (1B0.B0)16

The octal equivalent of the decimal number (417)10 is _____

a) (641)8

b) (619)8
c) (640)8
d) (598)8
. Convert the hexadecimal number (1E2)16 to decimal.
a) 480
b) 483
c) 482
d) 484

(170)10 is equivalent to _____
a) (FD)16
b) (DF)16
c) (AA)16
d) (AF)16
6. Convert (214)8 into decimal.
a) (140)10
b) (141)10
c) (142)10
d) (130)10

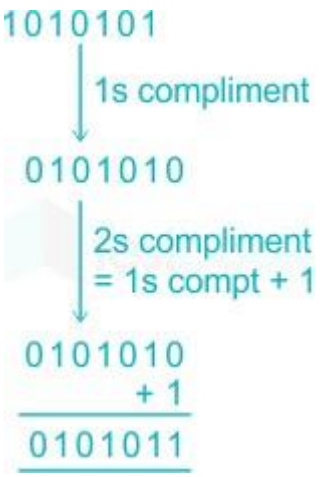
7. Convert (0.345)10 into an octal number.
a) (0.16050)8
b) (0.26050)8
c) (0.19450)8
d) (0.24040)8
8. Convert the binary number (01011.1011)2 into decimal.
a) (11.6875)10
b) (11.5874)10
c) (10.9876)10
d) (10.7893)10
9. Octal to binary conversion: (24)8 =?
a) (111101)2
b) (010100)2
c) (111100)2
d) (101010)2

10. Convert binary to octal: (110110001010)2 =?
a) (5512)8

b) (6612)8
c) (4532)8
d) (6745)8

The 2's complement of 1010101 is _____.

1. 0101010
2. 1110011
3. 0101011
4. 1101010



In Binary-coded Decimal (BCD) systems, the decimal number 81 is represented as

1. 10000001
2. 10100010
3. 01010001
4. 00011000

- In BCD each decimal digit is represented by a 4-bit binary number.
- The binary representation of 8 → 1000
- The binary representation of 1 → 0001
- $(81)_{10}$ = 10000001

Convert the following decimal number to 8-bit binary.
187

A. 10111011_2

B. 11011101_2

C. 10111101_2

D. 10111100_2

Convert binary 111111110010 to hexadecimal.

A. $EE2_{16}$

B. $FF2_{16}$

C. $2FE_{16}$

D. $FD2_{16}$

Convert the binary number 1001.0010_2 to decimal.

A. 90.125

B. 9.125

C. 125

D. 12.5

Convert 59.72_{10} to BCD.

A. 111011

B. 01011001.01110010

C. 1110.11

D. 0101100101110010

Convert $8B3F_{16}$ to binary.

A. 35647

B. 011010

C. 1011001111100011

D. 1000101100111111

The binary code of $(21.125)_{10}$ is

1. 10101.001
2. 10100.001
3. 10101.010
4. 10100.111

The decimal number $(57.375)_{10}$ when converted to binary number takes the form:

1. $(111001.011)_2$
2. $(100111.110)_2$
3. $(110011.101)_2$
4. $(111011.011)_2$

The decimal equivalent of the binary number $(1101)_2$ is

1. 9
2. 11
3. 13
4. 15

The hexadecimal representation of 657_8 is

1. D78
2. 1AF
3. D71
4. 32F

Concept:

Hexadecimal number: In this, value of the base is 16. Each digit is represented by 4-bit binary no.

Octal number: For octal number, value of base is 8. Each digit of an octal number is represented by 3-bit binary no.

Explanation:

Octal number = 657

Binary representation for this number (each digit of a octal number is converted into 3 binary bits)

So, 657 in binary is equivalent to 110 101 111

Now group this binary number into 4 bits starting from right to left.

i.e. 0001 1010 1111

Hexadecimal representation for this number is : 1AF

What is Digital Electronics?
a) Field of electronics involving the study of digital signal
b) Engineering of devices that digital signal
c) Engineering of devices that produce digital signal
d) All pf the mentioned

Which of the following is correct for Digital Circuits?
a) Less susceptible to noise or degradation in quality
b) Use transistors to create logic gates to perform Boolean logic
c) Easier to perform error detection and correction with digital signal
d) All of the mentioned

3. What is a Circuit?
a) Open-loop through which electrons can pass
b) Closed-loop through which electrons can pass
c) Closed-loop through which Neutrons can pass
d) None of the mentioned

Which of the following is an example of a digital Electronic?
a) Computers
b) Information appliances
c) Digital cameras
d) All of the mentioned

. Which of the following is a type of digital logic circuit?
a) Combinational logic circuits
b) Sequential logic circuits
c) Both a & b
d) None of the mentioned

Which characteristic of IC in Digital Circuits represents a function of the switching time of a particular transistor?
a) Fan – out
b) Fan – in
c) Power dissipation
d) Propagation delay

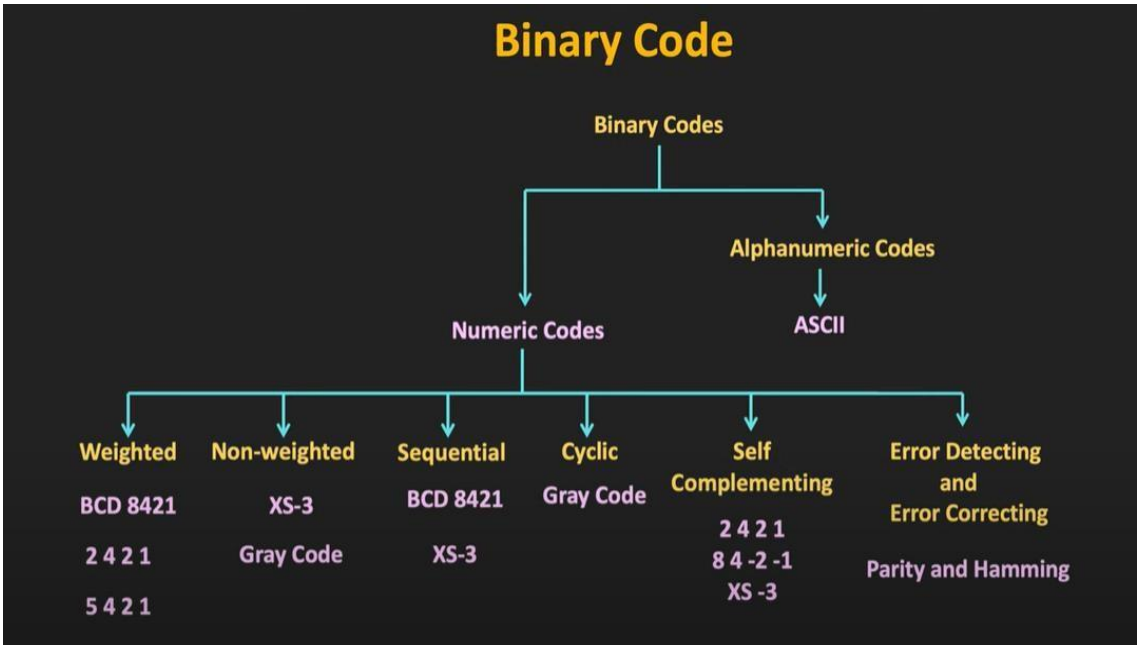
When can one logic gate drive many other logic gates in Digital Electronics?
a) When its output impedance is low and the input impedance is low
b) When its output impedance is high and the input impedance is high
c) When its output impedance is high and the input impedance is low
d) When its output impedance is low and the input impedance is high

Classification of binary codes

The codes are broadly categorized into following four categories.

- Weighted Codes
- Non-Weighted Codes
- Binary Coded Decimal Code
- Alphanumeric Codes
- Error Detecting Codes

- Error Correcting Codes



Binary Coded Decimal (BCD)/8421/4 bits Code

BCD code plays an important role in digital circuits. The BCD stands for Binary Coded Decimal Number. In BCD code, each digit of the decimal number is represented as its equivalent binary number. So, the LSB and MSB of the decimal numbers are represented as its binary numbers. There are the following steps to convert the binary number to BCD:

1. First, we will convert the binary number into decimal.
2. We will convert the decimal number into BCD.

Example 1: $(11110)_2$

1. First, convert the given binary number into a decimal number.

Binary Number: $(11110)_2$

1)	$(11110)_2$	$((1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0))_{10}$
2)	$(11110)_2$	$(16 + 8 + 4 + 2 + 0)_{10}$
3)	$(11110)_2$	$(30)_{10}$

Steps	Decimal Number	Conversion
Step 1	30_{10}	$(0011)_2$ $(0000)_2$
Step 2	30_{10}	$(00110000)_{BCD}$

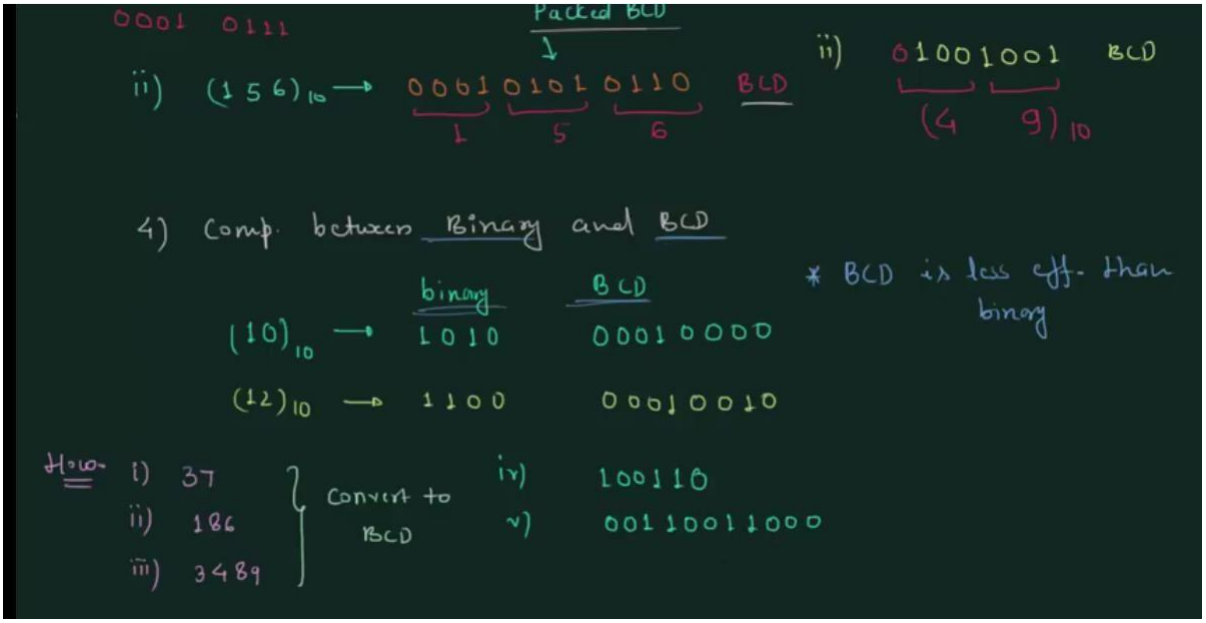
2. Now, we convert the decimal to the BCD

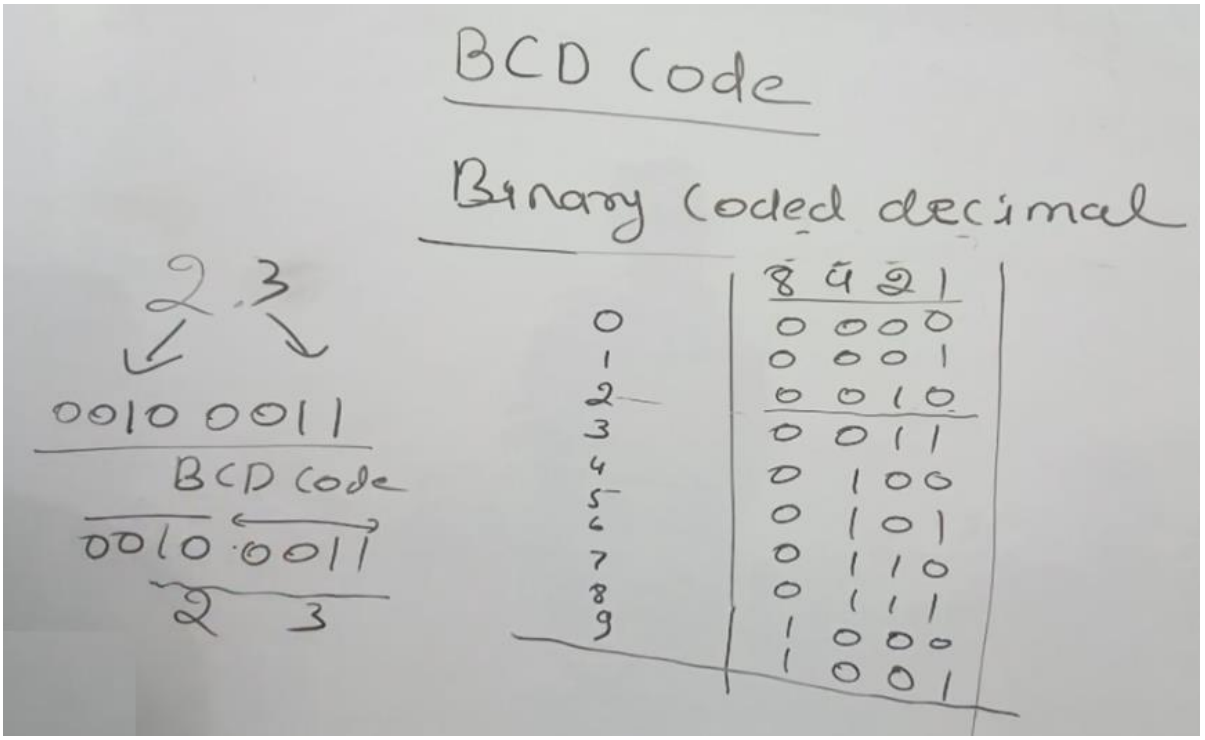
We convert each digit of the decimal number into groups of the four-bit binary number.

$(11110)_2 = (00110000)_{BCD}$

Decimal Number	Binary Number	Binary Coded Decimal (BCD)
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111

Decimal Number	Binary Number	Binary Coded Decimal (BCD)
8	1000	1000
9	1001	**1001**
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010
13	1101	0001 0011
14	1110	0001 0100
15	1111	0001 0101





Binary Coded Decimal, or **BCD**, is another process for converting decimal numbers into their binary equivalents. 8421 BCD code , with 8,4,2 and 1 representing the weights of different bits in the four-bit groups, Starting from MSB and proceeding towards LSB. This feature makes it a weighted code , which means that each bit in the four bit group representing a given decimal digit has an assigned weight.

In this code each decimal digit is represented by a 4-bit binary number. BCD is a way to express each of the decimal digits with a binary code. In the BCD, with four bits we can represent sixteen numbers (0000 to 1111). But in BCD code only first ten of these are used (0000 to 1001). The remaining six code combinations i.e. 1010 to 1111 are invalid in BCD.

Decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Binary Coded Decimal (BCD)

- Simplest binary code for decimal digits
- Only encodes ten digits from 0 to 9
- BCD is a weighted code
- The weights are 8,4,2,1
- Same weights as a binary number
- There are six invalid code words
- 1010, 1011, 1100, 1101, 1110, 1111
- Example on BCD coding:
- $13 \Leftrightarrow (0001\ 0011)_{BCD}$

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Warning: Conversion or Coding?

- Do NOT mix up conversion of a decimal number to a binary number with coding a decimal number with a binary code
- $13_{10} = (1101)_2$ This is conversion
- $13 \Leftrightarrow (0001\ 0011)_{BCD}$ This is coding
- In general, coding requires more bits than conversion
- A number with n decimal digits is coded with $4n$ bits in BCD

1. Convert (123)10 in BCD

From the truth table above,
1 -> 0001
2 -> 0010
3 -> 0011
thus, BCD becomes -> 0001 0010 0011

2. Convert (324)10 in BCD

(324)10 -> 0011 0010 0100 (BCD)

Again from the truth table above,
3 -> 0011
2 -> 0010
4 -> 0100
thus, BCD becomes -> 0011 0010 0100

The BCD_{8421} code is so called because each of the four bits is given a 'weighting' according to its column value in the binary system.

The least significant bit (lsb) has the weight or value 1, the next bit, going left, the value 2. The next bit has the value 4, and the most significant bit (msb) the value 8,

So the 8421_{BCD} code for the decimal number 6_{10} is 0110_{8421}

24_{10} in 8 bit binary would be 00011000 but in BCD_{8421} is 0010 0100.

992_{10} in 16 bit binary would be 0000001111100000_2 but in BCD_{8421} is 1001 1001 0010.

