21CSS201T C - **COMPUTER ORGANIZATION AND ARCHITECTURE**

Dr. S. Anbukkarasi
AP/CSE

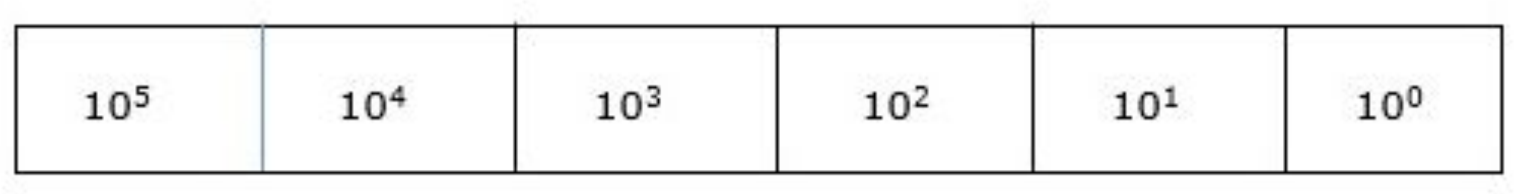
Number System

The technique to represent and work with numbers is called number system.

- Binary
- Decimal
- Octal
- Hexadecimal

Decimal Number System

- Decimal number system is a base 10 number system having 10 digits from 0 to 9.
- This means that any numerical quantity can be represented using these 10 digits.
- Decimal number system is also a positional value system.



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

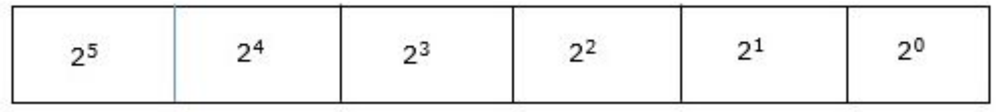
Binary Number System

The easiest way to vary instructions through electric signals is two-state system – on and off.

On is represented as 1 and off as 0, though 0 is not actually no signal but signal at a lower voltage.

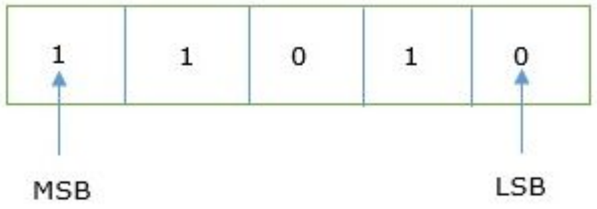
The number system having just these two digits – 0 and 1 – is called binary number system.

Each binary digit is also called a bit. Binary number system is also positional value system, where each digit has a value expressed in powers of 2, as displayed here



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

In any binary number, the rightmost digit is called least significant bit (LSB) and leftmost digit is called most significant bit (MSB).



Decimal Equivalent of : 110011

Memory

Computer memory is measured in terms of how many bits it can store. Here is a chart for memory capacity conversion.

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

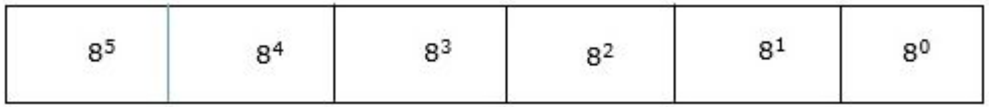
1 Exabyte (EB) = 1024 PB

1 Zettabyte = 1024 EB

1 Yottabyte (YB) = 1024 ZB

Octal Number System

Octal number system has eight digits – 0, 1, 2, 3, 4, 5, 6 and 7. Octal number system is also a positional value system with where each digit has its value expressed in powers of 8, as shown here −

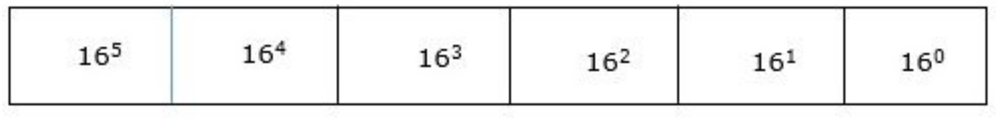


8^5	8^4	8^3	8^2	8^1	8^0

Decimal Equivalent of : 645_8 ->

Hexadecimal Number System

Octal number system has 16 symbols – 0 to 9 and A to F where A is equal to 10, B is equal to 11 and so on till F. Hexadecimal number system is also a positional value system with where each digit has its value expressed in powers of 16, as shown here −



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

$35AB_{16}$ equivalent to: ?

ASCII

Besides numerical data, computer must be able to handle alphabets, punctuation marks, mathematical operators, special symbols, etc. that form the complete character set of English language.

The most widely used alphanumeric code is **American Standard Code for Information Interchange (ASCII)**.

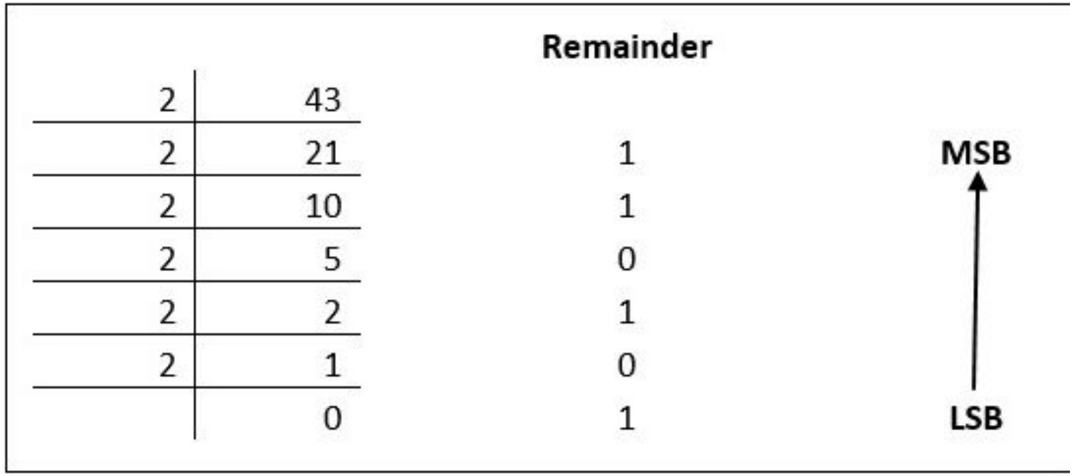
ASCII is a 7**-bit code that has 128 (27) possible codes**.

Unicode

Conversion

Decimal to Binary

Decimal numbers can be converted to binary by repeated division of the number by 2 while recording the remainder. Let’s take an example to see how this happens.