321_{10} to BCD_{8421}

65231_{10} to BCD_{8421}

001101110110 BCD_{8421} to decimal.

0011001011000110 BCD_{8421} to decimal.

1. Examples

1. $(93F)_{16}$ = (___?___)$_{BCD}$

Solution:
$(93F)_{16}$ = (_____)$_{BCD}$

1. Convert hexadecimal to decimal
$(93F)_{16}$ = (_____)$_{10}$

$93F$

$= 9 \times 16^2 + 3 \times 16^1 + F \times 16^0$

$= 9 \times 256 + 3 \times 16 + 15 \times 1$

$= 2367$

$\therefore (93F)_{16} = (2367)_{10}$

2. Convert decimal to BCD
$(2367)_{10}$ = (_____)$_{BCD}$

2 3 6 7
0010 0011 0110 0111

$\therefore (2367)_{10} = (0010001101100111)_{BCD}$

$\therefore (93F)_{16} = (0010001101100111)_{BCD}$

2. $(94D)_{16}$ = (___?___)$_{BCD}$

Solution:
$(94D)_{16}$ = (_____)$_{BCD}$

1. Convert hexadecimal to decimal
$(94D)_{16}$ = (_____)$_{10}$

$94D$

$= 9 \times 16^2 + 4 \times 16^1 + D \times 16^0$

$= 9 \times 256 + 4 \times 16 + 13 \times 1$

$= 2381$

$\therefore (94D)_{16} = (2381)_{10}$

2. Convert decimal to BCD
$(2381)_{10}$ = (_____)$_{BCD}$

2 3 8 1
0010 0011 1000 0001

$\therefore (2381)_{10} = (0010001110000001)_{BCD}$

$\therefore (94D)_{16} = (0010001110000001)_{BCD}$

1. $(1011001)_{BCD}$ = (___?___)$_{10}$

Solution:
$(1011001)_{BCD}$ = (_____)$_{10}$

0101 1001
5 9

∴ $(1011001)_{BCD} = (\underline{59})_{10}$

2. $(101100111)_{BCD}$ = (___?___)$_{10}$

Solution:
$(101100111)_{BCD}$ = (_____)$_{10}$

0001 0110 0111
1 6 7

∴ $(101100111)_{BCD} = (\underline{167})_{10}$

1. $(1011001)_{BCD}$ = (___?___)$_2$

Solution:
$(1011001)_{BCD}$ = (_____)$_2$

1. Convert BCD to decimal
$(1011001)_{BCD}$ = (_____)$_{10}$

0101 1001
5 9

∴ $(1011001)_{BCD} = (\underline{59})_{10}$

2. Convert decimal to binary
$(59)_{10}$ = (_____)$_2$

2	59	
2	29	1 ↑
2	14	1 ↑
2	7	0 ↑
2	3	1 ↑
2	1	1 ↑
	0	1 ↑

∴ $(59)_{10} = (\underline{111011})_2$

∴ **$(1011001)_{BCD} = (\underline{111011})_2$**

2. $(101100111)_{BCD} = (\_\_?\_\_)_2$

Solution:
$(101100111)_{BCD} = (\_\_\_\_\_)_2$

1. Convert BCD to decimal
$(101100111)_{BCD} = (\_\_\_\_\_)_{10}$

0001	0110	0111
1	6	7

$\therefore (101100111)_{BCD} = (\underline{167})_{10}$

2. Convert decimal to binary
$(167)_{10} = (\_\_\_\_\_)_2$

2	167	
2	83	1 ↑
2	41	1 ↑
2	20	1 ↑
2	10	0 ↑
2	5	0 ↑
2	2	1 ↑
2	1	0 ↑
	0	1 ↑

$\therefore (167)_{10} = (\underline{10100111})_2$

$\therefore (101100111)_{BCD} = (\underline{10100111})_2$

1. Examples

1. $(1011001)_{BCD} = (\_\_?\_\_)_8$

Solution:
$(1011001)_{BCD} = (\_\_\_\_\_)_8$

1. Convert BCD to decimal
$(1011001)_{BCD} = (\_\_\_\_\_)_{10}$

0101	1001
5	9

$\therefore (1011001)_{BCD} = (\underline{59})_{10}$

2. Convert decimal to octal
$(59)_{10} = (\_\_\_\_\_)_8$

8	59	
8	7	3 ↑
	0	7 ↑

$\therefore (59)_{10} = (\underline{73})_8$

$\therefore (1011001)_{BCD} = (\underline{73})_8$

2. $(101100111)_{BCD}$ = (___?___)$_8$

Solution:
$(101100111)_{BCD}$ = (_____)$_8$

1. Convert BCD to decimal
$(101100111)_{BCD}$ = (_____)$_{10}$

0001 0110 0111
1 6 7

∴ $(101100111)_{BCD} = (\underline{167})_{10}$

2. Convert decimal to octal
$(167)_{10}$ = (_____)$_8$

8	167	
8	20	7 ↑
8	2	4 ↑
	0	2 ↑

∴ $(167)_{10} = (\underline{247})_8$

∴ **$(101100111)_{BCD} = (\underline{247})_8$**

1. Examples

1. $(1011001)_{BCD}$ = (___?___)$_{16}$

Solution:
$(1011001)_{BCD}$ = (_____)$_{16}$

1. Convert BCD to decimal
$(1011001)_{BCD}$ = (_____)$_{10}$

0101 1001
5 9

∴ $(1011001)_{BCD} = (\underline{59})_{10}$

2. Convert decimal to hexadecimal
$(59)_{10}$ = (_____)$_{16}$

16	59	
16	3	B ↑
	0	3 ↑

∴ $(59)_{10} = (\underline{3B})_{16}$

∴ **$(1011001)_{BCD} = (\underline{3B})_{16}$**

2. $(101100111)_{BCD} = (\_\_?\_\_)_{16}$

Solution:
$(101100111)_{BCD} = (\_\_\_\_\_)_{16}$

1. Convert BCD to decimal
$(101100111)_{BCD} = (\_\_\_\_\_)_{10}$

0001 0110 0111
1 6 7

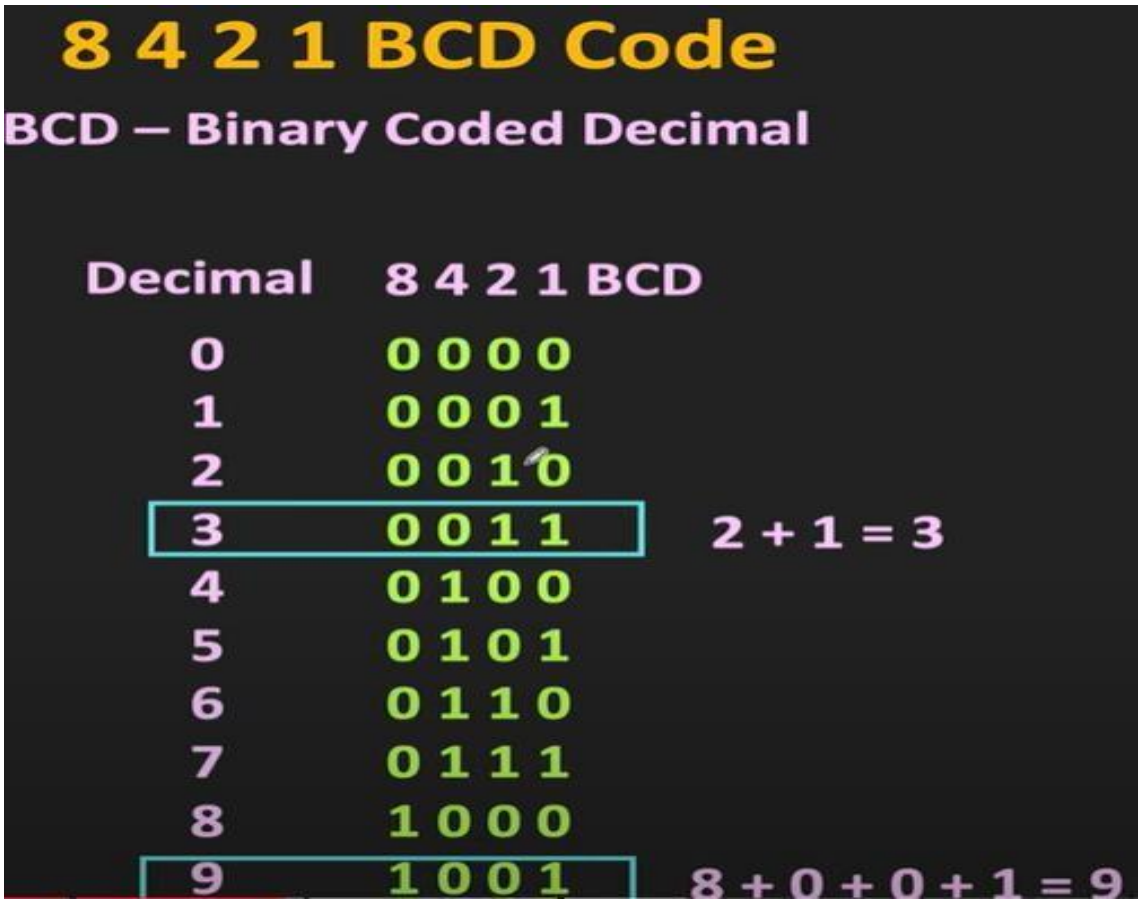
$\therefore (101100111)_{BCD} = (167)_{10}$

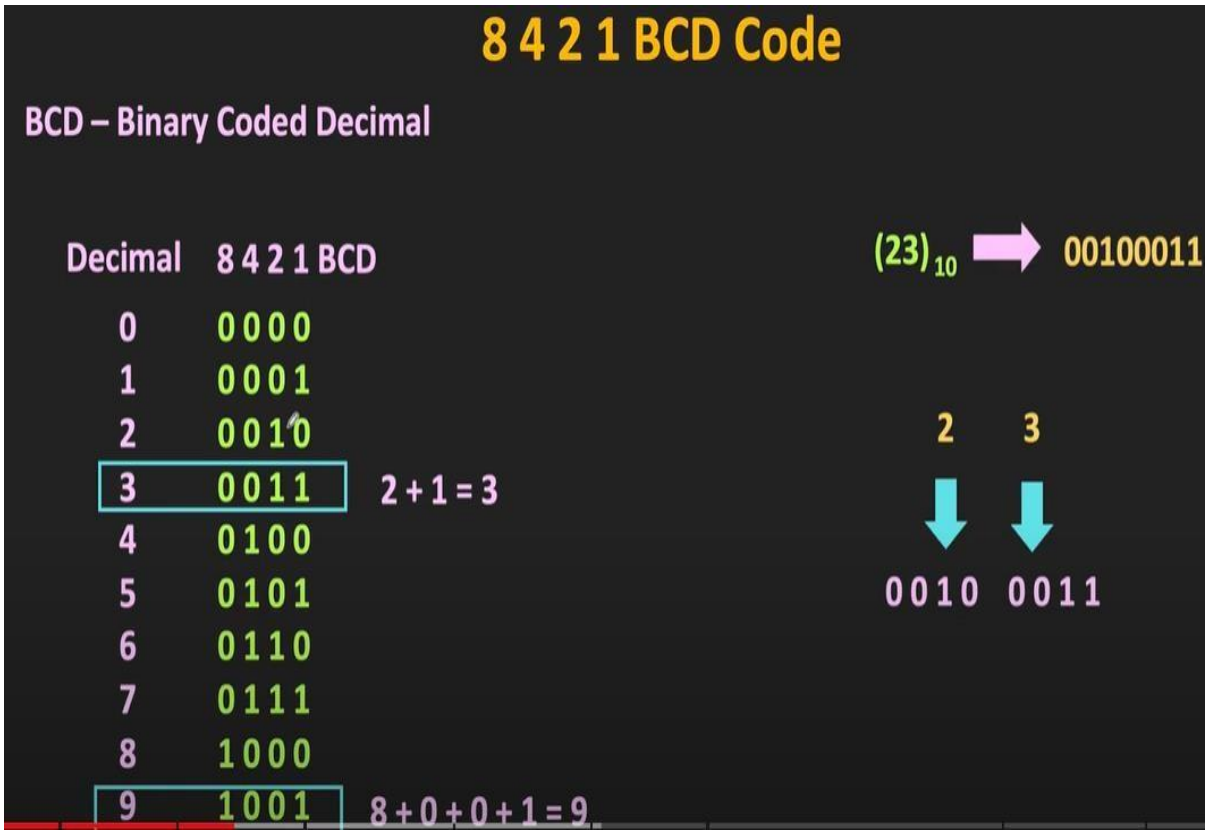
2. Convert decimal to hexadecimal
$(167)_{10} = (\_\_\_\_\_)_{16}$

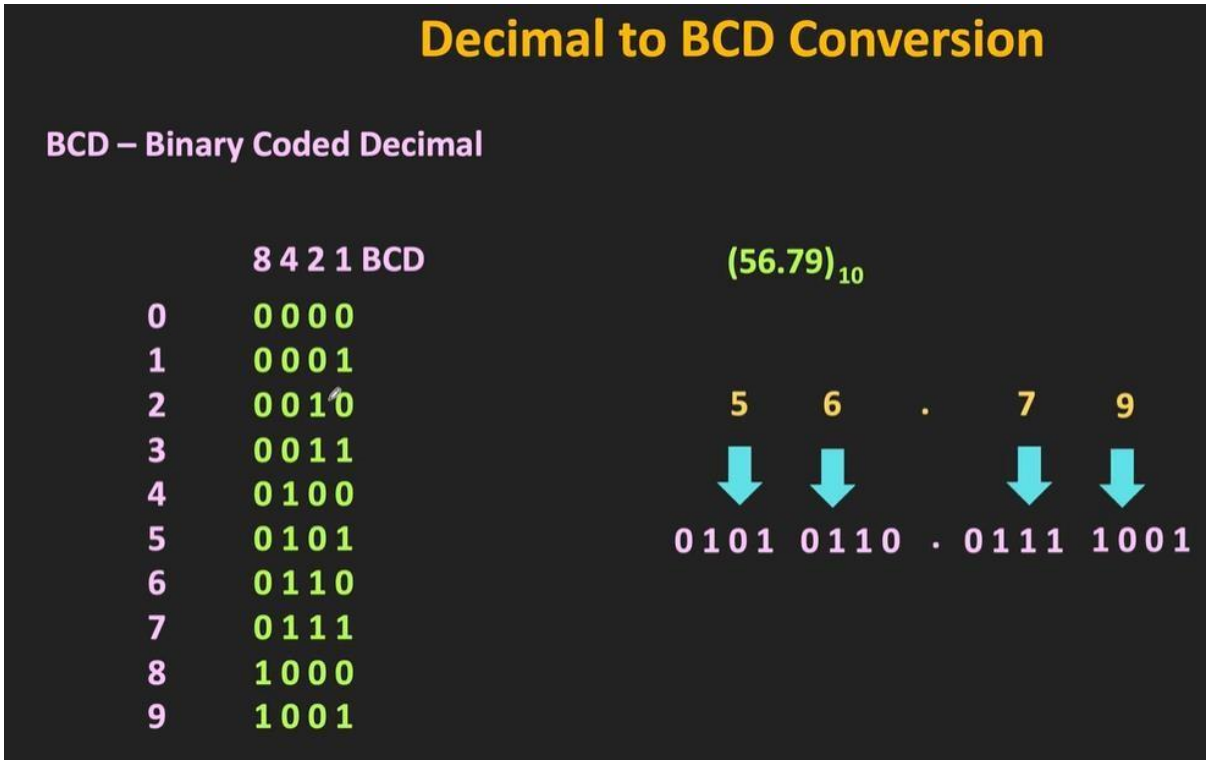
16	167	
16	10	7 ↑
	0	A ↑

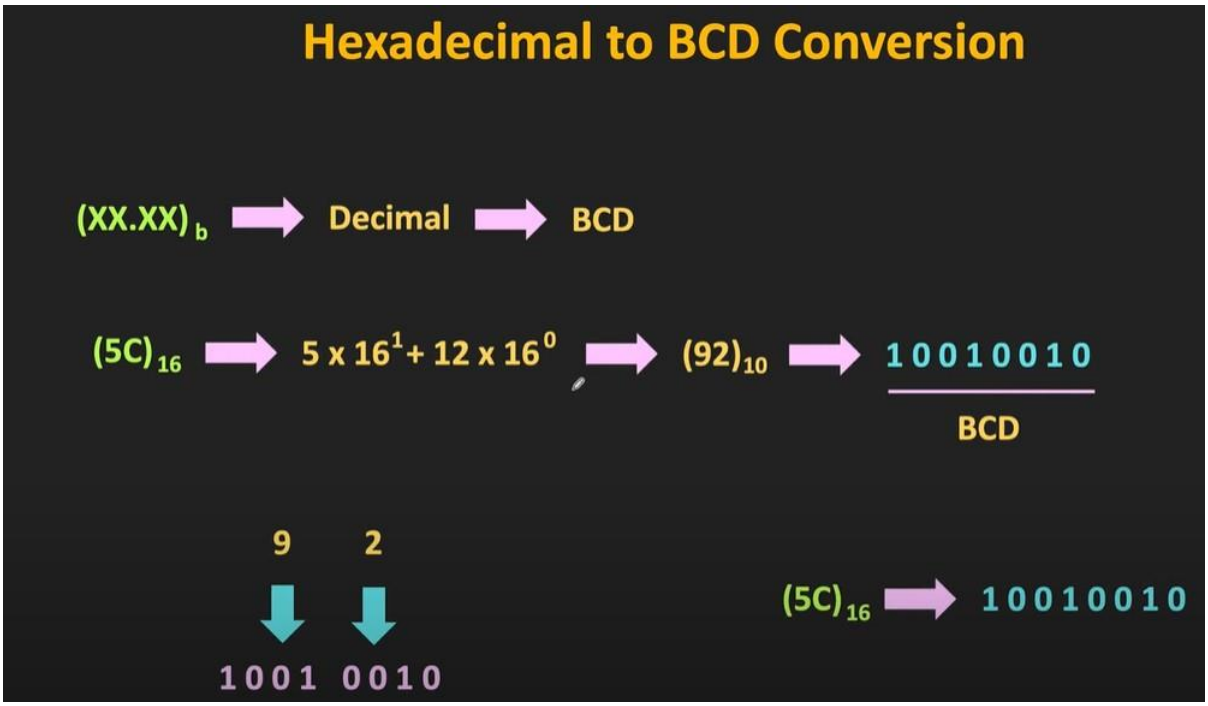
$\therefore (167)_{10} = (A7)_{16}$

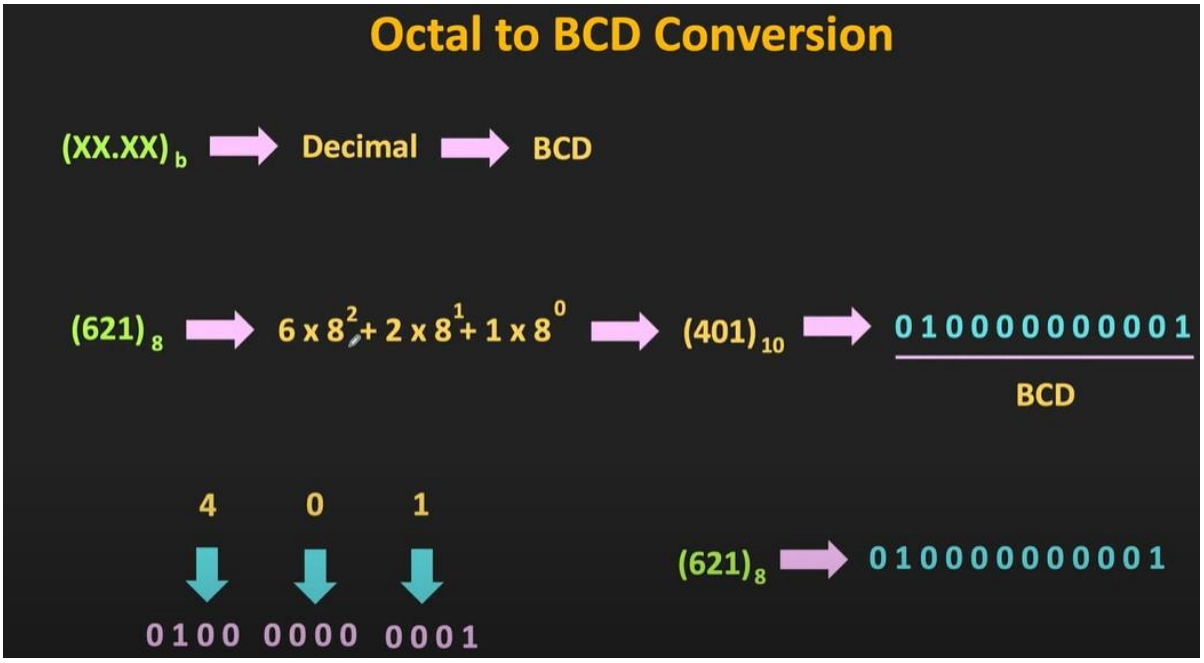
$\therefore (101100111)_{BCD} = (A7)_{16}$

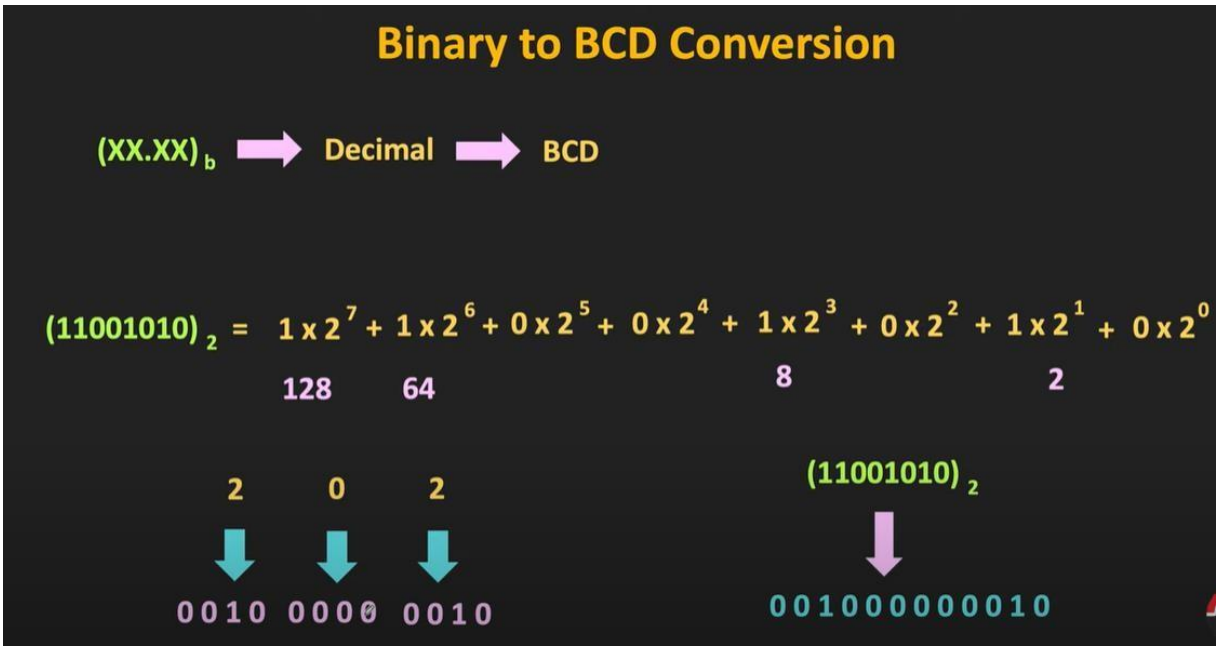



8 4 2 1 BCD Code
BCD – Binary Coded Decimal
Decimal 8 4 2 1 BCD
0 0000
1 0001
2 0010
3 0011
2+1=3
4 0100
5 0101
6 0110
7 0111
8 1000
9 1001
8+0+0+1=9
(23)10
00100011
2 3
0010 0011


Decimal to BCD Conversion
BCD – Binary Coded Decimal
8 4 2 1 BCD
0 0000
1 0001
2 0010
3 0011
4 0100
5 0101
6 0110
7 0111
8 1000
9 1001
(56.79)10
5 6 . 7 9
0101 0110 . 0111 1001


Hexadecimal to BCD Conversion
(XX.XX)b → Decimal → BCD
(5C)16 → 5 x 16^1 + 12 x 16^0 → (92)10 → 10010010
BCD
9 2
1001 0010
(5C)16 → 10010010


Octal to BCD Conversion
(XX.XX)b → Decimal → BCD
(621)8 → 6 x 8^2 + 2 x 8^1 + 1 x 8^0 → (401)10 → 010000000001
BCD
4 0 1
0100 0000 0001
(621)8 → 010000000001



Binary to Gray Code Conversion

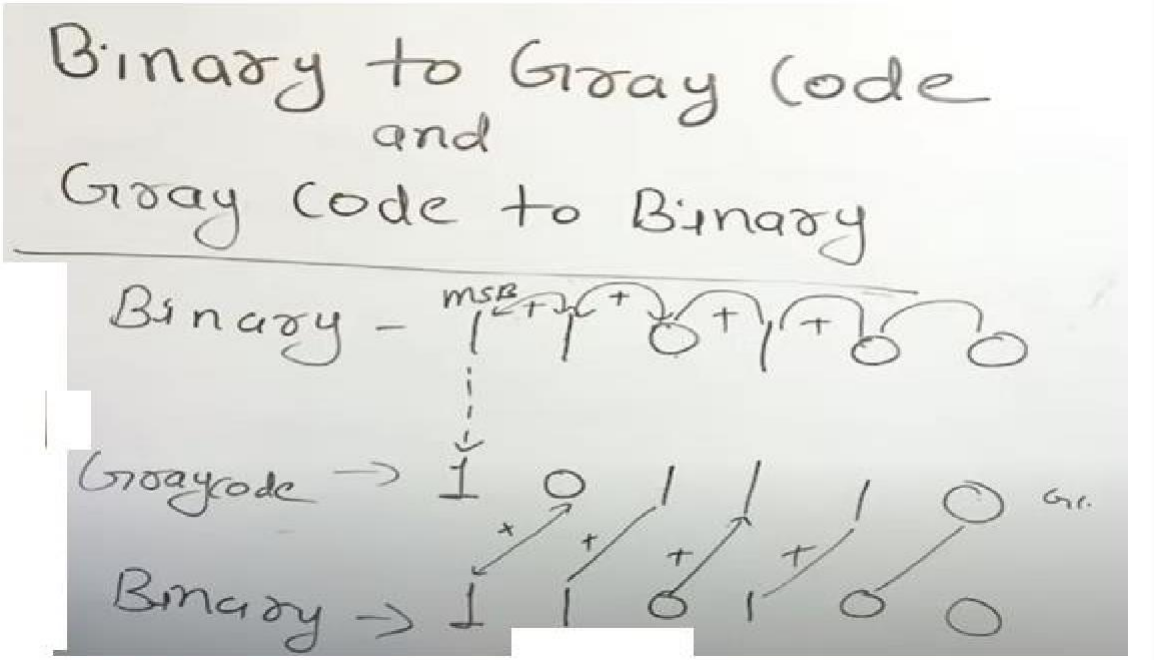
How to Convert Binary to Gray Code

- In the Gray code, the MSB will always be the same as the 1'st bit of the given binary number.
- In order to perform the 2nd bit of the gray code, we perform the exclusive-or (XOR) of the 1'st and 2nd bit of the binary number. It means that if both the bits are different, the result will be one else the result will be 0.
- In order to get the 3rd bit of the gray code, we need to perform the exclusive-or (XOR) of the 2nd and 3rd bit of the binary number. The process remains the same for the 4th bit of the Gray code. Let's take an example to understand these steps.
- The gray code of the binary number 01101 is 01011.

Binary *- Gray Code converter***,** truth table & example conversion to perform binary to gray code or gray code to binary conversion in digital electronics & communications. Select the radio button to perform the appropriate conversion. Both the conversions can be done by using the below EX-OR gate logic.

EX-OR Gate Truth Table

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0



Solved Example :

The below solved example may useful to understand how to perform binary to gray code conversion.

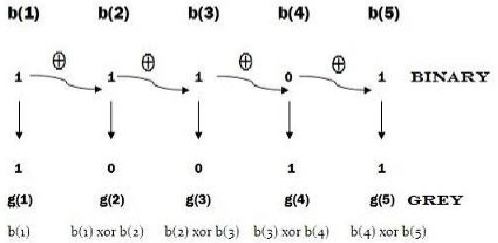
Problem

Find the equivalent gray code for the binary 11101_2.

Solution :

Binary to Grey Code Conversion

Convert the binary 11101_2 to its equivalent Grey code



Solved Example :

The below solved example may useful to understand how to perform gray code to binary convers

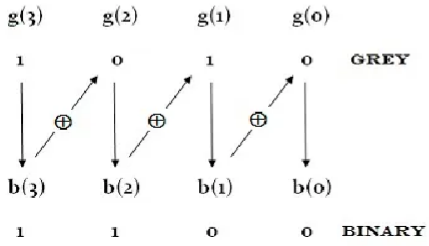
Problem

Find the equivalent binary number for the gray code 1010.

Solution :

Grey Code to Binary Conversion

Convert the Grey code 1010 to its equivalent Binary



---

1. Examples

1. $(A1)_{16}$ = (___?___)Gray Code

Solution:
$(A1)_{16}$ = (_____)GrayCode

1. Convert hexadecimal to binary
$(A1)_{16}$ = (_____)$_2$

A 1
1010 0001

$\therefore (A1)_{16} = (10100001)_2$

2. Convert binary to GrayCode
Binary code : 10100001

Method-1: (Binary to Gray code)

$g_7 = b_7 = 1$

$g_6 = b_7 \oplus b_6 = 1 \oplus 0 = 1$

$g_5 = b_6 \oplus b_5 = 0 \oplus 1 = 1$

$g_4 = b_5 \oplus b_4 = 1 \oplus 0 = 1$

$g_3 = b_4 \oplus b_3 = 0 \oplus 0 = 0$

$g_2 = b_3 \oplus b_2 = 0 \oplus 0 = 0$

$g_1 = b_2 \oplus b_1 = 0 \oplus 0 = 0$

$g_0 = b_1 \oplus b_0 = 0 \oplus 1 = 1$

$\therefore$ **Gray code : 11110001**

Method-2: (Binary to Gray code)

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	**Binary code**
1	**0**	**1**	**0**	**0**	**0**	**0**	**1**	
b_7	$b_7 \oplus b_6$	$b_6 \oplus b_5$	$b_5 \oplus b_4$	$b_4 \oplus b_3$	$b_3 \oplus b_2$	$b_2 \oplus b_1$	$b_1 \oplus b_0$	
1	$1 \oplus 0$	$0 \oplus 1$	$1 \oplus 0$	$0 \oplus 0$	$0 \oplus 0$	$0 \oplus 0$	$0 \oplus 1$	
↓	↓	↓	↓	↓	↓	↓	↓	
1	**1**	**1**	**1**	**0**	**0**	**0**	**1**	**Gray code**
g_7	g_6	g_5	g_4	g_3	g_2	g_1	g_0	

∴ **Gray code : 11110001**

∴ **$(A1)_{16} = (\underline{11110001})_{GrayCode}$**

2. $(123)_{16} = (\_\_?\_\_)_{Gray\ Code}$

Solution:
$(123)_{16} = (\_\_\_\_\_)_{GrayCode}$

1. Convert hexadecimal to binary
$(123)_{16} = (\_\_\_\_\_)_2$

1 2 3
0001 0010 0011

∴ $(123)_{16} = (\underline{100100011})_2$

2. Convert binary to GrayCode
Binary code : 100100011

Method-1: (Binary to Gray code)

$g_8 = b_8 = 1$

$g_7 = b_8 \oplus b_7 = 1 \oplus 0 = 1$

$g_6 = b_7 \oplus b_6 = 0 \oplus 0 = 0$

$g_5 = b_6 \oplus b_5 = 0 \oplus 1 = 1$

$g_4 = b_5 \oplus b_4 = 1 \oplus 0 = 1$

$g_3 = b_4 \oplus b_3 = 0 \oplus 0 = 0$

$g_2 = b_3 \oplus b_2 = 0 \oplus 0 = 0$

$g_1 = b_2 \oplus b_1 = 0 \oplus 1 = 1$

$g_0 = b_1 \oplus b_0 = 1 \oplus 1 = 0$

∴ **Gray code : 110110010**

Method-2: (Binary to Gray code)

b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	**Binary code**
1	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	
b_8	$b_8 \oplus b_7$	$b_7 \oplus b_6$	$b_6 \oplus b_5$	$b_5 \oplus b_4$	$b_4 \oplus b_3$	$b_3 \oplus b_2$	$b_2 \oplus b_1$	$b_1 \oplus b_0$	
1	$1 \oplus 0$	$0 \oplus 0$	$0 \oplus 1$	$1 \oplus 0$	$0 \oplus 0$	$0 \oplus 0$	$0 \oplus 1$	$1 \oplus 1$	
↓	↓	↓	↓	↓	↓	↓	↓	↓	
1	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**Gray code**
g_8	g_7	g_6	g_5	g_4	g_3	g_2	g_1	g_0	

∴ **Gray code : 110110010**

∴ $(123)_{16} = (\underline{110110010})_{GrayCode}$

Logic Gates

Basic logic gates

AND, OR, XOR, NOT, NAND, NOR, and XNOR.

AND | OR | XOR | NOT | NAND | NOR | XNOR

A logic gate is a basic building block of a digital circuit that has two inputs and one output. The relationship between the i/p and the o/p is based on a certain logic. These gates are implemented using electronic switches like transistors, diodes. But, in practice, basic logic gates are built using CMOS technology

It is a building block of a digital system and an electronic circuit that always have only one output. These gates can have one input or more than one input, but most of the gates have two inputs. On the basis of the relationship between the input and the output, these gates are named as AND gate, OR gate, NOT gate, etc

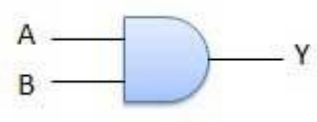
AND Gate

A circuit which performs an AND operation is shown in figure. It has n input (n >= 2) and one output.

Y = A AND B AND C ..... N
Y = A.B.C ..... N
Y = ABC ..... N

Inputs		Output
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

Logic diagram **Truth Table**



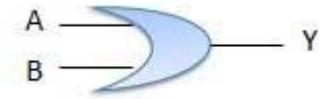
Inputs		Output
A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate **Truth Table**

A circuit which performs an OR operation is shown in figure. It has n input (n >= 2) and one output.

Y = A OR B OR C ..... N
Y = A + B + C ..... N

Logic diagram

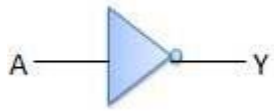


NOT Gate

NOT gate is also known as **Inverter**. It has one input A and one output Y.

Y = NOT A
Y = $\overline{A}$

Logic diagram **Truth Table**



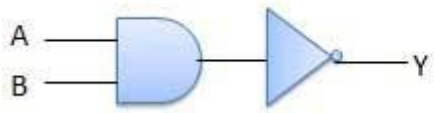
Inputs	Output
A	B
0	1
1	0

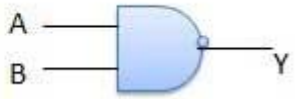
NAND Gate

A NOT-AND operation is known as NAND operation. It has n input (n >= 2) and one output.

Y = A NOT AND B NOT AND C ..... N
Y = A NAND B NAND C ..... N

Logic diagram





Truth Table

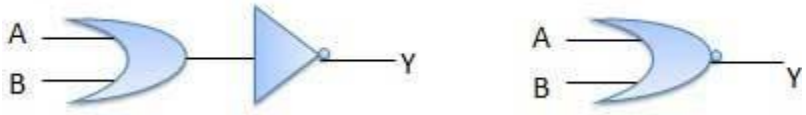
Inputs		Output
A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR Gate

A NOT-OR operation is known as NOR operation. It has n input (n >= 2) and one output.

Y = A NOT OR B NOT OR C ..... N
Y = A NOR B NOR C ..... N

Logic diagram



Truth Table

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

XOR Gate

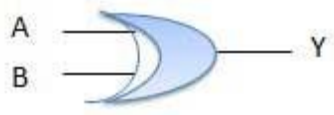
XOR or Ex-OR gate is a special type of gate. It can be used in the half adder, full adder and subtractor. The exclusive-OR gate is abbreviated as EX-OR gate or sometime as X-OR gate. It has n input (n >= 2) and one output.

Y = A XOR B XOR C ..... N

Y = $A \oplus B \oplus C$ ..... N

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

Logic diagram



Truth Table

Inputs		Output
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

XNOR Gate

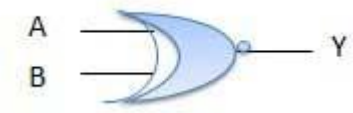
XNOR gate is a special type of gate. It can be used in the half adder, full adder and subtractor. The exclusive-NOR gate is abbreviated as EX-NOR gate or sometime as X-NOR gate. It has n input (n >= 2) and one output.

Y = A XOR B XOR C ..... N

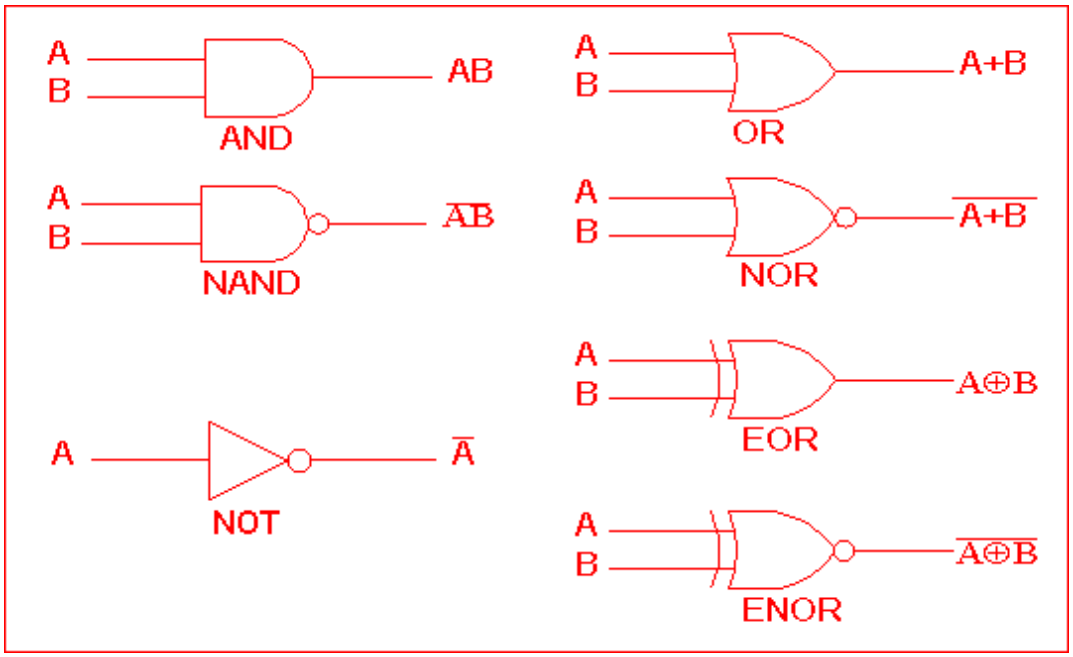
Y = A ⊙ B ⊙ C ..... N

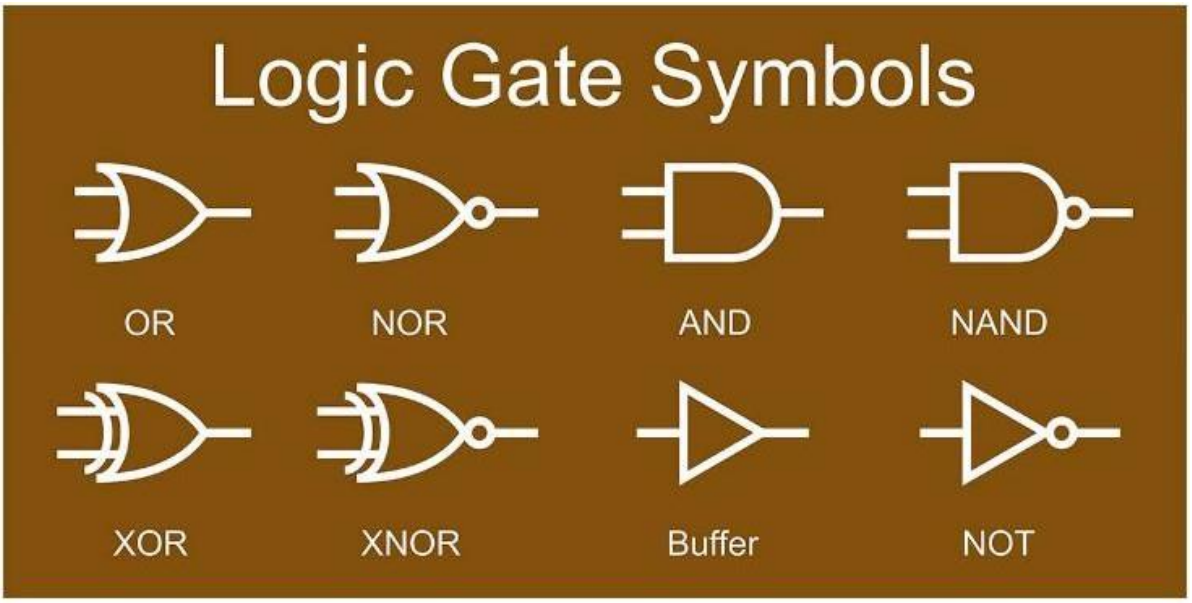
$Y = \overline{A}B + A\overline{B}$

Logic diagram **Truth Table**



Inputs		Output
A	B	A ⊙ B
0	0	1
0	1	0
1	0	0
1	1	1

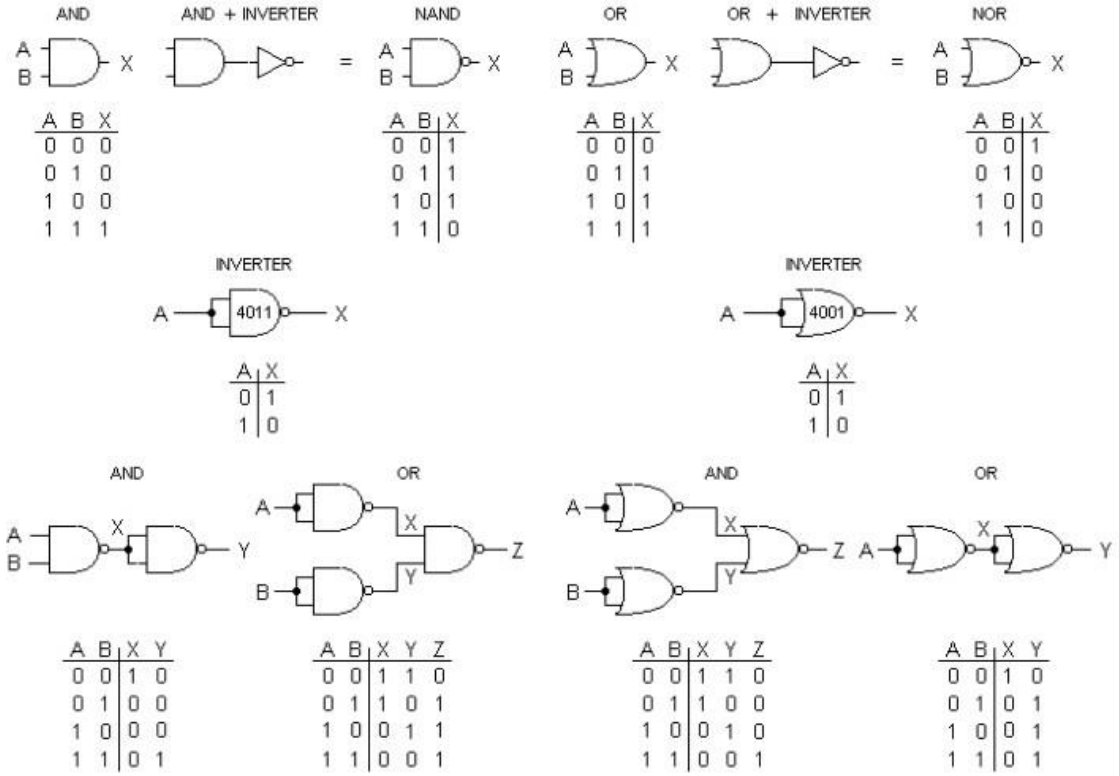




Expressed algebraically, the Boolean equations are:

Expression	Proposition	Statement
$x \cdot y$	x AND y	true
$x + y$	x OR y, x AND y	true
$\bar{x} \cdot \bar{y}$	NOT x AND NOT y	true
$\bar{x} + \bar{y}$	NOT x OR NOT y,	
	NOT x AND NOT y	true

summary

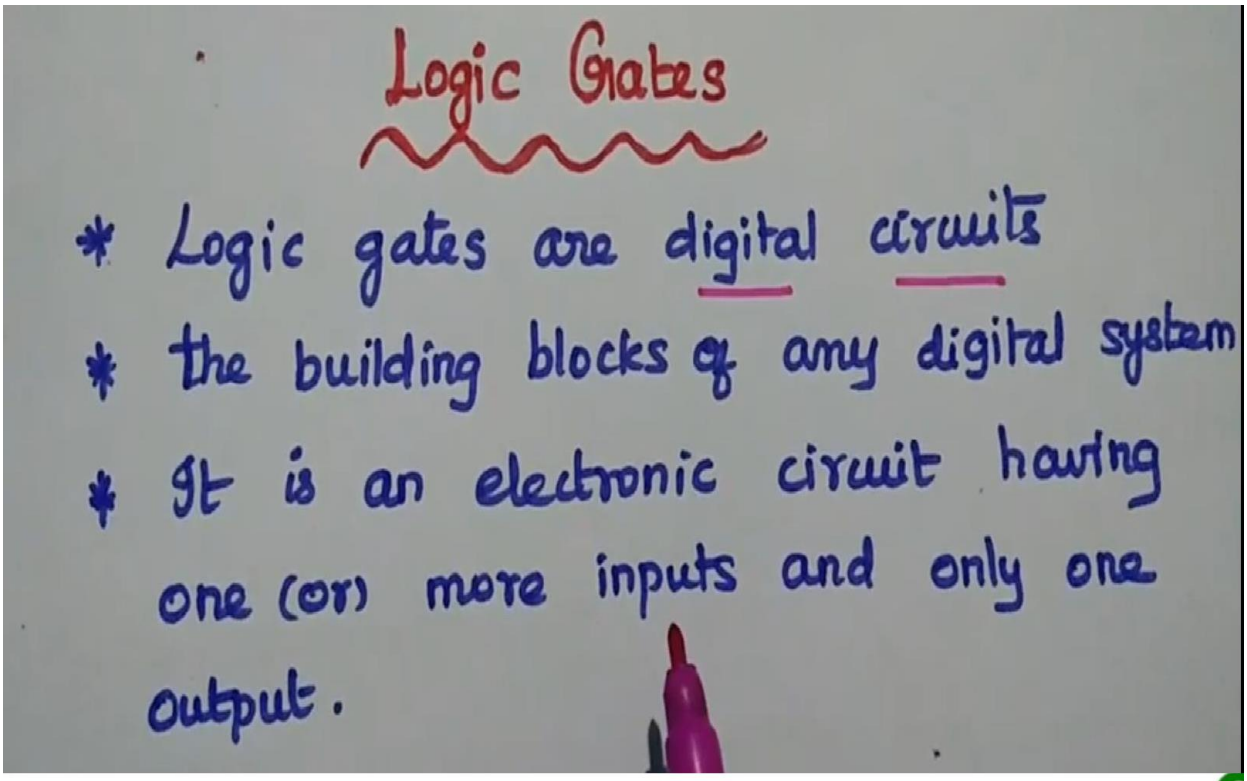


Ans → Logic gate is a electronic circuit which can take one or more inputs but output will be one.

Note :-

⊙ Logic gates are made up of diodes and transistors.

⊙ A logic gate is used to allow and denied a digital signal.