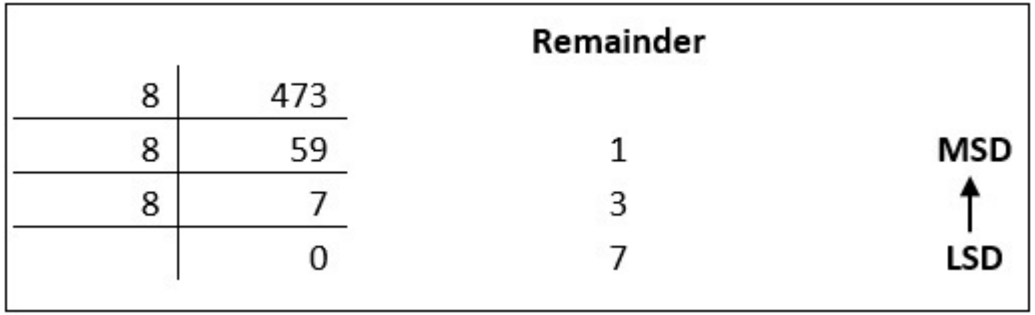
		Remainder	
2	43		
2	21	1	MSB
2	10	1	
2	5	0	
2	2	1	
2	1	0	
	0	1	LSB

ıe remainders are to be read from bottom to top to obtain the binary equivalent.

$43_{10} = 101011_2$

Decimal to Octal

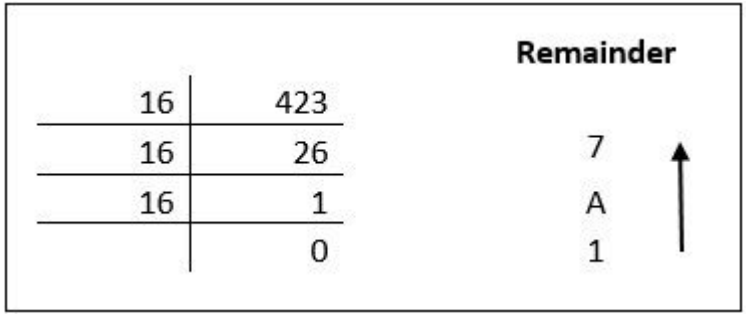
Decimal numbers can be converted to octal by repeated division of the number by 8 while recording the remainder. $473_{10} = 731_8$



		Remainder	
8	473		
8	59	1	MSD
8	7	3	↑
	0	7	LSD

Decimal to Hexadecimal

Decimal numbers can be converted to octal by repeated division of the number by 16 while recording the remainder.



		Remainder
16	423	
16	26	7
16	1	A
	0	1

$423_{10} = 1A7_{16}$

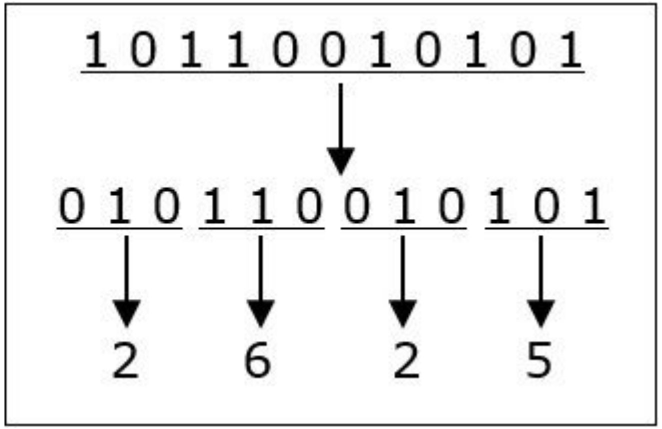
Binary to Octal and Vice Versa

To convert a binary number to octal number, these steps are followed −

Starting from the least significant bit, make groups of three bits.

If there are one or two bits less in making the groups, 0s can be added after the most significant bit

Convert each group into its equivalent octal number



Binary to Hexadecimal

To convert a binary number to hexadecimal number, these steps are followed −

- Starting from the least significant bit, make groups of four bits.
- If there are one or two bits less in making the groups, 0s can be added after the most significant bit.
- Convert each group into its equivalent hexa number.

Decimal to Binary (With Fractions)

8.235

.235 => .235 * 2 => 0. 470

0. 470 => 0. 470 * 2 => 0.94

0.94 => 0. 94 * 2 => 1.88

8=> 1000

ANSWER: 1000 . 001

Binary to Decimal (With Fractions)

1110.001

.001 => $0 * 2^{-1} + 0 * 2^{-2} + 1 * 2^{-3}$

=> 0 + 0 + ⅛ = > 0.125

1110 => 14

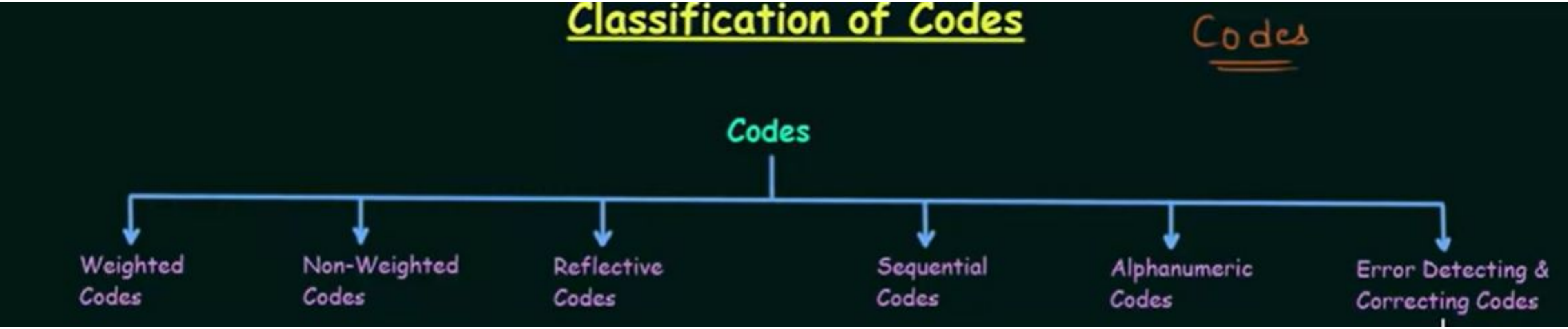
ANSWER: 14.125

Binary Representation (Other way)