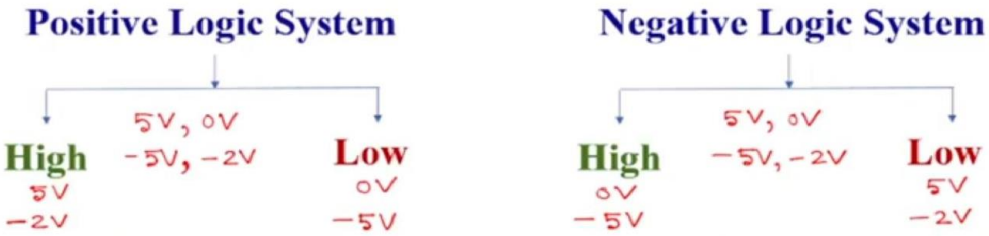


~: Logic Gates :~

An electronic circuit which makes logical decisions.

Two logics or logical values in logic gates or digital circuits: 1 and 0.

Two logic systems: Positive logic and Negative logic.



They have 2 or more inputs and 1 output except NOT gate (only 1 input).

Basic Gates: OR, AND, NOT

Use - Digital watch ✓

Computers ✓

Calculator ✓ 1.10

Types :-

(1) AND Gate.
(2) OR Gate.
(3) NOT Gate.
(4) NAND Gate.
(5) XOR Gate.
(6) XNOR Gate.
(7) NOR Gate.

Types :-

* **Basic Gates :-**
  - (i) AND Gate
  - (ii) OR Gate
  - (iii) NOT Gate
* **Universal Gates :-**
  - (iv) NAND Gate
  - (v) NOR Gate
* **Special Gates :-**
  - (vi) EXOR Gate
  - (vii) EXNOR Gate

AND Gate

→ An AND Gate has two or more inputs and only one output which is equal to the product of all inputs

Truth Table :-

Inputs		Output
A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

IC 7408.

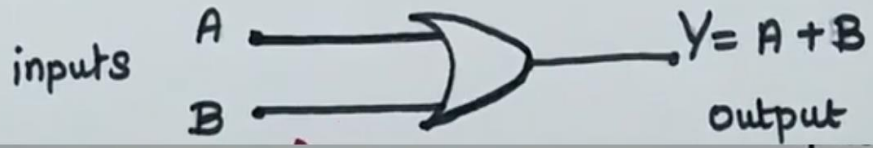
↳ 4 AND Gates are available.

→ If one of the inputs is low, then the output is also low.

→ When both the inputs are high, then the output is also high.

"OR" Gate

* It has two or more inputs and only one output.
* The output is equal to the sum of all inputs



Truth Table :-

Inputs		Output
A	B	Y= A + B
0	0	0
0	1	1
1	0	1
1	1	1

→ If one of the inputs is high, then the output is also high.

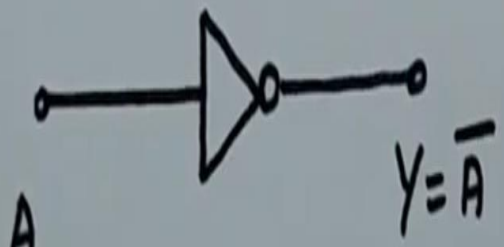
→ When both the inputs are low, the output is also low.

GATEs are available in chips.

Ex: IC 7432

"NOT" Gate (or) "Inverter" Gate

* Single input and Single output.
* Output is the complement of the input logic.

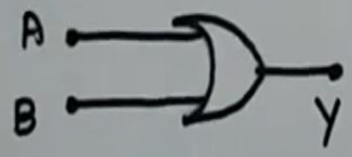


Input (A)	O/P ($\bar{A}$)
0	1
1	0

IC 7404 → NOT GATE IC. (6 NOT Gates).

OR GATE

Sum of the inputs

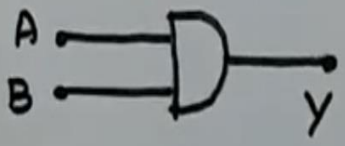


$Y = A + B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

AND GATE

Product of the inputs

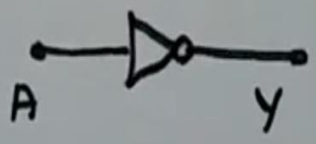


$Y = A \cdot B$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

NOT GATE

Complement of the input



$Y = \overline{A}$

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

Universal GATES

[NOR & NAND Gates].

* It is a gate which can be used to implement any other gates without using other types of gates.

Ex: NOR GATE
NAND GATE.

Universal GATES

[NOR & NAND Gates]

* It is a gate which can be used to implement any other gates without using other types of gates.

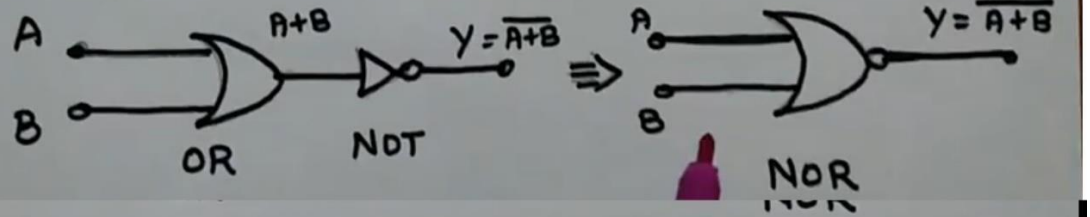
Ex: NOR GATE
NAND GATE.

* NOR GATE :-

→ NOT - OR

→ It has two or) more inputs but only one output.

→ The output is the complement of the sum of the inputs.



Truth Table :-

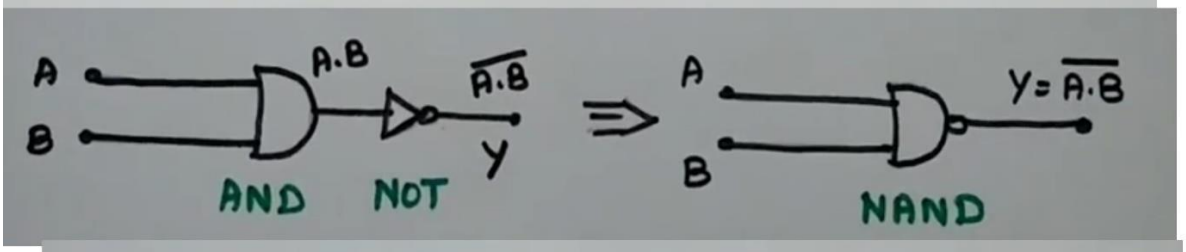
Inputs A	Inputs B	A+B	$Y = \overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

NAND GATE:-

→ NOT - AND

→ It has two or more inputs, but only one output

→ The output is the complement of the product of inputs.



Truth Table:-

Inputs		AND	NAND
A	B	A.B	$Y = \overline{A.B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

→ If any of the inputs is low, the output is high.

→ If all the inputs are high, then only the output is low.

Special Gates :-

(i) XOR Gate

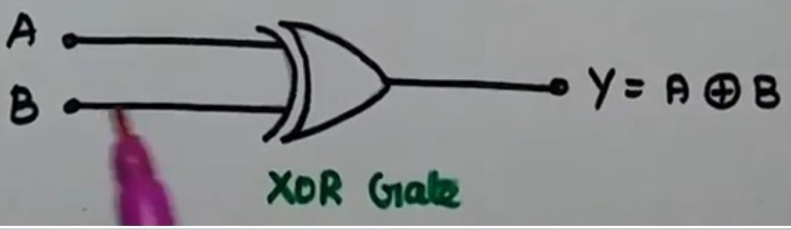
(ii) XNOR Gate.

XOR Gate :-

→ Exclusive - OR Gate [EX-OR]

→ It has two or more inputs & only one output.

$$Y = A \oplus B = A\bar{B} + \bar{A}B$$



Truth Table :-

Inputs		Output
A	B	Y = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

→ If any one of the inputs is low, then the output is high.

→ It can be used in Half adder, full adder & subtractor circuits.

~: XOR Gate :~

$Y = A \oplus B$

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

$0, 0 \Rightarrow 0\cdot 1 + 0\cdot 1 = 0 + 0 = 0$

$0, 1 \Rightarrow 0\cdot 0 + 1\cdot 1 = 0 + 1 = 1$

$1, 0 \Rightarrow 1\cdot 1 + 0\cdot 0 = 1 + 0 = 1$

$1, 1 \Rightarrow 1\cdot 0 + 0\cdot 1 = 0 + 0 = 0$

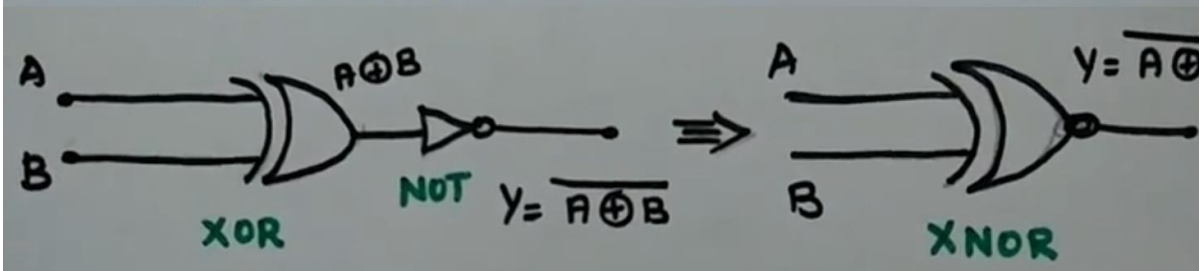
XOR Gate		
Truth Table		
A	**B**	**Y=AB' + BA'**
0	0	0
0	1	1
1	0	1
1	1	0

* XNOR GATE :-

→ Exclusive – NOR Gate [EX-NOR]

→ It has two (or) more inputs and only one output.

→ The output is the complement of the XOR Gate.

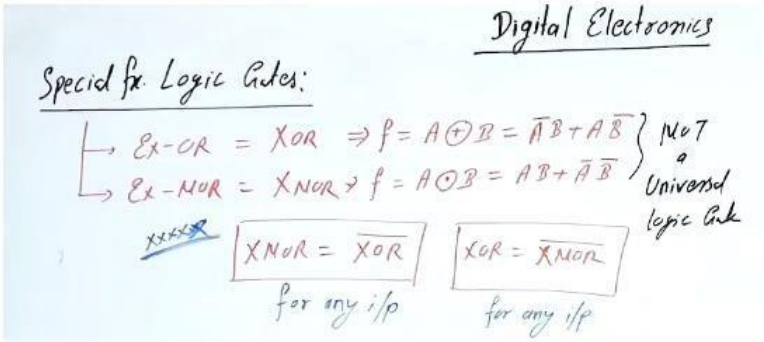


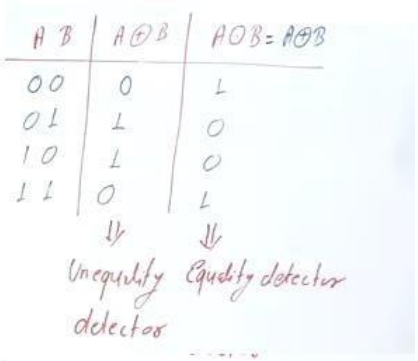
$$Y = \overline{A \oplus B} = AB + \bar{A}\bar{B}$$

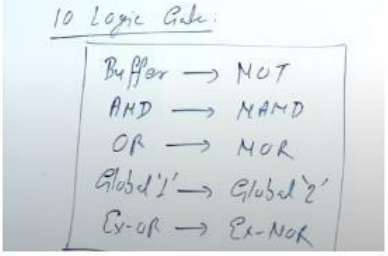
$$Y = A \odot B$$

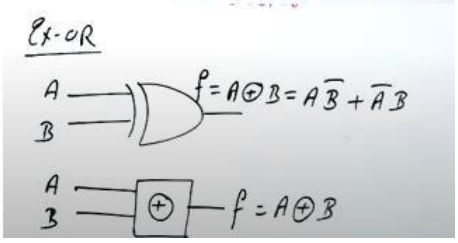
Truth Table :-

Inputs		XOR	XNOR
A	B	$A \oplus B$	$Y = \overline{A \oplus B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

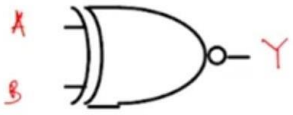








~: XNOR Gate :~



$$Y = \overline{A \oplus B}$$

0, 0 ⇒ XOR 0 ⇒ NOT $\overline{0} = 1$

0, 1 ⇒ 1 ⇒ $\overline{1} = 0$

1, 0 ⇒ 1 ⇒ $\overline{1} = 0$

1, 1 ⇒ 0

XNOR Gate		
Truth Table		
A	**B**	**Y=AB + A'B'**
0	0	1
0	1	0
1	0	0
1	1	1

_____ is also known as Inverter.

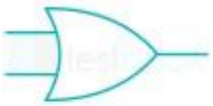
1. NOT Gate
2. OR Gate
3. NAND Gate
4. AND Gate

The symbol for two - input OR- gate in negative logic is:

1. 
2. 
3. 
4. 

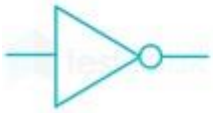
Symbols of various logic gates:

1.) OR Gate:



2.) AND Gate:

3.) NOT Gate:



4.) NOR Gate:

LOGIC GATE	OUTPUT
OR	A + B
AND	A.B
NAND	$\overline{A.B}$
NOR	$\overline{A+B}$
XOR	$A\overline{B}+\overline{A}B$
XNOR	$A.B+\overline{A.B}$

Symbols of various logic gates:

1.) OR Gate:

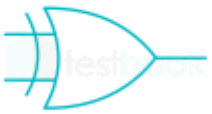
4.) NOR Gate:

2.) AND Gate:

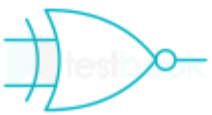
5.) NAND Gate:

3.) NOT Gate:

6.) XOR Gate:



7.) XNOR Gate:



DRAW the circuits

$$X = \bar{A}BC$$

$$X = \left(\overline{AB + \bar{C}}\right) \cdot B \cdot C$$

$$X = \left(\overline{AB}\right)\left(\bar{\bar{C}}\right) \cdot BC$$

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

AB + ĀC + BC = AB+ĀC
(A+B)(Ā+C)(B+C) = (A+B)(Ā+C)

1's complement

the binary number, but 1's and 2's complements are mostly used for binary numbers. We can find the 1's complement of the binary number by simply inverting the given number. For example, 1's complement of binary number 1011001 is 0100110. We can find the 2's complement of the binary number by changing each bit(0 to 1 and 1 to 0) and adding 1 to the least significant bit. For example, 2's complement of binary number 1011001 is (0100110)+1=0100111.

For finding 1's complement of the binary number, we can implement the logic circuit also by using NOT gate. We use NOT gate for each bit of the binary number. So, if we want to implement the logic circuit for 5-bit 1's complement, five NOT gates will be used.

Example 1: 11010.1101

For finding 1's complement of the given number, change all 0's to 1 and all 1's to 0. So the 1's complement of the number 11010.1101 comes out **00101.0010**.

Example 2: 100110.1001

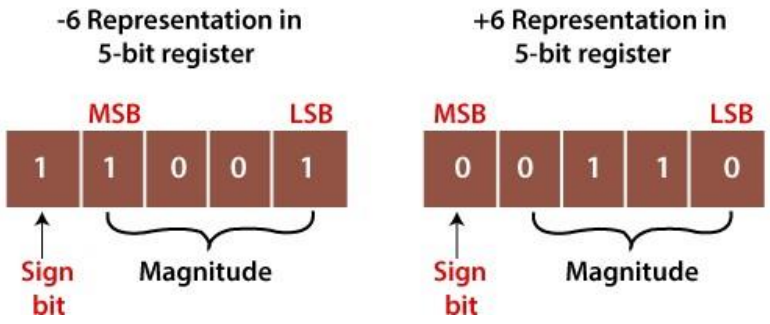
For finding 1's complement of the given number, change all 0's to 1 and all 1's to 0. So, the 1's complement of the number 100110.1001 comes out **011001.0110**.

1's Complement Table

Binary Number	1's Complement
0000	1111
0001	1110
0010	1101
0011	1100
0100	1011

The -6 is represented in the 5-bit register in the following way:

1. +6=0 0110
2. Find the 1's complement of the number 0 0110, i.e., 1 1001. Here, MSB denotes that a number is a negative number.



2's complement

1's complement, 2's complement is also used to represent the signed binary numbers. For finding 2's complement of the binary number, we will first find the 1's complement of the binary number and then add 1 to the least significant bit of it.

For example, if we want to calculate the 2's complement of the number 1011001, then firstly, we find the 1's complement of the number that is 0100110 and add 1 to the LSB. So, by adding 1 to the LSB, the number will be (0100110)+1=0100111. We can also create the logic circuit using OR, AND, and NOT gates. The logic circuit for finding 2's complement of the 5-bit binary number is as follows:

Example 1: 110100

For finding 2's complement of the given number, change all 0's to 1 and all 1's to 0. So the 1's complement of the number 110100 is 001011. Now add 1 to the LSB of this number, i.e., (001011)+1=001100.

Example 2: 100110

For finding 1's complement of the given number, change all 0's to 1 and all 1's to 0. So, the 1's complement of the number 100110 is 011001. Now add one the LSB of this number, i.e., (011001)+1=011010.

2's Complement Table

Binary Number	1's Complement	2's complement
0000	1111	0000
0001	1110	1111
0010	1101	1110
0011	1100	1101
0100	1011	1100
0101	1010	1011
0110	1001	1010

Compliment

1's Compliment
Get binary code for +X
flip every bit of step 1
Output = -X

2's Compliment
Get binary code for +X
flip every bit of step 1
Add 1 to setp 2

1. Examples

Method : 2's complement of a number is 1 added to it's 1's complement number.

---

1. Find 2's complement of 110

Note : 2's complement of a number is 1 added to it's 1's complement number.
1's complement of 110 is

```
    1  1  1
 -  1  1  0
 ------------
    0  0  1
```

Now add 1 : 001 + 1 = 010

2. Find 2's complement of 10110

Note : 2's complement of a number is 1 added to it's 1's complement number.
1's complement of 10110 is

```
    1  1  1  1  1
 -  1  0  1  1  0
 ------------------
    0  1  0  0  1
```

Now add 1 : 01001 + 1 = 01010

9's and 10's Complement

If the number is binary, then we use 1's complement and 2's complement. But in case, when the number is a decimal number, we will use the 9's and 10's complement. The 10's complement is obtained from the 9's complement of the number, and we can also find the 9's and 10's complement using the r's and (r-1)'s complement formula.

9's Complement

The 9's complement is used to find the subtraction of the decimal numbers. The 9's complement of a number is calculated by subtracting each digit of the number by 9. For example, suppose we have a number 1423, and we want to find the 9's complement of the number. For this, we subtract each digit of the number 1423 by 9. So, the 9's complement of the number 1423 is 9999-1423= 8576.

10's Complement

The 10's complement is also used to find the subtraction of the decimal numbers. The 10's complement of a number is calculated by subtracting each digit by 9 and then adding 1 to the result. Simply, by adding 1 to its 9's complement we can get its 10's complement value. For example, suppose we have a number 1423, and we want to find the 10's complement of the number. For this, we find the 9's complement of the number 1423 that is 9999-1423= 8576, and now we will add 1 to the

result. So the 10's complement of the number 1423 is 8576+1=8577.

decimal number 456, 9’s complement of this number will be

```
  999
(-) 456
-------
  543
```

10’s complement of this no

```
  543
(+)1
-------
  544
```

- 10’s Complement
- (10-1)’s Complement / 9’s Complement
- 9’s Complement – It is obtained by subtracting each digit of the number from 9.

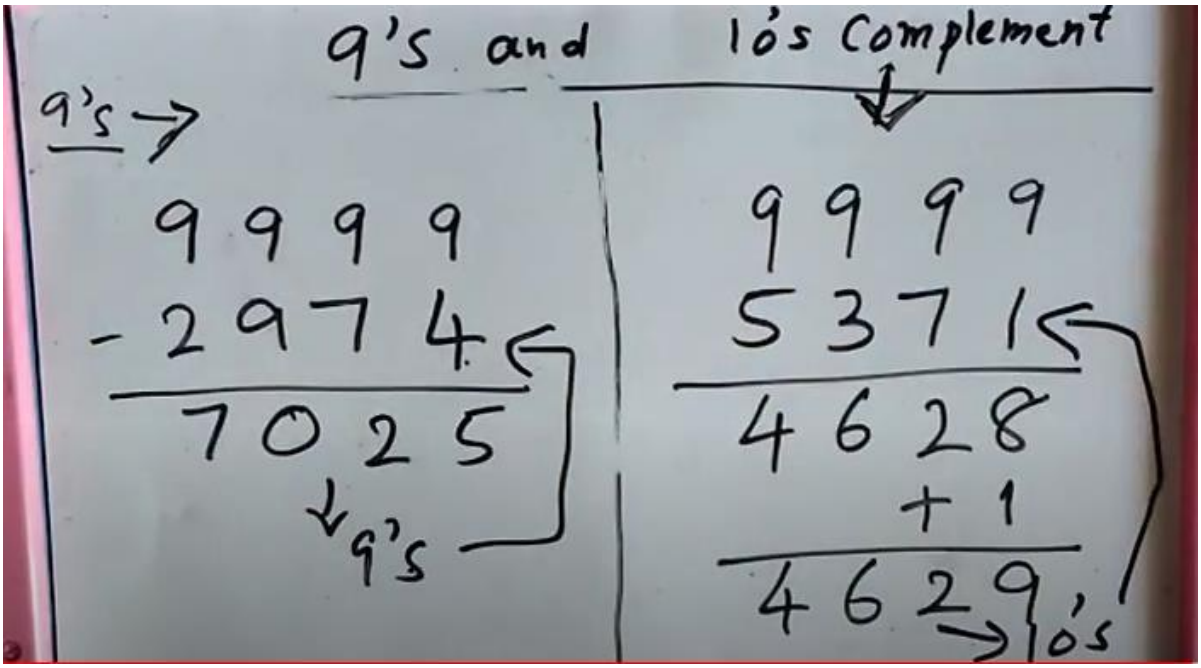
  Ex: - 347

  999 − 347 = 652 (9’s)

  652 + 1 = 653 (10’s)

- 10’s Complement – It is obtained by adding 1 in the 9’s complement.