	8	4	2	1
0 =>	0	0	0	0
1=>	0	0	0	1
2 =>	0	0	1	0
…				
10 =>	1	0	1	0

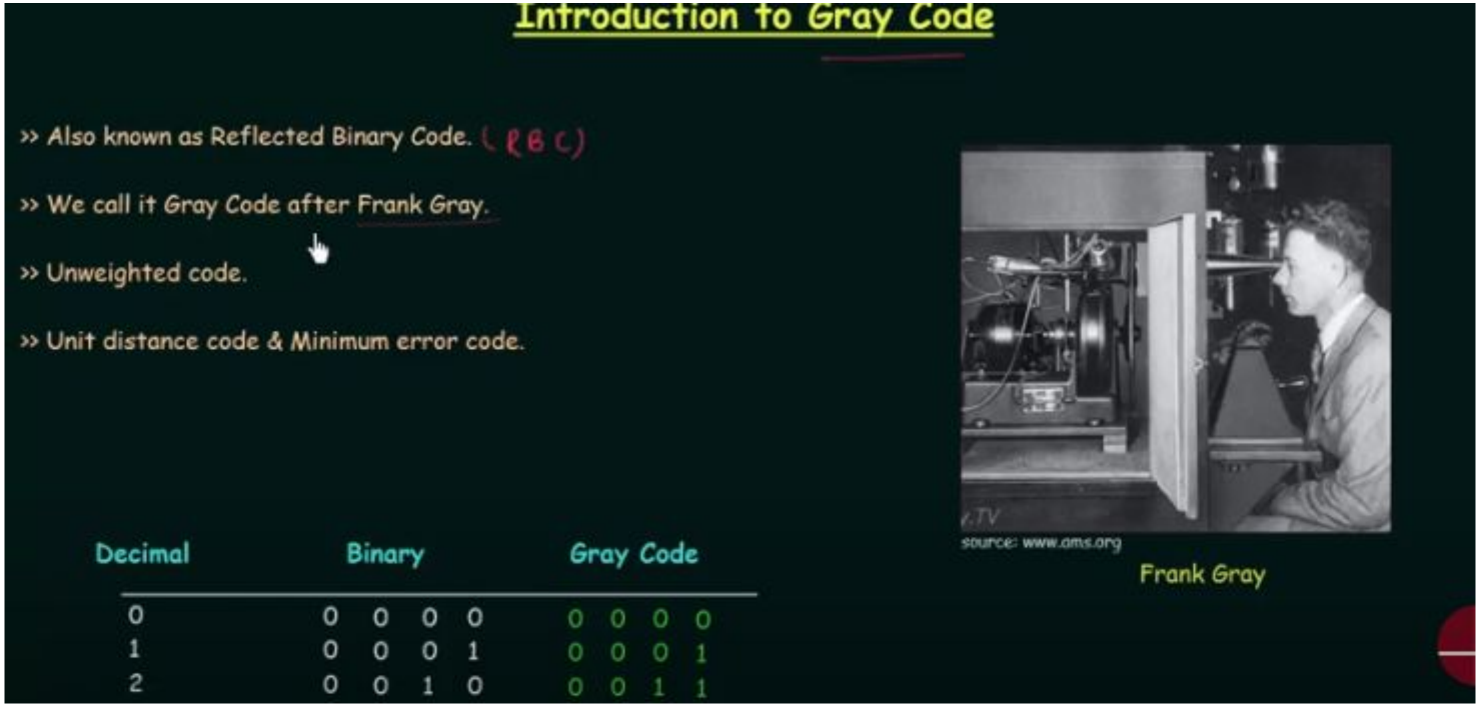
Codes

Group of symbols - Codes



- Weighted Codes - Binary, 8421
- Non Weighted Codes - XS- 3, Gray
- Reflective codes (Self complementing code) - 2421,XS-3 [9 and 0 , 8 and 1 etc are compliments with each other)
- Sequential code - 8421, XS-3
- Alphanumeric codes - ASCII
- Error Detecting and Correcting codes - Hamming

Gray



Introduction to Gray Code

>> Also known as Reflected Binary Code. (RBC)

>> We call it Gray Code after Frank Gray.

>> Unweighted code.

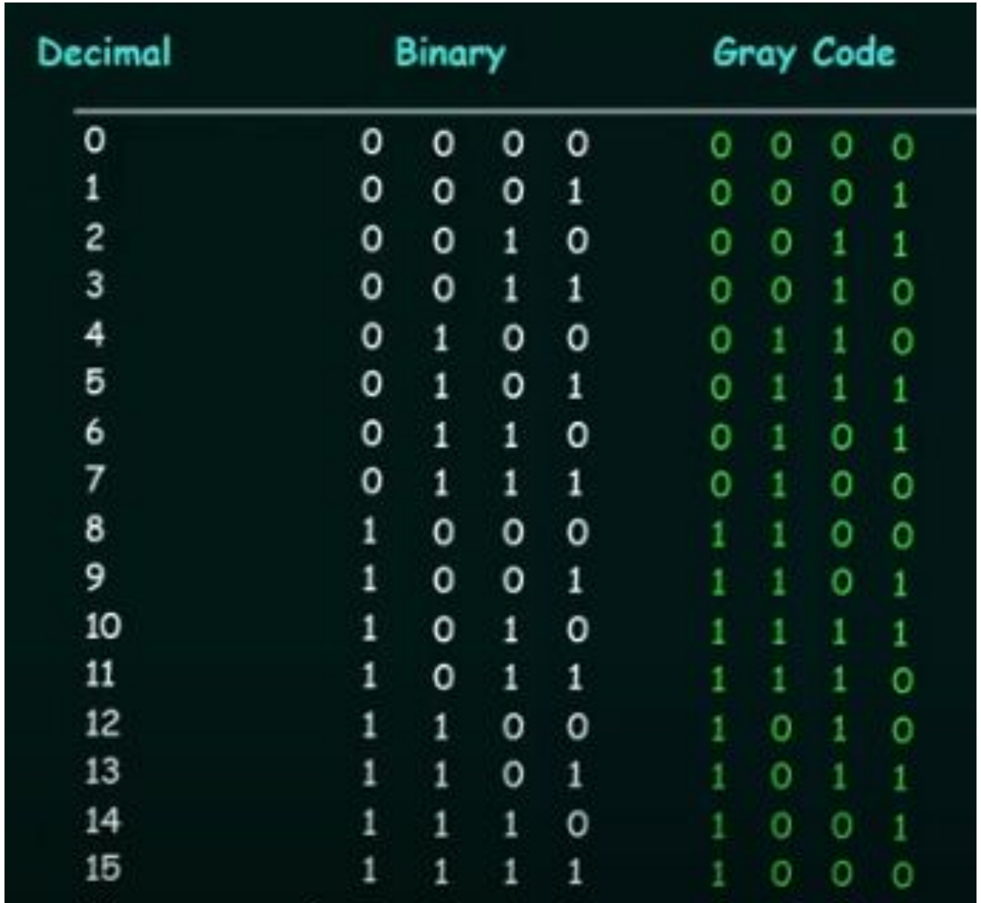
>> Unit distance code & Minimum error code.

Decimal	Binary	Gray Code
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1

source: www.ams.org

Frank Gray

-> Uses in Digital operations



Decimal	Binary				Gray Code			
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Binary to Gray:

1 0 1 1 =>

$\downarrow$

1 1 1 0

(1 + 0) (0 + 1) (1+1)

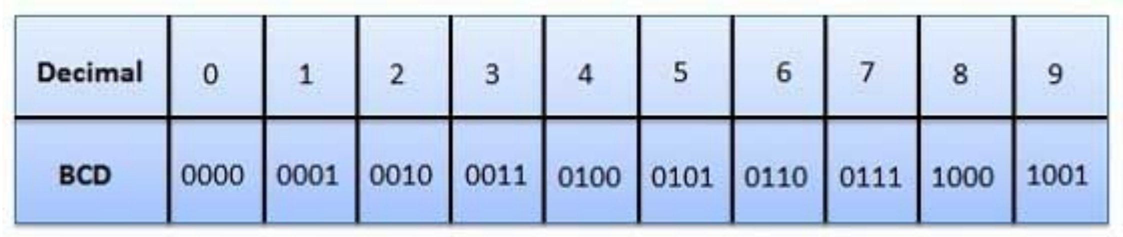
Gray to Binary:

$$\begin{tikzcd}
1 \arrow[d] & 1 & 1 & 0 \\
1 & 0 \arrow[ur] & 1 \arrow[ur] & 1
\end{tikzcd}$$

(1 + 1)

Binary Coded Decimal (BCD) code

- In this code each decimal digit is represented by a 4-bit binary number. [0-9]
- 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. (only decimal digits not numbers)**
- **Positional values 8 4 2 1**



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

Advantages of BCD Codes

It is very similar to decimal system.

We need to remember binary equivalent of decimal numbers 0 to 9 only.

Disadvantages of BCD Codes

The addition and subtraction of BCD have different rules.

The BCD arithmetic is little more complicated.

BCD needs more number of bits than binary to represent the decimal number. So BCD is less efficient than binary.

Decimal to BCD:

*Convert (123)10 in BCD = > **Packed BCD***

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

BCD to Decimal:

1. *0001 0010 0011*

123

1. *10 100*

0001 0100 => 14

BCD Addition

Simple Binary Addition

1. Sum <= 9 Final Carry = 0 (Answer Correct)
2. Sum <= 9 Final Carry = 1 (Answer InCorrect, add 0110)
3. Sum > 9 Final Carry = 0 (Answer InCorrect, add 0110)

1. 2 + 6

1. 3 + 7

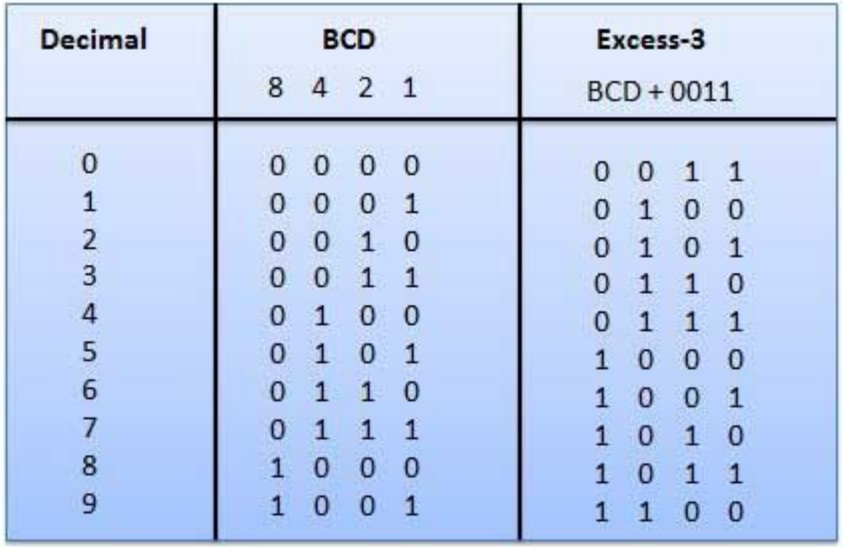
1. 8 + 9
2. 57 + 26

Excess-3 code

- The Excess-3 code is also called as XS-3 code.
- It is non-weighted code used to express decimal numbers.
- The Excess-3 code words are derived from the 8421 BCD code words adding $(0011)_2$ or (3)10 to each code word in 8421. The excess-3 codes are obtained as follows:

$$\begin{tikzcd}\text{Decimal Number} \arrow[r] & \text{8421 BCD} \arrow[r, "\text{Add}", "0011"'] & \text{Excess-3}\end{tikzcd}$$

- **Self Complementing Code [0-9, 1 - 8, 2-7 and so on]**



Decimal	BCD 8 4 2 1	Excess-3 BCD + 0011
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Decimal To Xs-3:

$$\begin{tikzcd}
45 \Rightarrow & 4 \arrow[d] & \mid & 5 \arrow[d] \\
& 0100 & \mid & 0101
\end{tikzcd}$$

0100 + 0011 => 0111 => 7

0101 + 0011 => 1000 => 8

=> 78

XS-3 Addition

Ex: $2_{10} + 5_{10}$

BCD - Xs-3

0010 + 0011 => 0101

0101 + 0011 => 1000

1101 (Excess of 6, so subtract 3) 1101 - 0011 =>1010

$(27)_{10} + (39)_{10}$

```
BCD              0010      0111
                 0011      1001
                  G2        G1
                  =          =
EXS3             0101      1010        If c=1, add 0011
                 0110      1100           c=0, sub 0011
                 ----      ----
            (-)  1100      0110 (+)
                 0011      0011
                 ----      ----
                 1001      1001
                 ----      ----
                  9         9
                 ------------
                                              27
                                              39
                                              --
                                (+3) <-       66
                                              --

                                              66
                                              33
                                              --
                                              99
```

ASCII Code

- The ASCII stands for American Standard Code for Information Interchange.
- The ASCII code is an alphanumeric code used for data communication in digital computers.
- The ASCII is a 7-bit code capable of representing 2^7 or 128 number of different characters.
- The ASCII code is made up of a three-bit group, which is followed by a four-bit code.