  Ex: -


9's and 1's Complement
9's →
9999
-2974
7025
9's
9999
5371
4628
+1
4629
1's

1. Find Subtraction of 470 and 231 using 9's complement method

Here A = 470, B = 231.
Find A - B = ? using 9's complement
First find 9's complement of B = 231

Note : 9's complement of a number is obtained by subtracting all bits from 999.
9's complement of 231 is

```
    9  9  9
  - 2  3  1
 ------------
    7  6  8
```

Now Add this 9's complement of B to A

```
  1  1
     4  7  0
  +  7  6  8
 ------------
  1  2  3  8
```

+ Hints : (Move mouse over the steps for detail calculation highlight)

The left most bit of the result is called carry and add it to the rest part of the result 238.

```
     2  3  8
  +        1
 ------------
     2  3  9
```

So answer is 239

. Find Subtraction of 670 and 831 using 9's complement method

lere A = 670, B = 831.
ind A - B = ? using 9's complement
irst find 9's complement of B = 831

lote : 9's complement of a number is obtained by subtracting all bits from 999.
's complement of 831 is

```
   9  9  9
-  8  3  1
-------------
   1  6  8
```

low Add this 9's complement of B to A

```
      1
   6  7  0
+  1  6  8
-------------
   8  3  8
```

+ Hints : (Move mouse over the steps for detail calculation highlight)

lere there is no carry, answer is - (9's complement of the sum obtained 838)

lote : 9's complement of a number is obtained by subtracting all bits from 999.
's complement of 838 is

```
   9  9  9
-  8  3  8
-------------
   1  6  1
```

o answer is -161

1. Find Subtraction of 470 and 231 using 10's complement method

Here A = 470, B = 231.
Find A - B = ? using 10's complement
First find 10's complement of B = 231

Note : 10's complement of a number is 1 added to it's 9's complement number.
9's complement of 231 is

```
   9  9  9
 - 2  3  1
 ---------
   7  6  8
```

Now add 1 : 768 + 1 = 769

Now Add this 10's complement of B to A

```
 1  1
    4  7  0
 +  7  6  9
 ---------
 1  2  3  9
```

+ Hints : (Move mouse over the steps for detail calculation highlight)

The left most bit of the result is called carry and it is ignored.
So answer is 239

2. Find Subtraction of 670 and 831 using 10's complement method

Here A = 670, B = 831.
Find A - B = ? using 10's complement
First find 10's complement of B = 831

Note : 10's complement of a number is 1 added to it's 9's complement number.
9's complement of 831 is

```
   9  9  9
 - 8  3  1
 ----------
   1  6  8
```

Now add 1 : 168 + 1 = 169

Now Add this 10's complement of B to A

```
      1
   6  7  0
 + 1  6  9
 ----------
   8  3  9
```

+ Hints : (Move mouse over the steps for detail calculation highlight)

Here there is no carry, answer is - (10's complement of the sum obtained 839)

Note : 10's complement of a number is 1 added to it's 9's complement number.
9's complement of 839 is

1. Logic AND Function
2. Logic OR Function
3. Logic NOT Function
4. Logic NAND Function
5. Logic NOR Function
6. Laws of Boolean Algebra
7. Boolean Algebra Truth Tables
8. Boolean Algebra Examples
9. DeMorgan's Theorem
10. Switching Theory
11. Sum of Product
12. Product of Sum
13. Boolean Algebra Simplification

Digital Circuit?

Definition: A digital circuit is designed by using a number of logic gates on a single integrated circuit – IC. The input to any digital circuit is in the binary form "0's" and "1's". The output obtained on processing raw digital data is of a precise value. These circuits can be represented in 2 ways either in a combinational way or a sequential way.

Digital Circuit Basics

Digital circuit design was first started up with a design of relays, later vacuum tubes, TTL Transistor-Transistor Logic, Emitter coupled logic, and CMOS logic. These designs use a large number of logical gates like AND, OR, NOT, etc integrated on a single IC. The input and output of digital data are represented in the logical truth table and timing diagram.

Gates

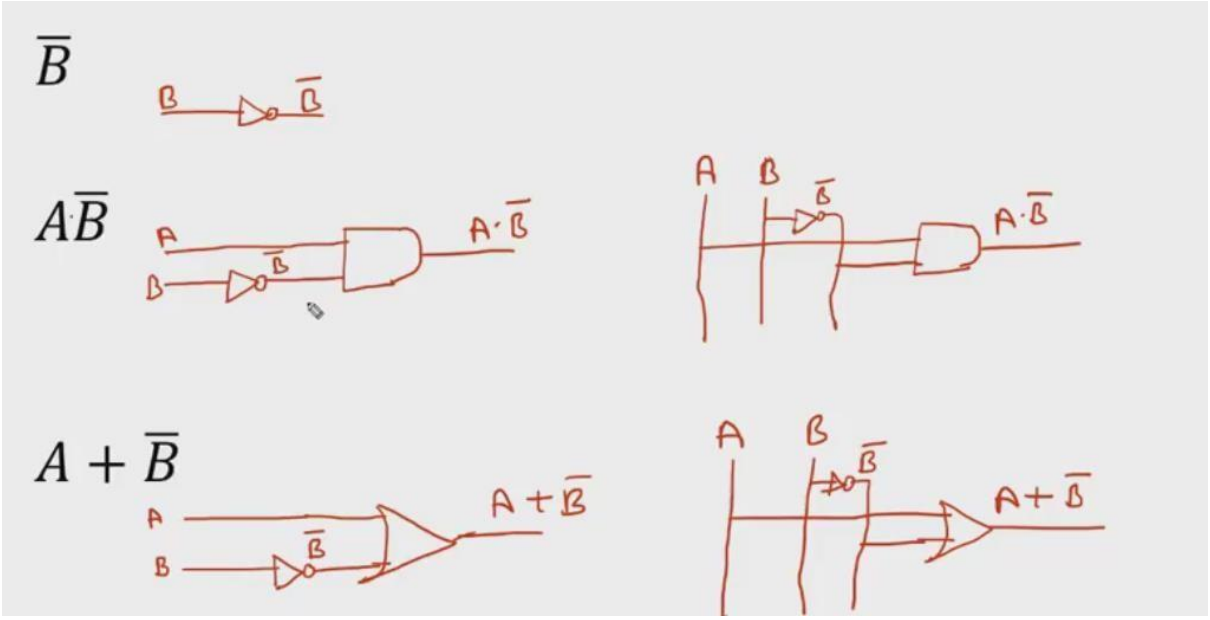
A logic gate is an electronic component that is implemented using a Boolean function. Gates are usually implemented using diodes, transistors, and relays. There are different types of logical gates they are, AND, OR, NOT, NANAD, NOR, XOR. Among which AND, OR, NOT are basic gates and NAND and NOR are the universal gate. Let us consider AND gate representation as below, which has 2 inputs and one output.

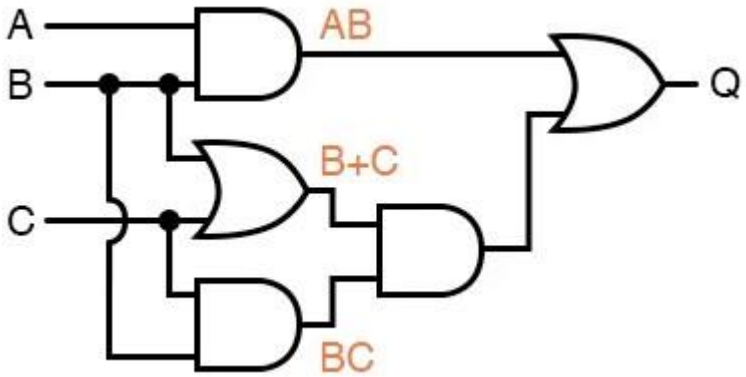
1). What are digital circuits used for?
Digital circuits are used to perform Boolean logic operations.

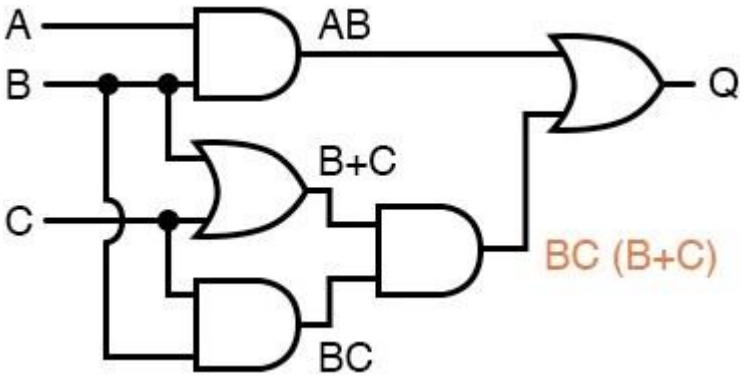
2). How does digital circuit work?
Digital circuit works with discrete signals, which are represented in the binary form of 0's and 1's.

3). What are the basic components of the digital circuit?
The basic components of the digital circuits are flip flops, diodes, transistors, Gates, etc.

4). What is a circuit made of?
An electronic circuit is made up of a number of passive and active components, which are connected using conducting wires.

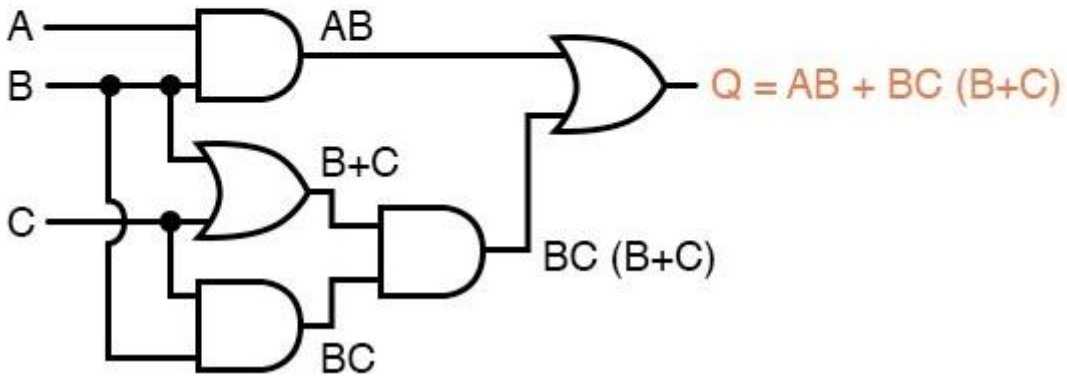




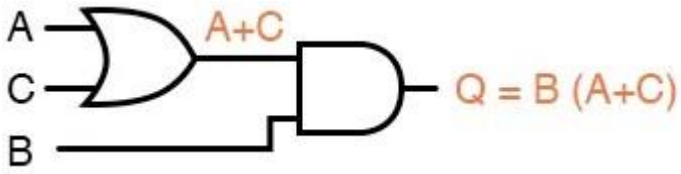


QN expression AB + BC(B + C):

output ("Q") is seen to be equal to the expression AB + BC(B + C):

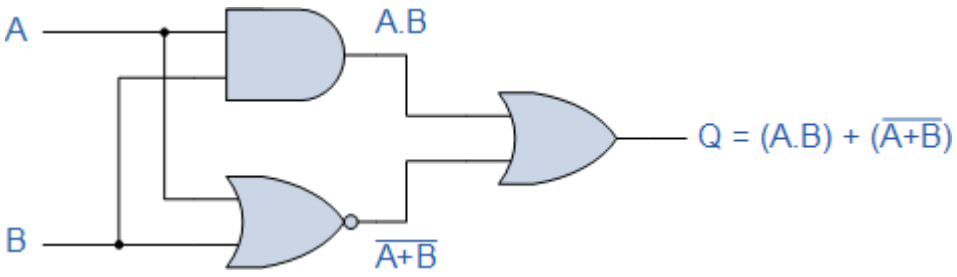


QN expression "B(A + C)"



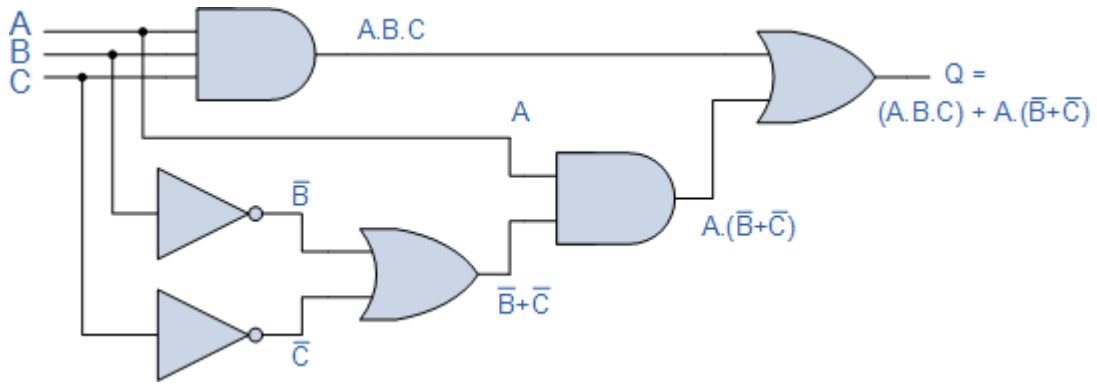
QN given as

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

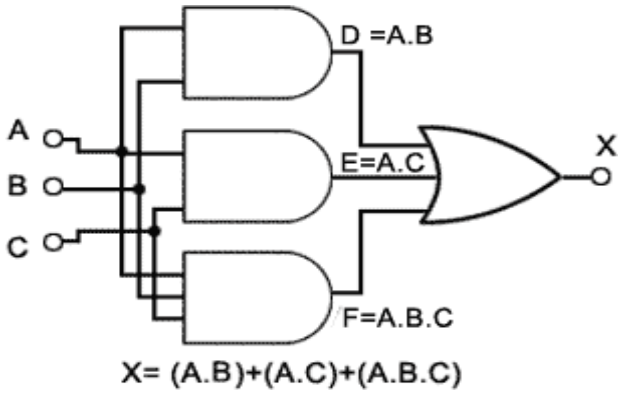


QN-

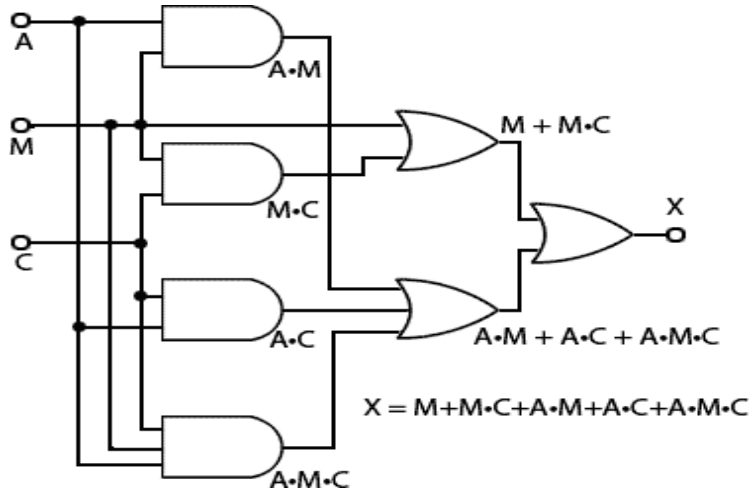
Q= $(A.B.C) + A.(\overline{B}+\overline{C})$