Parity

- To detect errors
- Single bit error can be detected
- Additional bit which is sent along with the input signal

Two Types:

Odd Parity

Even Parity

ODD => Number of ones should be odd along with parity

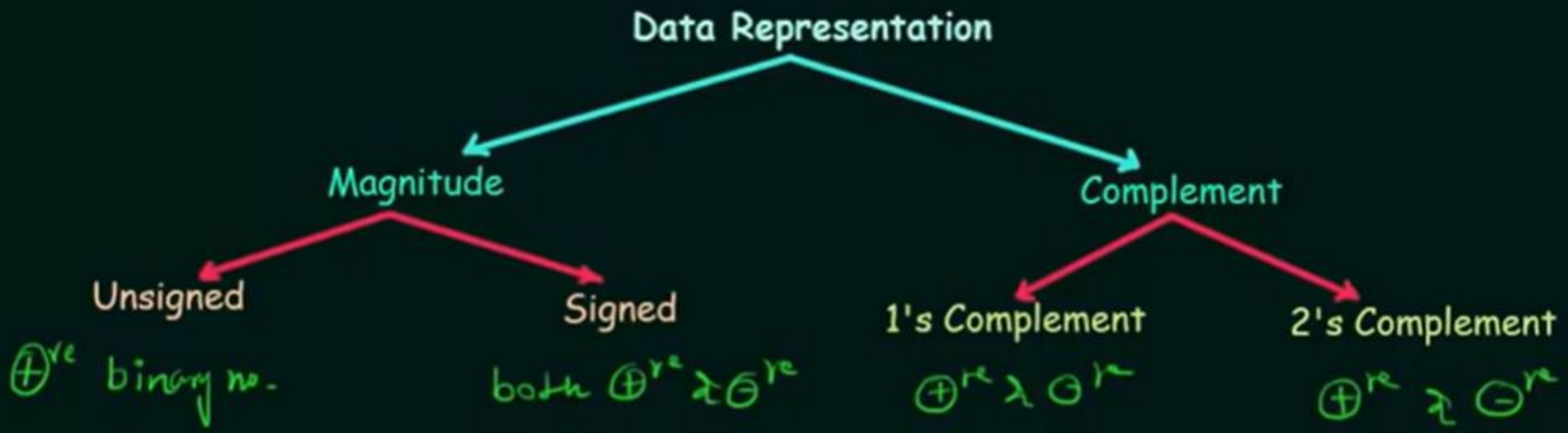
EVEN => No of ones should be even along with parity

For Ex: Data word: 1101

If want use ODD Parity then,   1101 0(Parity)

If want to use EVEN Parity then, 1101 1(Parity)

Data Representation using Signed Magnitude



Unsigned:

+6 => 0110

-6=> No representation

Signed:

+6=> 0 (SIGN) 110 (Magnitude)

-6=> 1   110

Range of Sign Magnitude: $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$ #n => no of variables

If n=4, range will be -7 to 7

1’s Complement

There are two types of complement of Binary number: 1’s complement and 2’s complement.

To get 1’s complement of a binary number, simply invert the given number.

To get 2’s complement of binary number is 1’s complement of given number plus 1 to the least significant bit (LSB).

Uses of 1’s Complement Binary Numbers:

Mainly in signed Binary number representation and various arithmetic operations for Binary numbers, e.g., additions, subtractions, etc.

Example: 110011 => 001100

Negative numbers in 1's complement

+6 => 0110

-6=> 1001

Problem: -0 & +0

0 => 0000 -0 => 1111

Range : $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$ #n => no of variables

If n=4, range will be -7 to 7

2’s Complement

10101110 => 01010001 + 1 => 01010010

Find 2’s complement of binary number 10001.001.

Uses of 2’s Complement Binary Numbers

There are various uses of 2’s complement of Binary numbers, mainly in signed Binary number representation and various arithmetic operations for Binary numbers, e.g., additions, subtractions, etc.

Since 2’s complement representation is unambiguous, so it very useful in Computer number representation.

+6 => 0110

-6=> 1001 + 1 => 1010

Range: -2^{n-1} to $+(2^{n-1}-1)$ #n => no of variables

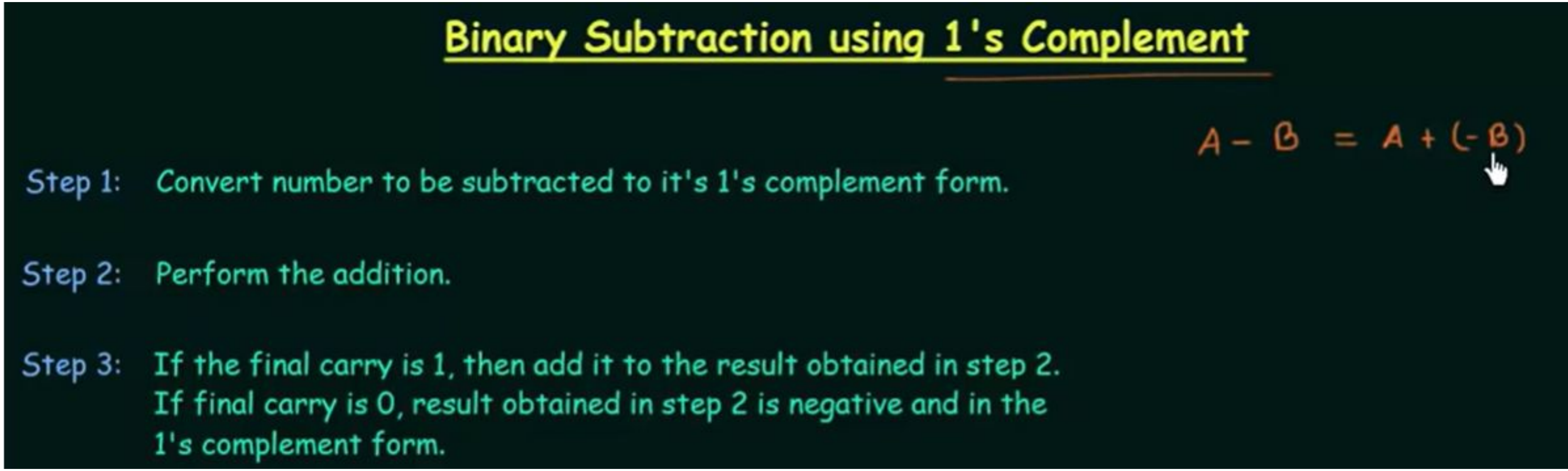
If n=4 (4 bits) => -8 to 7

In 2’s comp => MSB represents sign

1 means -ve

0 means +ve

Subtraction using 1’s complement:



Binary Subtraction using 1's Complement

$A - B = A + (-B)$

Step 1: Convert number to be subtracted to it's 1's complement form.

Step 2: Perform the addition.

Step 3: If the final carry is 1, then add it to the result obtained in step 2.
If final carry is 0, result obtained in step 2 is negative and in the
1's complement form.

Examples: Subtraction using 1's complement:

1100_2 - 0101_2 => ?

1100

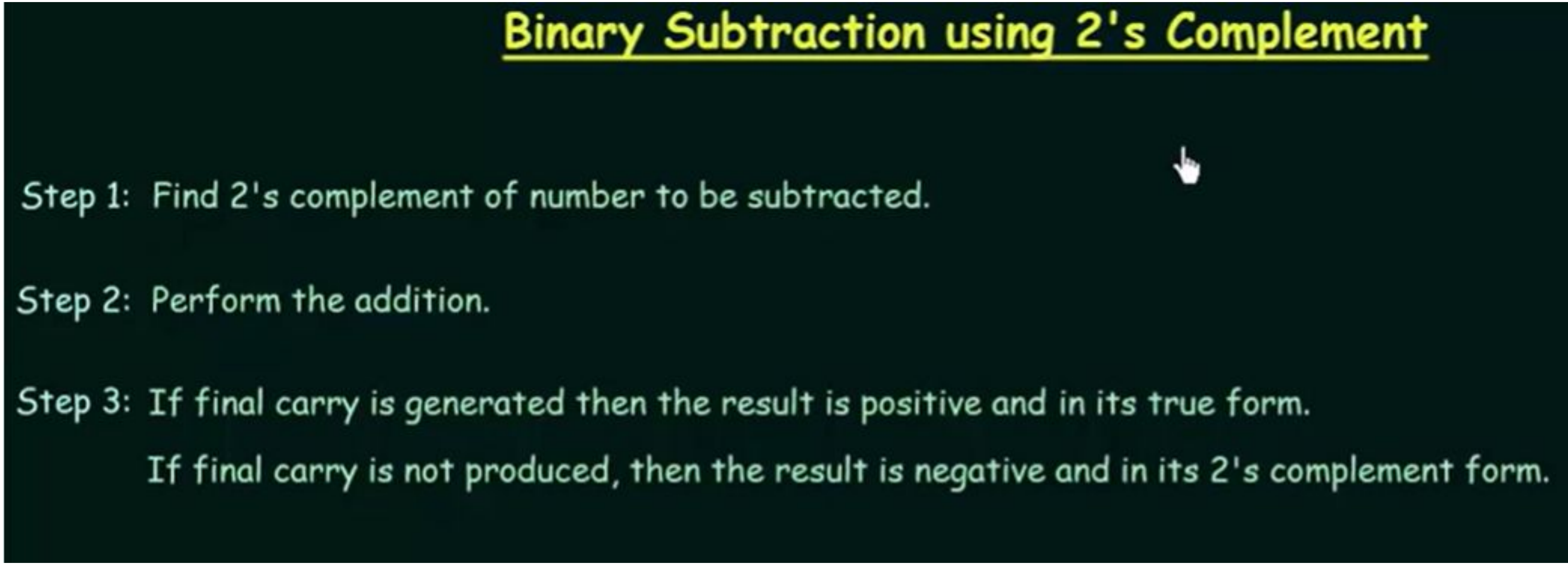
1010 => 1's comp

1 0110 => Addition

Add 1 => 0111

0101_2 - 1100_2 => ?

Subtraction using 2’s complement:



Binary Subtraction using 2's Complement

Step 1: Find 2's complement of number to be subtracted.

Step 2: Perform the addition.

Step 3: If final carry is generated then the result is positive and in its true form.

If final carry is not produced, then the result is negative and in its 2's complement form.

Examples: Subtraction using 2’s complement:

1001_2 - 0100_2 => ? (9-4)

2’s comp of 0100 =>

1011 + 1 => 1100

1001 (+) => 9

---

1 0101 => 5

Find: 0110_2 - 1011_2

Binary Multiplication

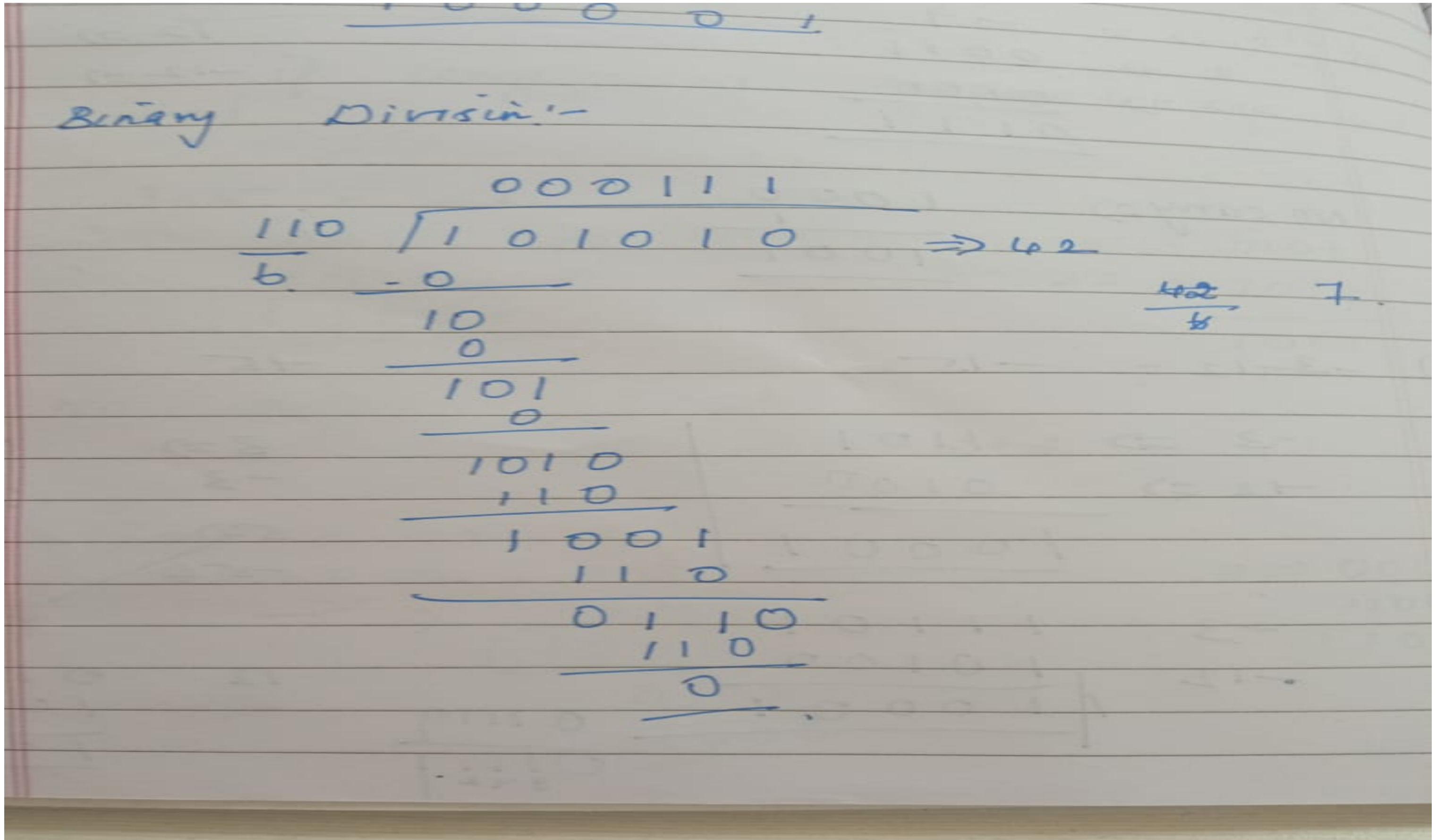
- $0 \times 0 = 0$
- $0 \times 1 = 0$
- $1 \times 0 = 0$
- $1 \times 1 = 1$