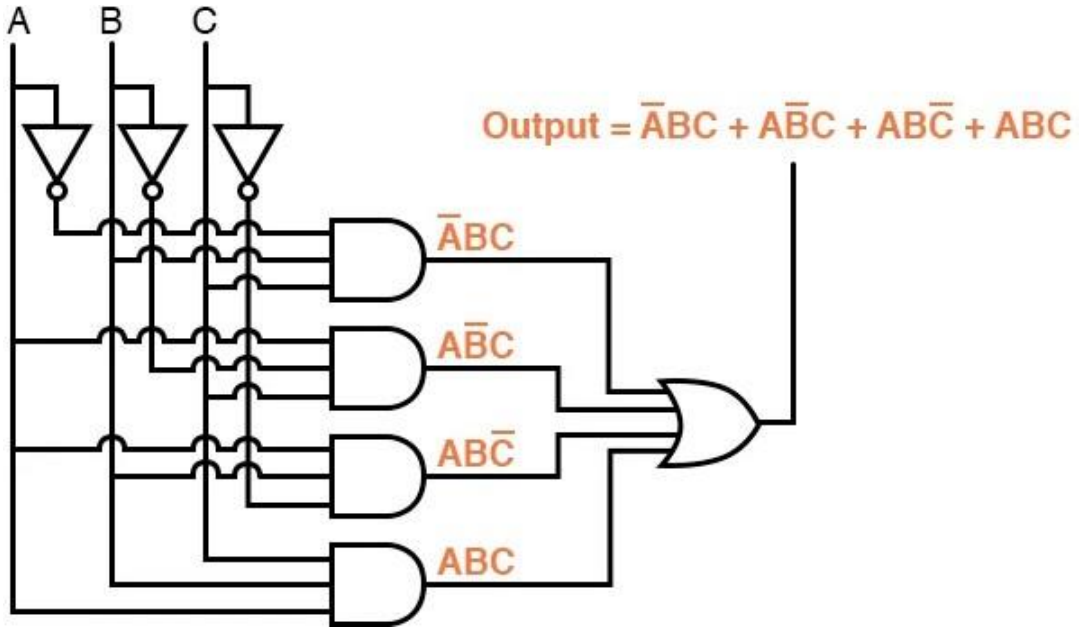


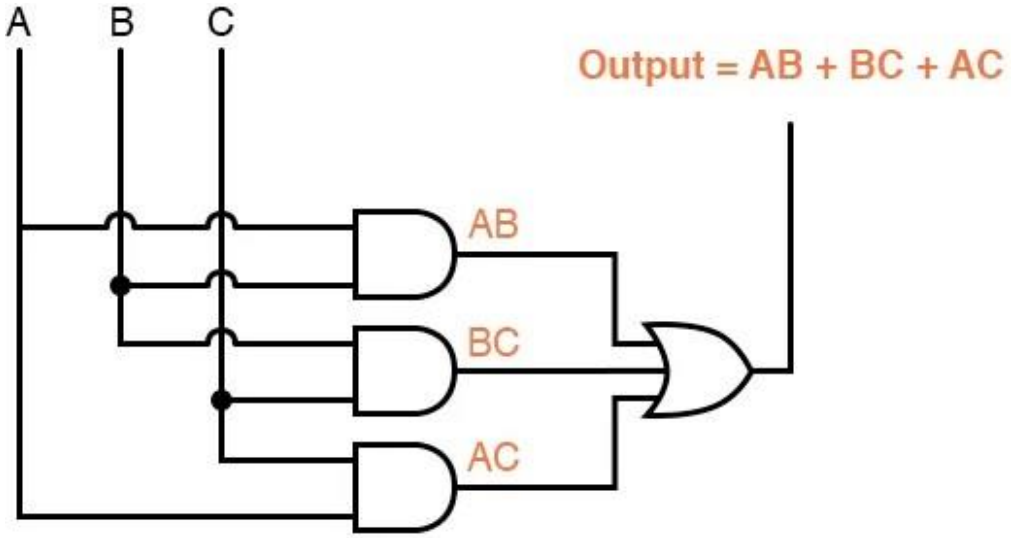
QN **X = (A•B) + (A•C) + (A•B•C)**



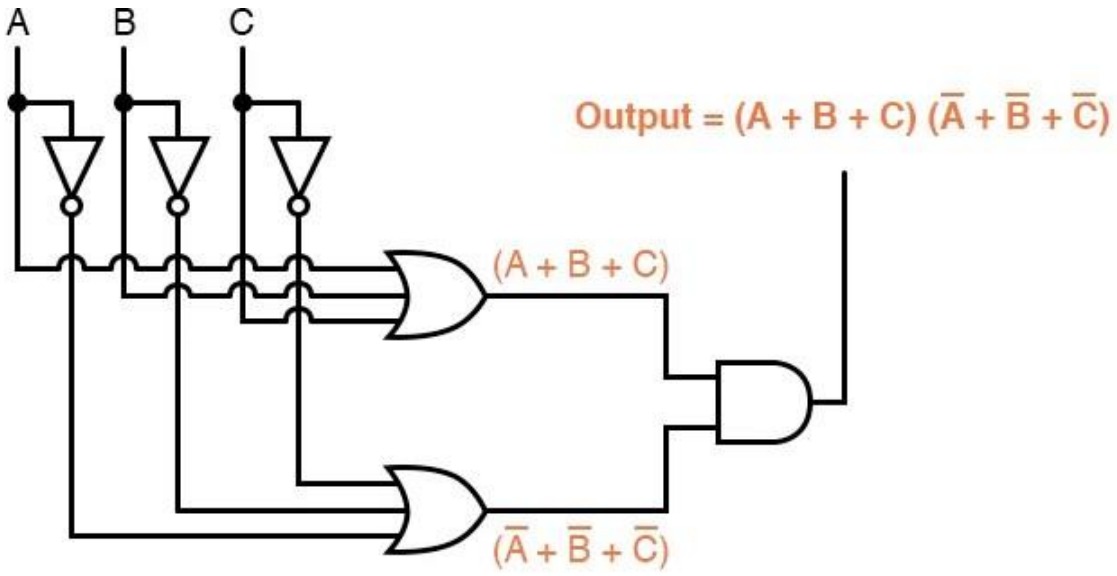
QN **X = M + M•C + A•C + A•M + A•C•M**

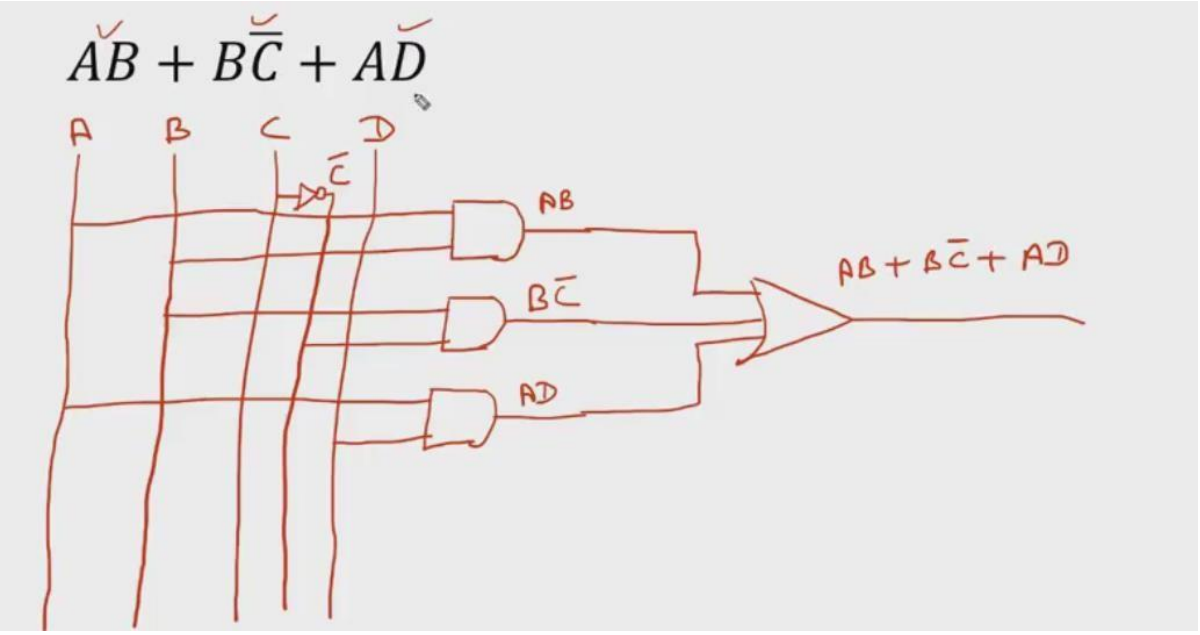


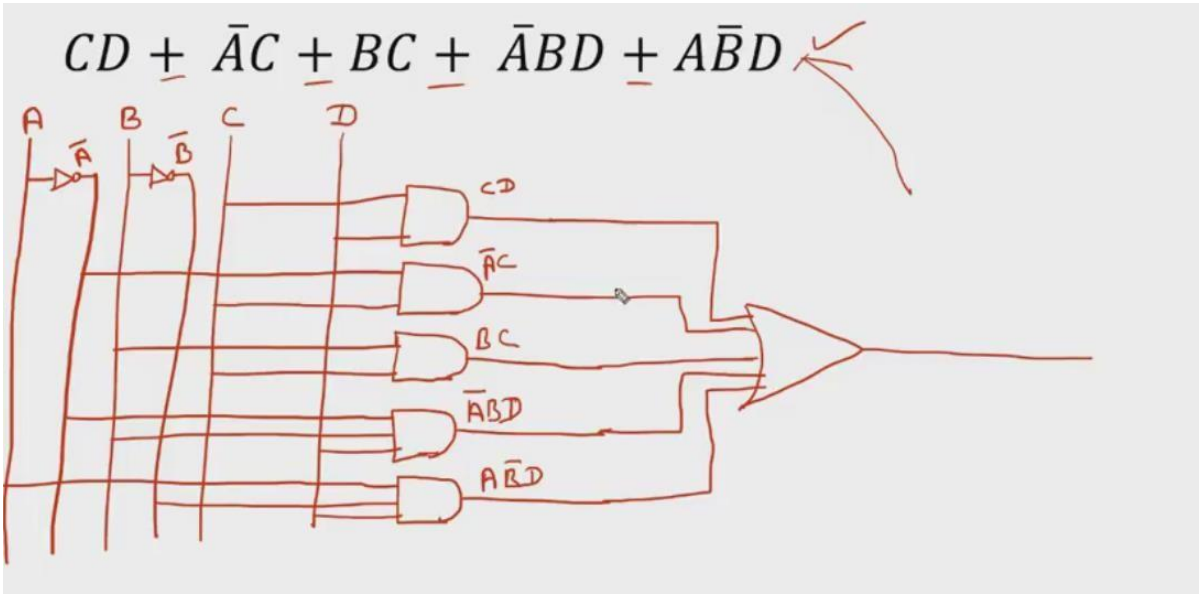

A
B
C
Output = $\bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$
$\bar{A}BC$
$A\bar{B}C$
$AB\bar{C}$
ABC

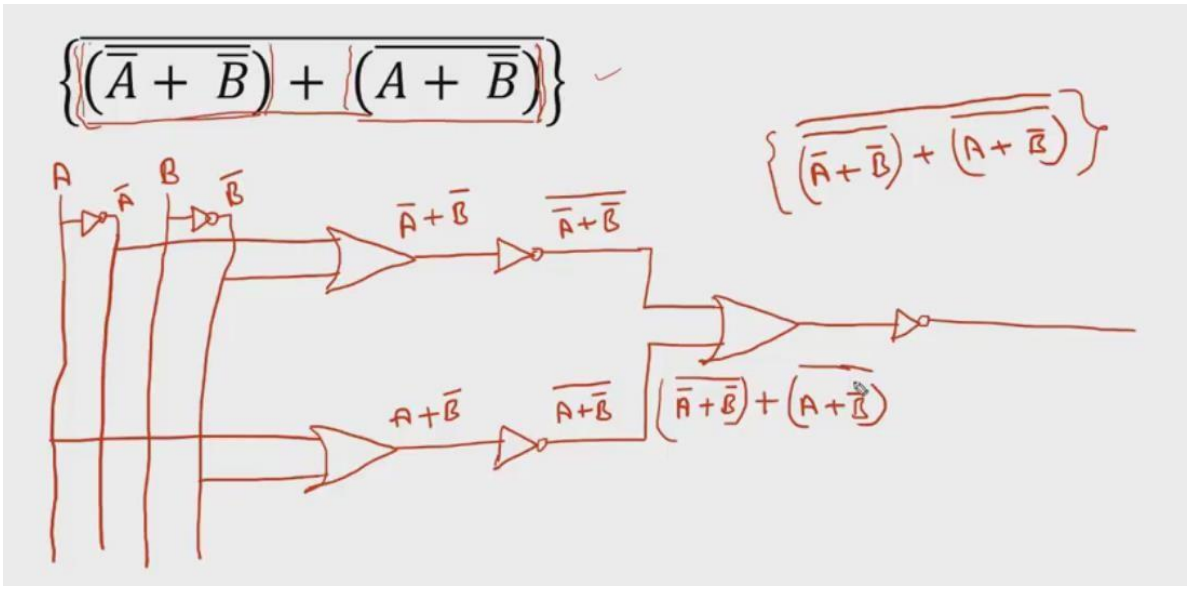

A
B
C
Output = AB + BC + AC
AB
BC
AC

S


A
B
C
Output = (A + B + C) (Ā + B̄ + C̄)
(A + B + C)
(Ā + B̄ + C̄)


$A\breve{B} + B\breve{\bar{C}} + A\breve{D}$
A
B
C
D
C̄
AB
BC̄
AD
AB + BC̄ + AD


$CD + \bar{A}C + BC + \bar{A}BD + A\bar{B}D$
A
B
C
D
$\bar{A}$
$\bar{B}$
CD
$\bar{A}C$
BC
$\bar{A}BD$
$A\bar{B}D$


$\{(\overline{\bar{A} + \bar{B}}) + (\overline{A + \bar{B}})\}$
A
B
$\bar{A}$
$\bar{B}$
$\bar{A}+\bar{B}$
$\overline{\bar{A}+\bar{B}}$
$A+\bar{B}$
$\overline{A+\bar{B}}$
$\{(\overline{\bar{A}+\bar{B}}) + (\overline{A+\bar{B}})\}$

LAKSHMAN JI

Boolean Algebra — CSO.

0 1 — Circuit / TruthTable

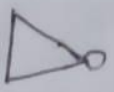
Mobile
Desktop — Circuit used in motherboard

Three circuits

NAND
•
(multiplication of number/digit/bit)

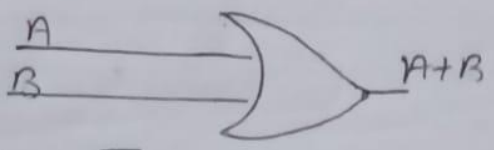
OR
+
(sum number digit)

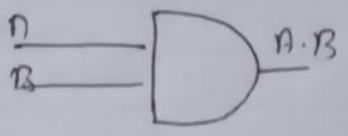
NOT
' or -
(bar (number digit))



Q1) Draw the circuit of the ① A+B, ② A·B, ③ $\bar{A}$
Ans. A and B is input.

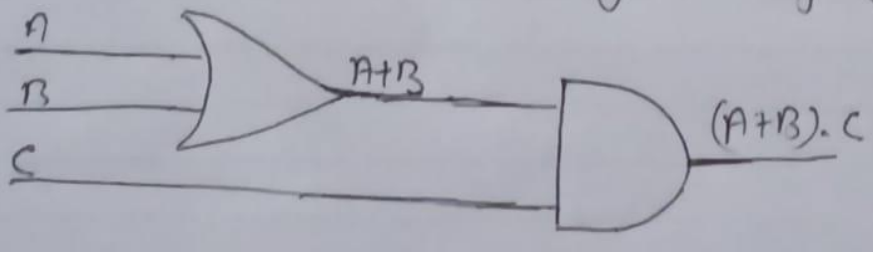
Solution





BODMOS - Rule

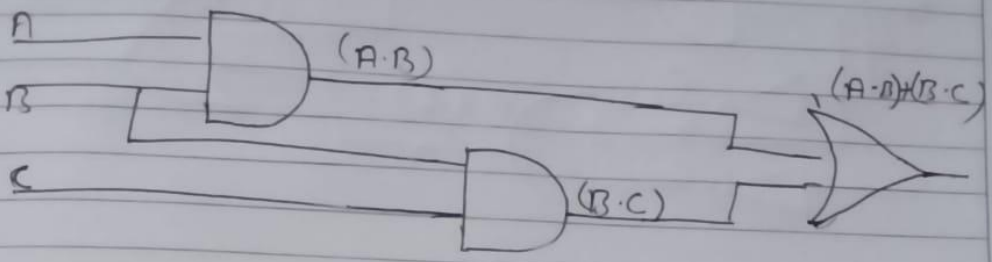
Q2) Draw the circuit Diagram of (A+B).C



Expt. No. Date

Page No.

Q1) $(A \cdot B) + (B \cdot C)$

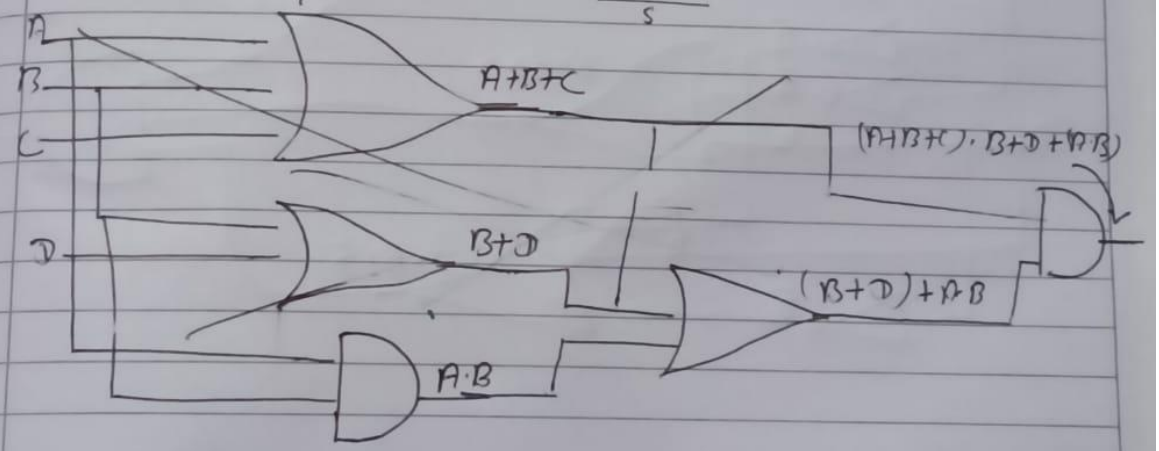


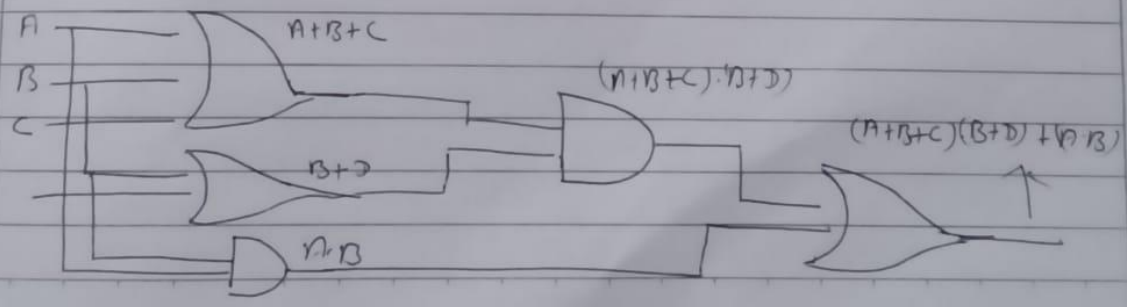
Q2) Draw the circuit Diagrams of:

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

① ② ③

4 5

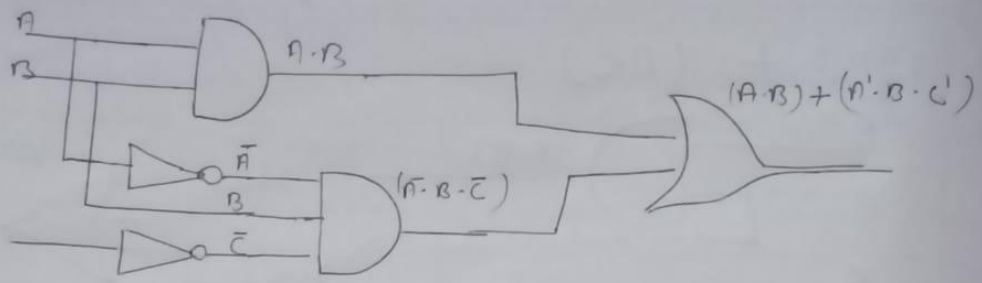




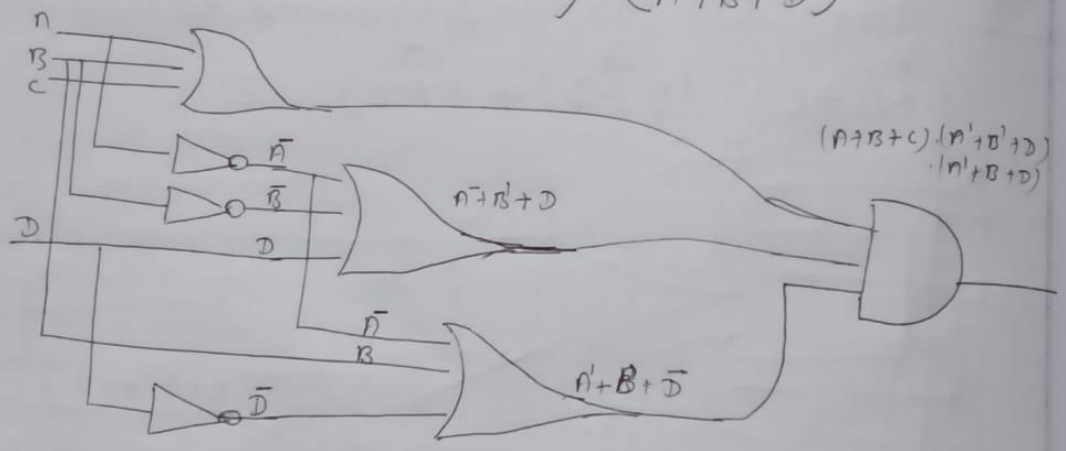
Teacher's Signature

QN.7 - Draw the circuit diagram

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

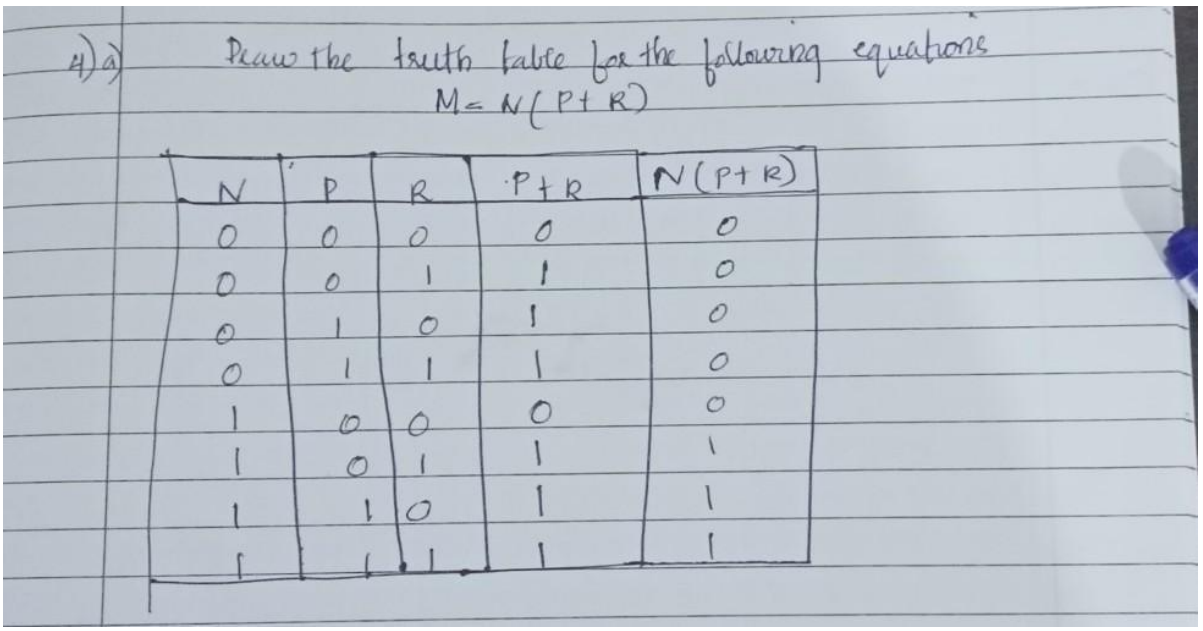


QN 8. $(A+B+C) \cdot (A'+B'+D) \cdot (A'+B+D')$



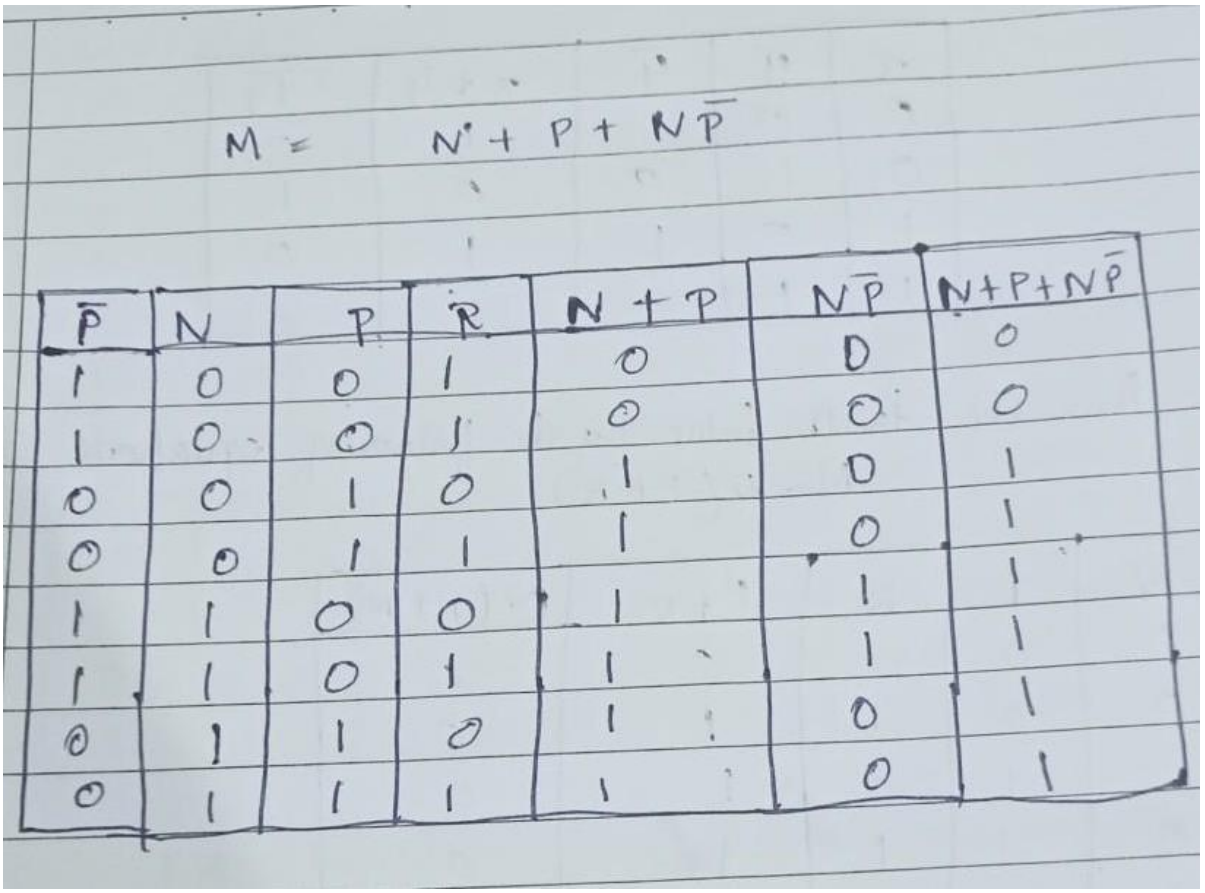
(a) M = N (P + R)