10 * 5 = 50

```
      1 0 1 0 *

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

Binary Division



```
        1  0  0  0  0  1

Binary Division:-

                 0 0 0 1 1 1
  110    ___________________
  ---   / 1 0 1 0 1 0          => 42
   6     -0
        -----                              42
          10                               --   7.
           0                                6
        -----
          101
            0
        -----
          1010
           110
        -------
           1001
            110
        ---------
             0110
              110
           -------
                0
              -----.
```

```
         001000.
      _____________
 111 /  111000
        0
        ______
        11
         0
        ______
        111
        111
        ______
          0000
```

Date
Page

```
          0001.111
1000 ) 1111
         0
        ____
        1111
        1000
        ____
        01110            ⇒ 0001.11
         1000
         _____
         01100
          1000
          _____
          01000
           1000
           _____
              0
```

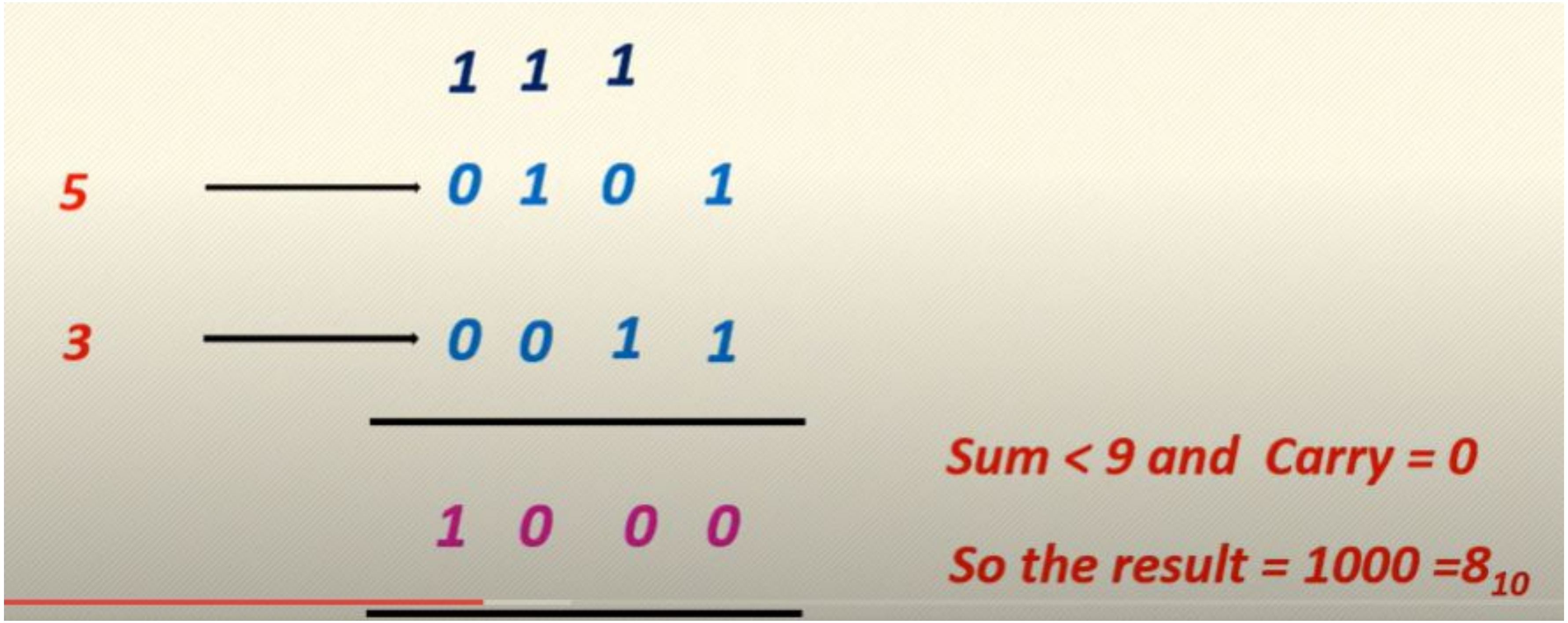
BCD Arithmetic

BCD Addition

BCD Subtraction

Binary Coded Decimal Addition

1. Sum < 9 and Carry = 0	Correct
2. Sum < 9 and Carry = 1	Incorrect and add 6 (0110)
3. Sum > 9 and Carry = 0	Incorrect and add 6 (0110)



$$\begin{array}{lcl} & & \;1\;1\;1 \\ 5 & \longrightarrow & 0\;1\;0\;1 \\ 3 & \longrightarrow & 0\;0\;1\;1 \\ \hline & & 1\;0\;0\;0 \end{array}$$

Sum < 9 and Carry = 0

So the result = 1000 $=8_{10}$

$5 \longrightarrow 0\ 1\ 0\ 1$

$8 \longrightarrow 1\ 0\ 0\ 0$

$$\begin{array}{rcccc}
 & 1 & 1 & 0 & 1 \\
+ & 0 & 1 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 & 1 \\
\hline
\end{array}$$

Sum > 9 and Carry = 0

Add 0110 (6)

So the result is 0001 0011

1. 235 + 852

```
                         1 1 1
235   ──→      0 0 1 0   0 0 1 1    0 1 0 1

852   ──→      1 0 0 0   0 1 0 1    0 0 1 0
______         ______________________________

               1 0 1 0   1 0 0 0    0 1 1 1
               0 1 1 0
               ______________________________