N	P	R	P + R	N (P + R)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1



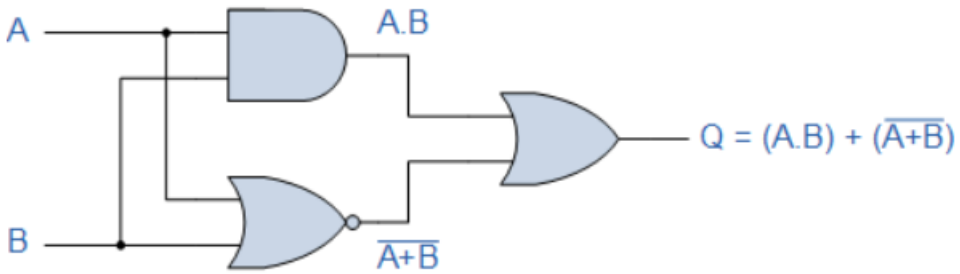
(b) M = N + P + NP'

N	P	R	P'	NP'	N + P + NP'
0	0	0	1	0	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	0	0	1



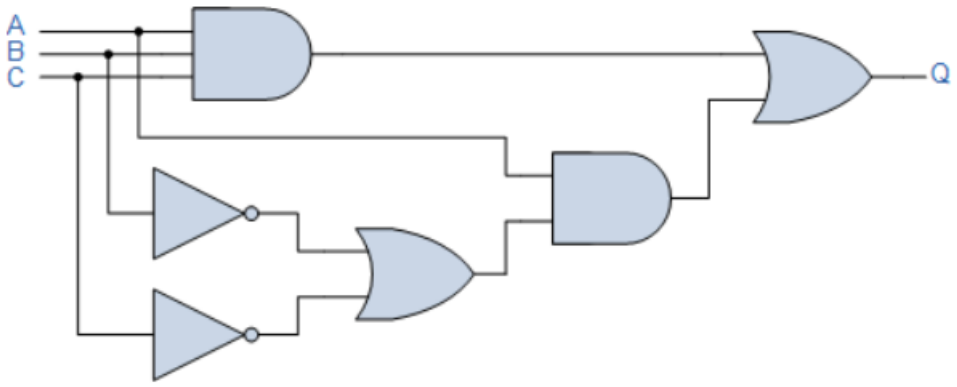
Q = (A.B) + (A+B),

Inputs		Intermediates		Output
B	A	A.B	$\overline{A+B}$	Q
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

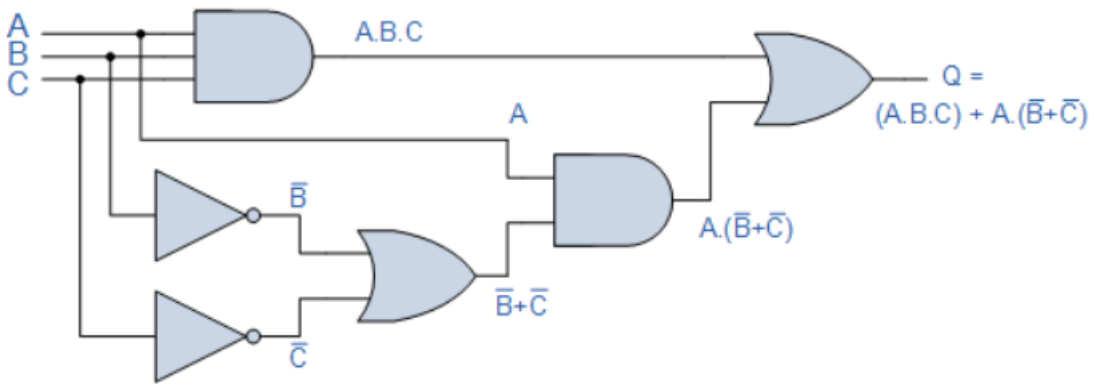


Example No3

Find the Boolean algebra expression for the following system.



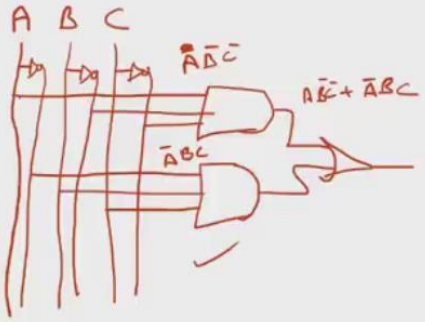
solution



Inputs			Intermediates					Output
C	B	A	A.B.C	$\overline{B}$	$\overline{C}$	$\overline{B}+\overline{C}$	$A.(\overline{B}+\overline{C})$	Q
0	0	0	0	1	1	1	0	0
0	0	1	0	1	1	1	1	1
0	1	0	0	0	1	1	0	0
0	1	1	0	0	1	1	1	1
1	0	0	0	1	0	1	0	0
1	0	1	0	1	0	1	1	1
1	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	1

$A\bar{B}\bar{C} + \bar{A}BC$

A	B	C	$\bar{A}$	$\bar{B}$	$\bar{C}$	$A\bar{B}\bar{C}$	$\bar{A}BC$	$A\bar{B}\bar{C} + \bar{A}BC$



$A\bar{B}\bar{C} + \bar{A}BC$

$2^n = 2^3 = 8/2 = 4/2 = 2/2 = 1$

A	B	C	$\bar{A}$	$\bar{B}$	$\bar{C}$	$A\bar{B}\bar{C}$	$\bar{A}BC$	$A\bar{B}\bar{C} + \bar{A}BC$
0	0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	0	0
0	1	0	1	0	1	0	0	0
0	1	1	1	0	0	0	1	1
1	0	0	0	1	1	1	0	1
1	0	1	0	1	0	0	0	0
1	1	0	0	0	1	0	0	0
1	1	1	0	0	0	0	0	0

2^n AND

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

What Is Boolean Logic?

Boolean logic refers to the form of algebra where the variables have only 2 unique values i.e. TRUE or FALSE or ON and OFF. These values are often used as **1** or **0** in binary language or **High** and **low** logic respectively.

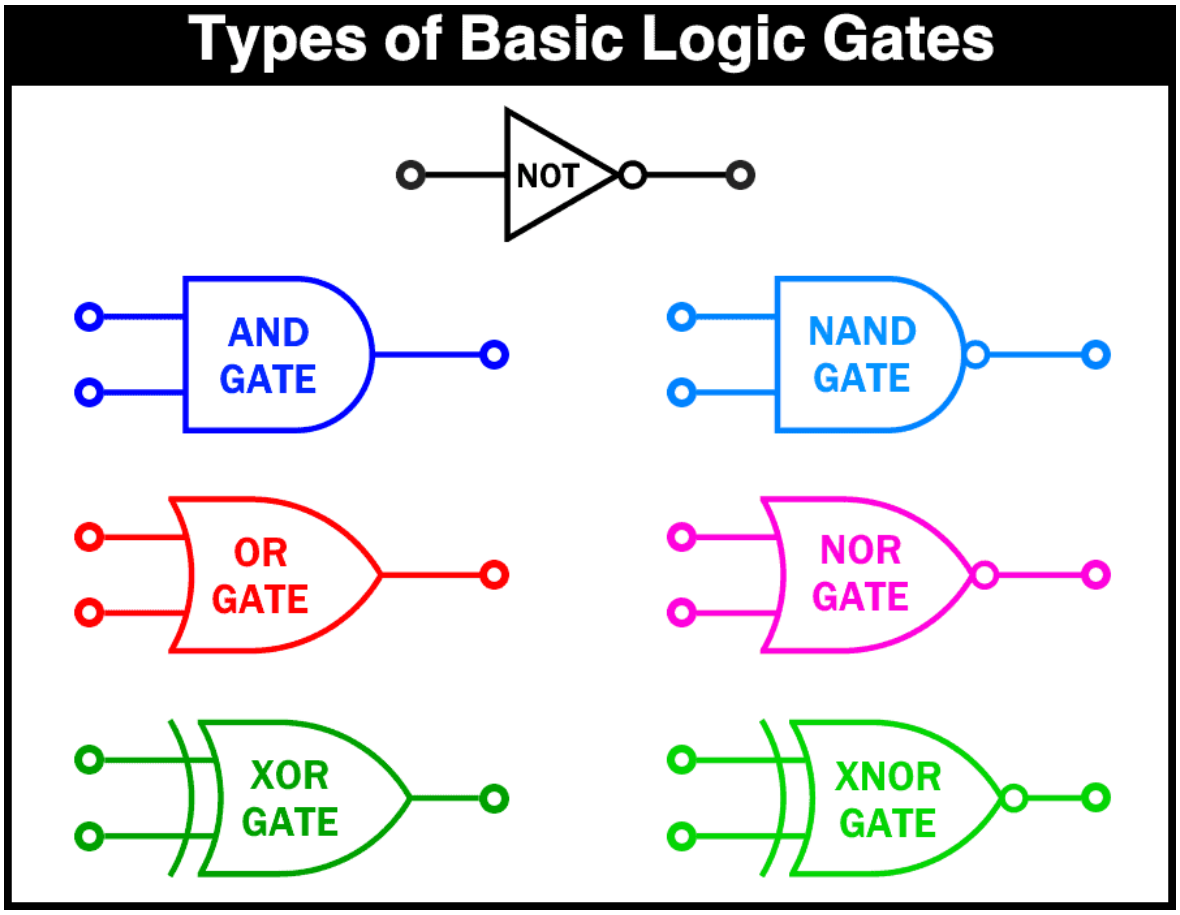
1	**0**
True	**False**
High	**Low**
ON	**OFF**

Boolean Function

A **Boolean function** is a logical operation of one or more than one variables whose resultant is a single binary bit. It can only be either TRUE or FALSE. Boolean functions are based on **Boolean logic**.

What is a Digital Logic Gate?

A **digital logic gate** is an electronic component which implements a Boolean function. these logic gates may have two or more than two binary inputs and provides a single binary output. Some of these basic logic gates are given below:



Boolean algebra

Boolean Algebra is used to analyze and simplify the digital (logic) circuits. It uses only the binary numbers i.e. 0 and 1. It is also called as **Binary Algebra** or **logical Algebra**. Boolean algebra was invented by **George Boole** in 1854.

The logical symbol 0 and 1 are used for representing the digital input or output. The symbols "1" and "0" can also be used for a permanently open and closed digital circuit. The digital circuit can be made up of several logic gates. To perform the logical operation with minimum logic gates, a set of rules were invented, known as the **Laws of Boolean Algebra**. These rules are used to reduce the number of logic gates for performing logic operations.

Rules in Boolean algebra

Following are the important rules used in Boolean algebra.

1. Only two values(1 for high and 0 for low) are possible for the variable used in Boolean algebra.
2. The overbar(-) is used for representing the complement variable. So, the complement of variable C is represented as .
3. The plus(+) operator is used to represent the OR in of the variables.
4. The dot(.) operator is used to represent the AND in of the variables.

OR

- operation between two variables denoted using plus (+) symbol (A OR B as A + B)

- AND operation between two variables denoted using dot (.) symbol (A AND B as A . B)

- NOT operation is a unary operation i.e. complement of a variable (NOT A as or A’)

  Why the use of Boolean Algebra
  Any Logical Expression minimization using of Boolean algebra reduced the hardiness of the Expression

Function	Description	Expression
1.	NULL	0
2.	IDENTITY	1
3.	Input A	A
4.	Input B	B
5.	NOT A	$\overline{A}$
6.	NOT B	$\overline{B}$
7.	A AND B (AND)	$A.B$
8.	A AND NOT B	$A.\overline{B}$
9.	NOT A AND B	$\overline{A}.B$
10.	NOT AND (NAND)	$\overline{A.B}$
11.	A OR B (OR)	$A+B$
12.	A OR NOT B	$A+\overline{B}$
13.	NOT A OR B	$\overline{A}+B$
14.	NOT OR (NOR)	$\overline{A+B}$
15.	Exclusive-OR	$A.\overline{B}+\overline{A}.B$
16.	Exclusive-NOR	$A.B+\overline{A}.\overline{B}$

Law/Theorem	Law of Addition	Law of Multiplication
Identity Law	$x + 0 = x$	$x \cdot 1 = x$
Complement Law	$x + x' = 1$	$x \cdot x' = 0$
Idempotent Law	$x + x = x$	$x \cdot x = x$
Dominant Law	$x + 1 = 1$	$x \cdot 0 = 0$
Involution Law	$(x')' = x$	
Commutative Law	$x + y = y + x$	$x \cdot y = y \cdot x$
Associative Law	$x+(y+z) = (x+y)+z$	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
Distributive Law	$x \cdot (y+z) = x \cdot y+x \cdot z$	$x+y \cdot z = (x+y) \cdot (x+z)$
Demorgan's Law	$(x+y)' = x' \cdot y'$	$(x \cdot y)' = x' + y'$
Absorption Law	$x + (x \cdot y) = x$	$x \cdot (x + y) = x$

OR

Boolean Algebra Simplification Table

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\overline{A} = 0$	$A + \overline{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$

PROPERTY	AND	OR
Commutative	$AB = BA$	$A + B = B + A$
Associative	$(AB)\,C = A\,(BC)$	$(A + B) + C = A + (B + C)$
Distributive	$A\,(B + C) = (AB) + (AC)$	$A + (BC) = (A + B)\,(A + C)$
Identity	$A1 = A$	$A + 0 = A$
Complement	$A(A') = 0$	$A + (A') = 1$
De Morgan's law	$(AB)' = A'$ OR B'	$(A + B)' = A'B'$

Canonical Form/ Standard Form

Boolean expression where each term contains all Boolean variables in their true or complemented form.

Ex:- ① $ABC + B\bar{C}$ ✗

② $ABC + AB\bar{C}$ ✓

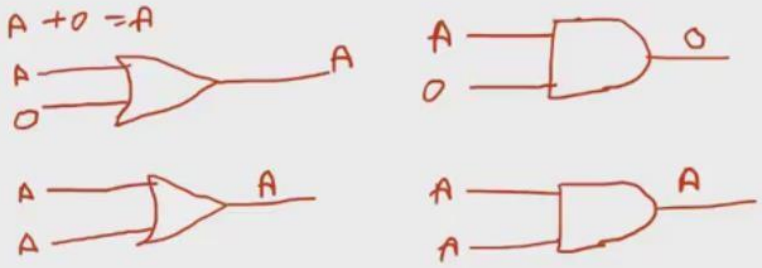
③ $ABC + \bar{A}B\bar{C}$ ✓

Boolean Laws

Law	Dual Pair	Remark
$A + B = B + A$	$A.B = B.A$	Commutative Law
$A + (B + C) = (A + B) + C$	$A\,(BC) = (AB)\,C$	Associative Law
$A\,(B + C) = AB + AC$	$A + (BC) = (A + B)(A + C)$	Distributive Law
$A + 1 = 1$	$A.1 = A$	Identity Law/ Redundancy Law
$A + 0 = A$	$A.0 = 0$	
$A + A = A$	$A.A = A$	
$A + \bar{A} = 1$	$A.\bar{A} = 0$	
$A + AB = A$	$A\,(A + B) = A$	Absorption Law
$\bar{\bar{A}} = A$	-	Involution Law
$A + \bar{A}B = A + B$	$A\,(\bar{A} + B) = A$	-
$\bar{B} + \bar{B} = B$	-	-

Identity Law/ Redundancy Law

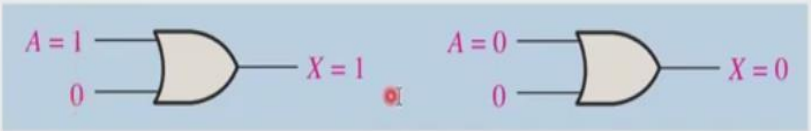
$A+1=1$	$A.1=A$	Identity Law/ Redundancy Law
$A+0=A$	$A.0=0$	
$A+A=A$	$A.A=A$	
$A+\bar{A}=1$	$A.\bar{A}=0$	



Basic rules of Boolean algebra.

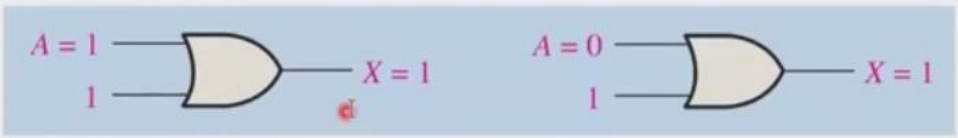
1. $A + 0 = A$
2. $A + 1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = A$
5. $A + A = A$
6. $A + \bar{A} = 1$
7. $A \cdot A = A$
8. $A \cdot \bar{A} = 0$
9. $\bar{\bar{A}} = A$
10. $A + AB = A$
11. $A + \bar{A}B = A + B$
12. $(A + B)(A + C) = A + BC$

Rule 1: $A + 0 = A$ A variable ORed with 0 is always equal to the variable. If the input variable A is 1, the output variable X is 1, which is equal to A. If A is 0, the output is 0, which is also equal to A. This rule is illustrated in Figure 4–8, where the lower input is fixed at 0.



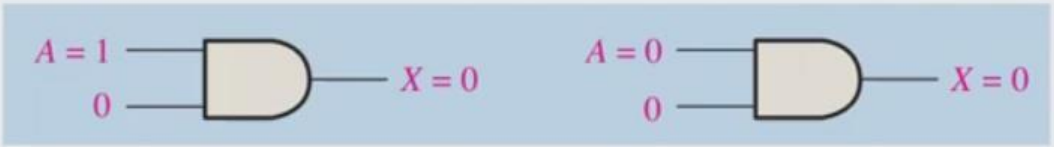
$X = A + 0 = A$

Rule 2: $A + 1 = 1$ A variable ORed with 1 is always equal to 1. A 1



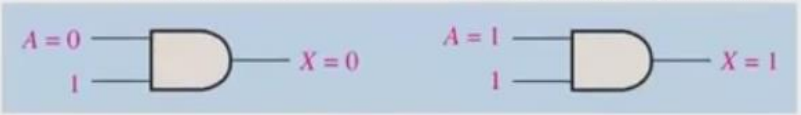
$X = A + 1 = 1$

Rule 3: $A \cdot 0 = 0$ A variable ANDed with 0 is always equal to 0.



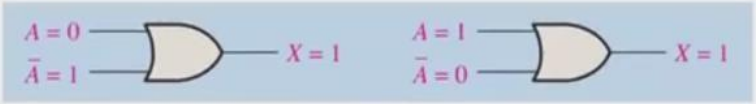
$X = A \cdot 0 = 0$

Rule 4: $A \cdot 1 = A$ A variable ANDed with 1 is always equal to the variable. If A is 0,



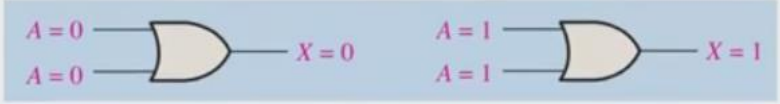
$X = A \cdot 1 = A$

Rule 6: $A + \bar{A} = 1$ A variable ORed with its complement is always equal to 1. If A is 0, then $0 + \bar{0} = 0 + 1 = 1$. If A is 1, then $1 + \bar{1} = 1 + 0 = 1$. See Figure 4–13, where one input is the complement of the other.



$X = A + \bar{A} = 1$

Rule 5: $A + A = A$ A variable ORed with itself is always equal to the variable. If A is 0, then $0 + 0 = 0$; and if A is 1, then $1 + 1 = 1$. This is shown in Figure 4–12, where both inputs are the same variable.