       0 0 0 1 0 0 0 0   1 0 0 0    0 1 1 1
```

Multi digit BCD Addition

2. 788+879

$$\begin{array}{rrrrrr}
 & & 1\,1\,1\,1 & & 1\,1\,1\,1 & \\
788 \longrightarrow & & 0\,1\,1\,1 & & 1\,0\,0\,0 & 1\,0\,0\,0 \\
879 \longrightarrow & & 1\,0\,0\,0 & & 0\,1\,1\,1 & 1\,0\,0\,1 \\
\hline
 & 1 & 0\,0\,0\,0 & 1 & 0\,0\,0\,0 & 1 \quad 0\,0\,0\,1 \\
 & & 0\,1\,1\,0 & & 0\,1\,1\,0 & 0\,1\,1\,0 \\
\hline
 & \underbrace{0\,0\,0\,1}_{1} & \underbrace{0\,1\,1\,0}_{6} & & \underbrace{0\,1\,1\,0}_{6} & \underbrace{0\,1\,1\,1}_{7}
\end{array}$$

Multi digit BCD Addition

3. 758+879

$$
\begin{array}{rcrrrr}
 & & & 1\,1\,1\,1 & 1\,1\,1\,1 & \\
758 & \longrightarrow & & 0\,1\,1\,1 & 0\,1\,0\,1 & 1\,0\,0\,0 \\
879 & \longrightarrow & & 1\,0\,0\,0 & 0\,1\,1\,1 & 1\,0\,0\,1 \\
\hline
1637 & & & 1\;0\,0\,0\,0 & 1\,1\,0\,1 & 1\;0\,0\,0\,1 \\
 & & & 0\,1\,1\,0 & 0\,1\,1\,0 & 0\,1\,1\,0 \\
\hline
 & & \underbrace{0\,0\,0\,1} & \underbrace{0\,1\,1\,0} & 1\;\underbrace{0\,0\,1\,1} & \underbrace{0\,1\,1\,1}
\end{array}
$$

BCD subtraction

Step 1 – convert to numbers into BCD

Step 2 – Take 9’s complement of subtrahend

Step 3– Add the complement number with minuend

Step 4 – If a four bit number is equal to or less than 9, then the result is valid BCD number.

Step 5 – If a four bit number is more than 9, then the result is invalid. Add 6 to get valid BCD number.

BCD subtraction

1. 695 - 238

9's comp of 238 $\longrightarrow$ 9 9 9 – 238 = 761

695	$\longrightarrow$	0 1 1 0	1 0 0 1	0 1 0 1
9's comp of 238	$\longrightarrow$	0 1 1 1	0 1 1 0	0 0 0 1

$$
\begin{array}{lccccc}
 & & 1\ 1\ \ & & & \ \ 1\ \\
695 \longrightarrow & & 0\ 1\ 1\ 0 & & 1\ 0\ 0\ 1 & 0\ 1\ 0\ 1 \\
\text{9's comp of 238} \longrightarrow & & 0\ 1\ 1\ 1 & & 0\ 1\ 1\ 0 & 0\ 0\ 0\ 1 \\
\hline
 & & 1\ 1\ 0\ 1 & & 1\ 1\ 1\ 1 & 0\ 1\ 1\ 0 \\
 & & (+)\,0\ 1\ 1\ 0 & & (+)\,0\ 1\ 1\ 0 & \\
\hline
 & \boxed{1} & 0\ 0\ 1\ 1 & \boxed{1} & 0\ 1\ 0\ 1\ \boxed{0} & 0\ 1\ 1\ 0 \\
 & & \ \ \ 1 & & & \\
 & & & & & \ \ \ 1 \\
\hline
 & & \underbrace{0\ 1\ 0\ 0}_{4} & & \underbrace{0\ 1\ 0\ 1}_{5} & \underbrace{0\ 1\ 1\ 1}_{7}
\end{array}
$$

Logic gates

- Building block of digital circuits.
- Make decisions based on a combination of digital signals coming from its inputs.
- Logic gates are based on Boolean algebra.
- False represents 0, and true represents 1.
- Seven basic logic gates: AND, OR, XOR, NOT, NAND, NOR, and XNOR.

GATES Boolean Algebra

AND => C (output) = $A . B$

OR => C (output) = $A + B$

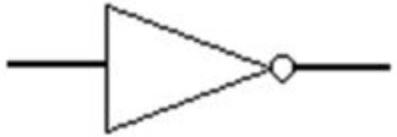
NAND => C (output) = $\overline{A. B}$

NOR => C (output) = $\overline{A+B}$

XOR => C (Output) = $\overline{A}.B + A\overline{B}$

XNOR => C (Output) = $\overline{AB} + AB$

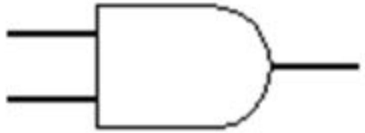
NOT gate



- Also called as the inverter gate.
- Has only one input.
- It just inverts the input.

Input 1	Result
1	0
0	1

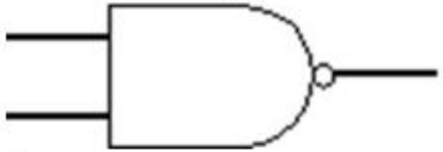
AND gate



- If both the inputs are true, the result is true else the result is false.

Input 1	Input 2	Result
1	1	1
1	0	0
0	1	0
0	0	0

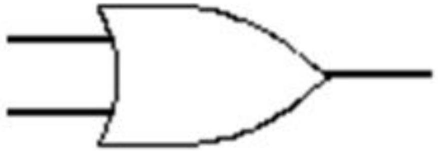
NAND gate



- AND gate followed by an inverter.
- The output is false if both inputs are true.

Input 1	Input 2	Result
1	1	0
1	0	1
0	1	1
0	0	1

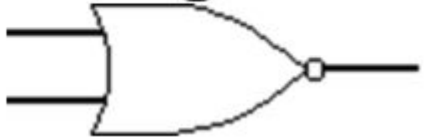
OR gate



- If any one of the input is true, the result is true else the result is false.

Input 1	Input 2	Result
1	1	1
1	0	1
0	1	1
0	0	0

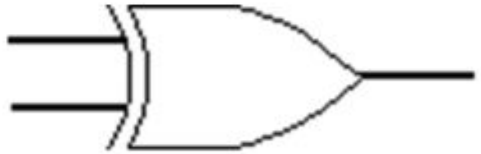
NOR gate



- NOR gate followed by an inverter.
- The output is true if both inputs are false.

Input 1	Input 2	Result
1	1	0
1	0	0
0	1	0
0	0	1

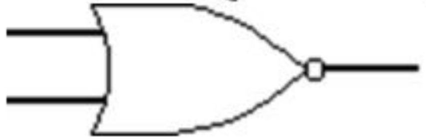
XOR gate (eXclusive OR)



- If either one of the input is true, the result is true else the result is false.
- If both the inputs are same, the result is false.

Input 1	Input 2	Result
1	1	0
1	0	1
0	1	1
0	0	0

eXclusive NOR (XNOR) gate



- XOR gate followed by an inverter.
- The output is true if both inputs are same.

Input 1	Input 2	Result
1	1	1
1	0	0
0	1	0
0	0	1