$X = A + A = A$

Associative Law

- Associative Law of Addition

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

A	B	C	B + C	A + (B+C)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

A	B	C	A + B	(A+B) + C
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Associative Law

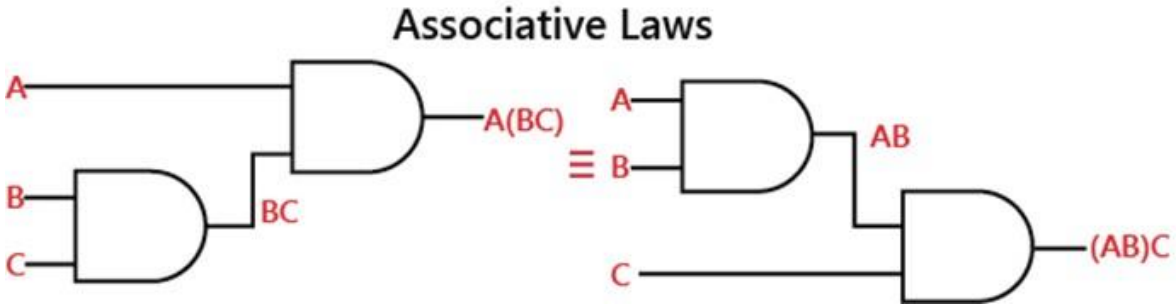
- Associative Law of Addition

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

A	B	C	B + C	A + (B+C)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

A	B	C	A + B	(A+B) + C
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

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



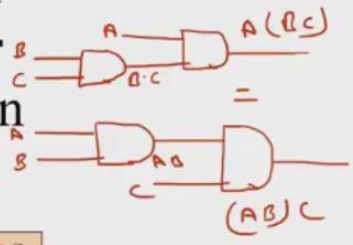
Associative Law

- Associative Law of Multiplication

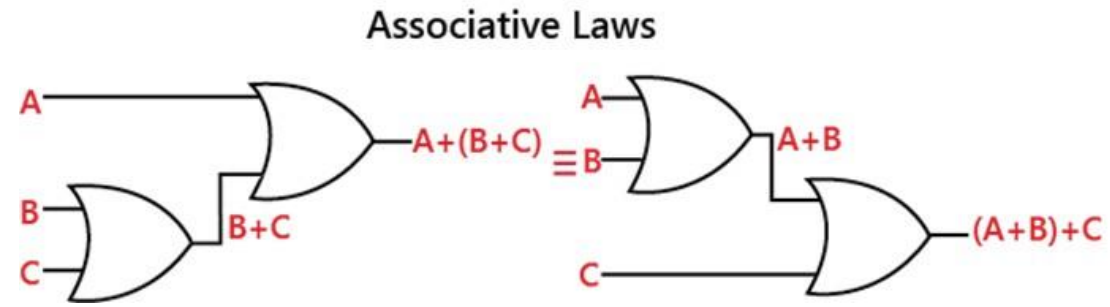
$$A\ (B\ C) = (A\ B)\ C$$

A	B	C	B C	A (B C)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

A	B	C	A B	(A B) C
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1



$A(BC) = (AB)C$



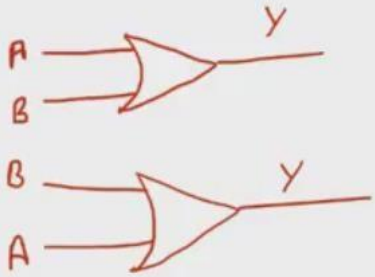
Commutative Law

- Commutative Law of Addition

$$A + B = B + A$$

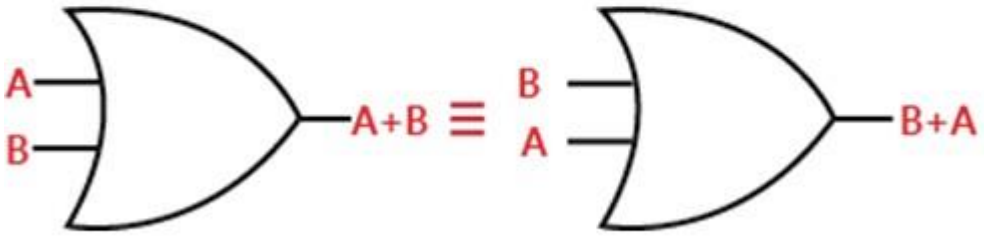
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

B	A	B + A
0	0	0
0	1	1
1	0	1
1	1	1



A+B = B+A

Commutative laws



For two variables, the commutative law of multiplication is written as:

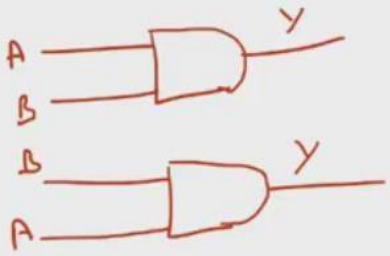
Commutative Law

- Commutative Law of Multiplication

A.B = B.A

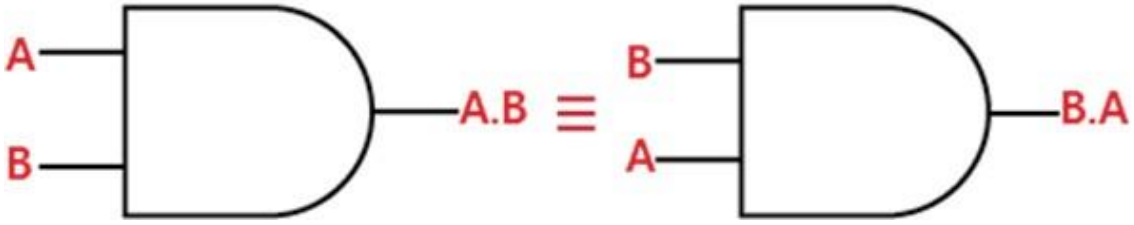
A	B	A B
0	0	0
0	1	0
1	0	0
1	1	1

B	A	B A
0	0	0
0	1	0
1	0	0
1	1	1



A.B = B.A

Commutative laws



De-Morgan's Theorem

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

  The complement of a sum is equal to the product of complement

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

  The complement of a product is the equal to sum of complement

De – morgan's law
1. (A.B)' = A' + B'
2.(A + B)' = A'.B'

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

A	B	$\bar{A}$	$\bar{B}$	A+B	$\overline{A+B}$	$\bar{A}\cdot\bar{B}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

$$\overline{(AB)} = \bar{A} + \bar{B}$$

A	B	$\bar{A}$	$\bar{B}$	AB	$\overline{AB}$	$\bar{A}+\bar{B}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

DeMorgan's First Theorem

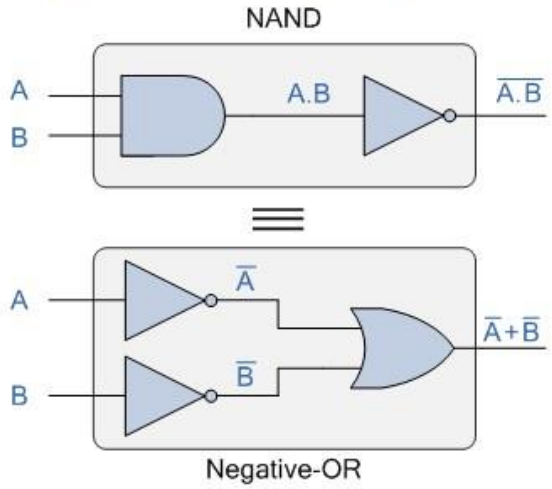
DeMorgan's First theorem proves that when two (or more) input variables are AND'ed and negated, they are equivalent to the OR of the complements of the individual variables. Thus the equivalent of the NAND function will be a negative-OR function, proving that $\overline{A.B} = \overline{A}+\overline{B}$. We can show this operation using the following table.

Verifying DeMorgan's First Theorem using Truth Table

Inputs		Truth Table Outputs For Each Term				
B	A	A.B	$\overline{A.B}$	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$
0	0	0	**1**	1	1	**1**
0	1	0	**1**	0	1	**1**
1	0	0	**1**	1	0	**1**
1	1	1	**0**	0	0	**0**

We can also show that $\overline{A.B} = \overline{A}+\overline{B}$ using logic gates as shown.

DeMorgan's First Law Implementation using Logic Gates



The top logic gate arrangement of: $\overline{A.B}$ can be implemented using a standard NAND gate with inputs A and B. The lower logic gate arrangement first inverts the two inputs producing $\overline{A}$ and $\overline{B}$. These then become the inputs to the OR gate. Therefore the output from the OR gate becomes: $\overline{A}+\overline{B}$

DeMorgan's Second Theorem

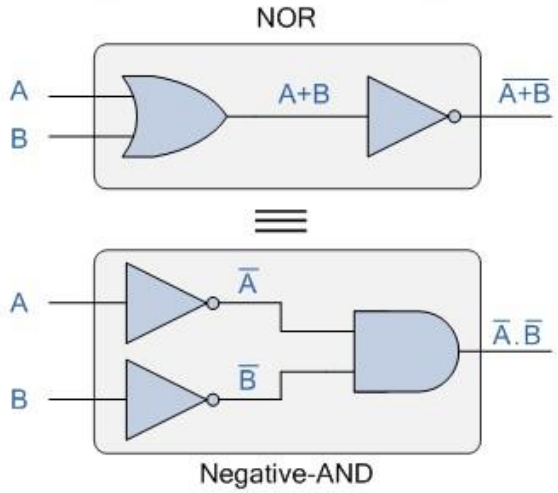
DeMorgan's Second theorem proves that when two (or more) input variables are OR'ed and negated, they are equivalent to the AND of the complements of the individual variables. Thus the equivalent of the NOR function is a negative-AND function proving that $\overline{A+B} = \overline{A}.\overline{B}$, and again we can show operation this using the following truth table.

Verifying DeMorgan's Second Theorem using Truth Table

Inputs		Truth Table Outputs For Each Term				
B	A	A+B	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
0	0	0	**1**	1	1	**1**
0	1	1	**0**	0	1	**0**
1	0	1	**0**	1	0	**0**
1	1	1	**0**	0	0	**0**

We can also show that $\overline{A+B} = \overline{A}.\overline{B}$ using the following logic gates example.

DeMorgan's Second Law Implementation using Logic Gates



The top logic gate arrangement of: $\overline{A+B}$ can be implemented using a standard NOR gate function using inputs A and B. The lower logic gate arrangement first inverts the two inputs, thus producing $\overline{A}$ and $\overline{B}$. Thus then become the inputs to the AND gate. Therefore the output from the AND gate becomes: $\overline{A}.\overline{B}$

Distributive Law

- Distributive Law of Addition

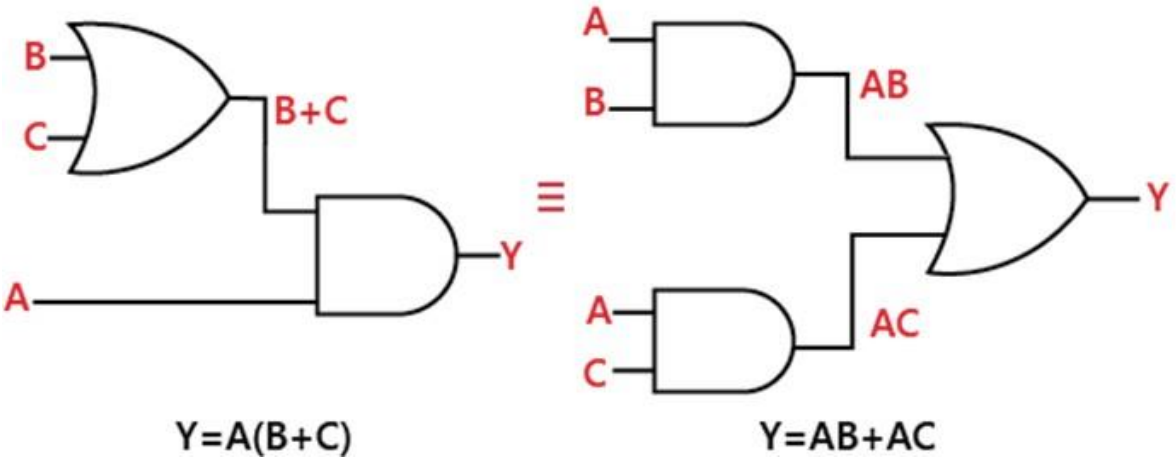
$$A + (BC) = (A + B)(A + C)$$

A	B	C	B C	A + (B C)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A	B	C	(A + B)	(A+C)	(A+B) (A+C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

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

Distributive law



Distributive Law

- Distributive Law of Multiplication

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

A	B	C	B + C	A (B + C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

A	B	C	A B	AC	A B + AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

Duality Theorem

A Boolean relation can be written to another Boolean relation by changing each OR operation to AND Operation and vice versa

Ex:-

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

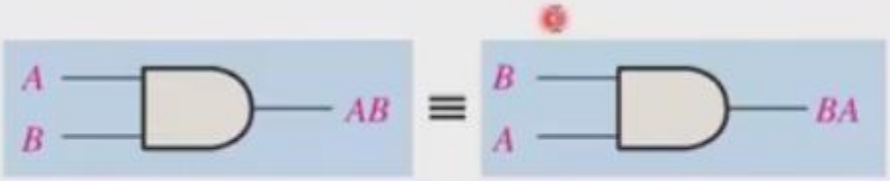
Commutative Laws

The *commutative law of addition* for two variables is written as

$$A + B = B + A$$

The *commutative law of multiplication* for two variables is

$$AB = BA$$

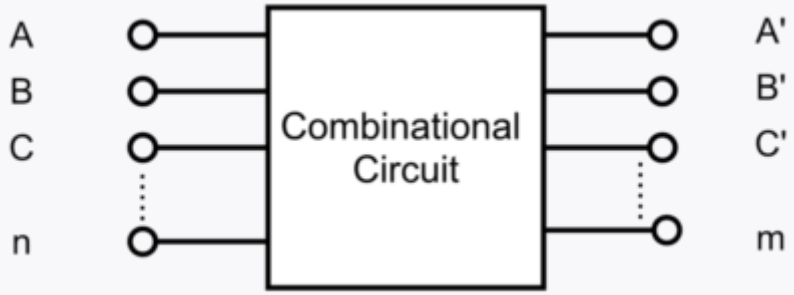


Difference Between Combinational and Sequential Circuit

Combinational Circuit output is dependent only on the inputs at the same instant of time. whereas Sequential Circuit output depends not only on the present inputs but also on the past history of inputs.

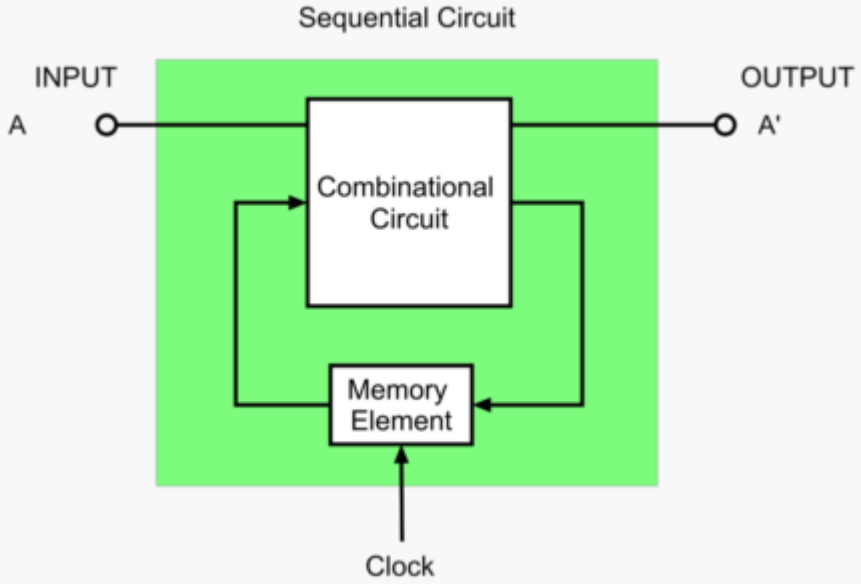
Combinational Circuit

- A combinational circuit can be defined as a circuit whose output is dependent only on the inputs at the same instant of time.
- Example of Combinational Circuit
  - Half Adder
  - Full Adder
  - Half Subtractor
  - Full Subtractor



Sequential Circuit

- A sequential circuit can be defined as a circuit whose output depends not only on the present inputs but also on the past history of inputs.
- Example of Sequential circuit.
  - Flip-Flops
  - Registers
  - Counters

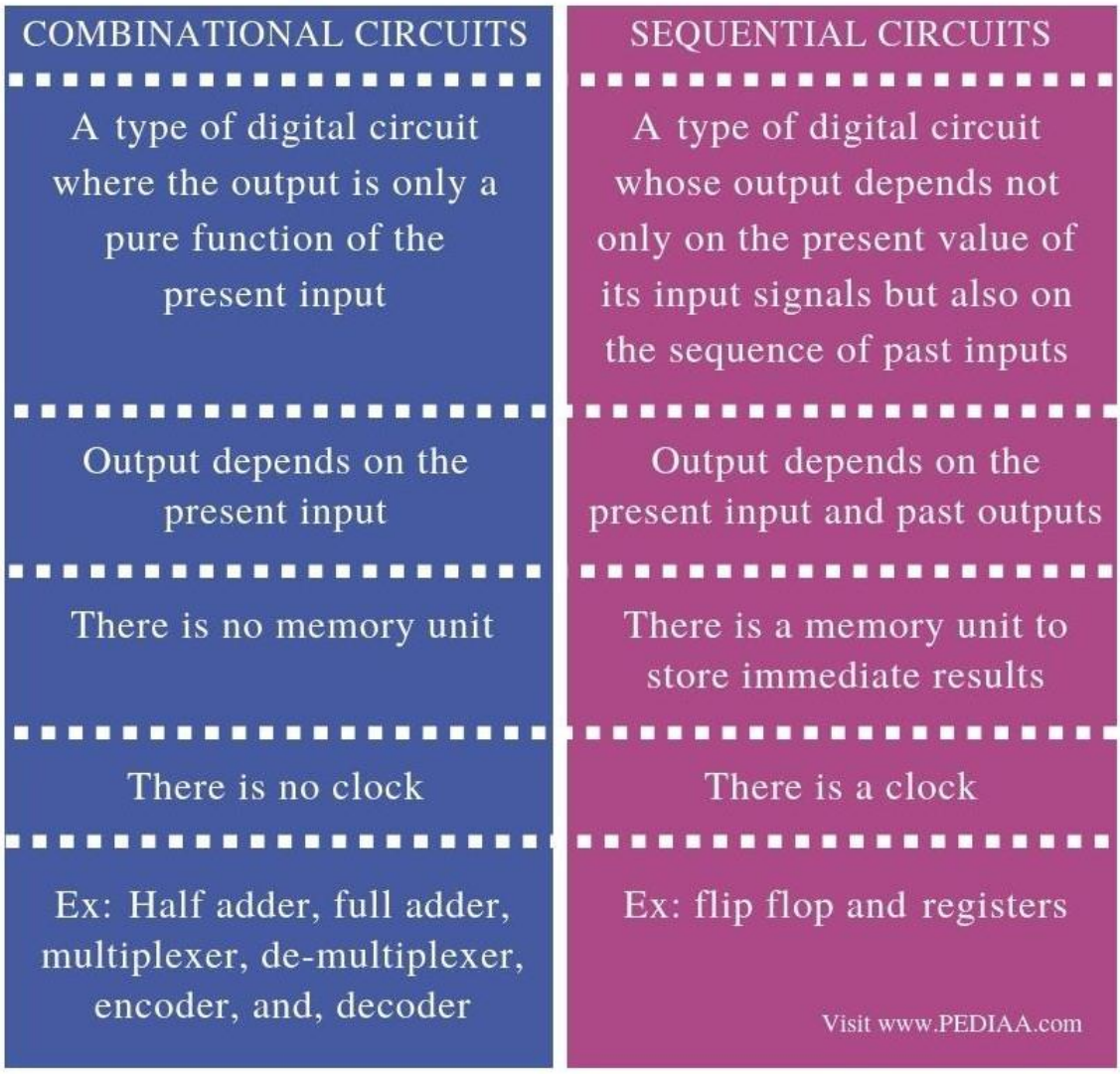


SEQUENTIAL CIRCUITS

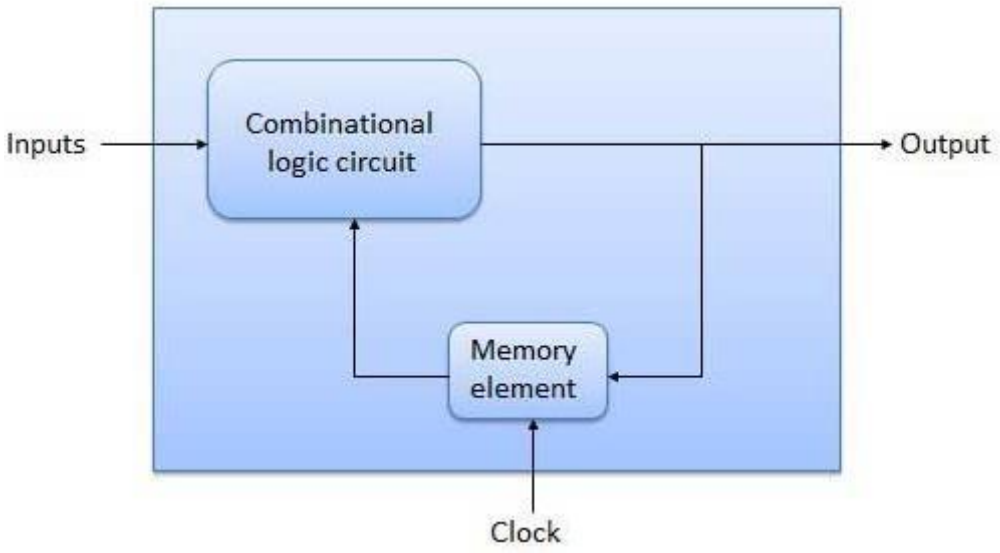
What is the difference between Combinational & Sequential Circuits?

	Combinational Circuit	Sequential Circuit
1	It does not contain memory elements.	It contains memory elements.
2	Output depends on the present state of input only.	Output depends not only on the present inputs but also on the past history of inputs.
3	Its behavior is described by the set of output functions.	Its behavior is described by the set of next-state (memory) functions and the set of output functions.
4	Do not use the Feedback path.	Use Feedback path.
5	Does not require a clock signal.	Most sequential circuits use a clock signal.
6	Faster than the sequential circuit.	Slower than the combinational circuit.
7	**Example**: Adder, Subtractor, MUX, Encoders, etc.	**Example**: Flip Flops, Registers, Counters, etc.

COMBINATIONAL CIRCUITS
VERSUS
SEQUENTIAL CIRCUITS



The combinational circuit does not use any memory. Hence the previous state of input does not have any effect on the present state of the circuit. But sequential circuit has memory so output can vary based on input. This type of circuits uses previous input, output, clock and a memory element.



Moore Machine : output logic does not depends on input directly.
Mealy Machine : output logic depends on input directly.

FLIP FLOP

Flip flop is a sequential circuit which generally samples its inputs and changes its outputs only at particular instants of time and not continuously. Flip flop is said to be edge sensitive or edge triggered rather than being level triggered like latches.
There are 4 types -

1. S-R Flip Flop
2. J-K Flip Flop
3. D Flip Flop
4. T Flip Flop

https://www.javatpoint.com/simplification-of-boolean-expressions-using-karnaugh-map