# Chapter 1

Subject Name: Digital Fundamentals

Subject code:-3130704

# Learning Outcomes

- **At the end of this chapter, you must be able to**

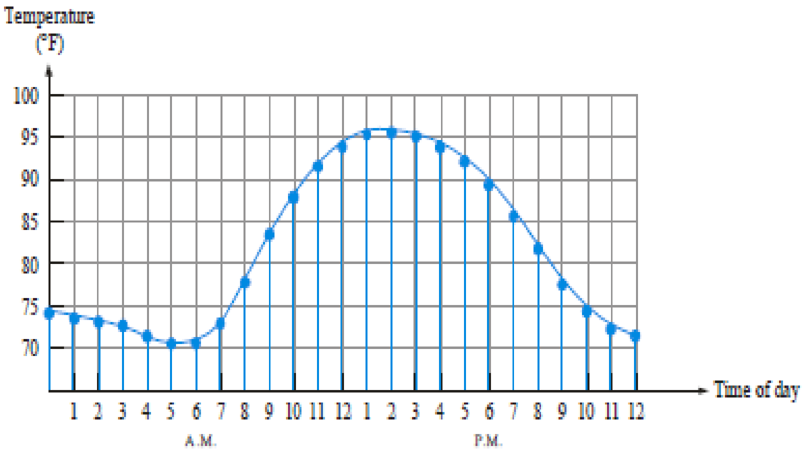
- Distinguish between analog and digital systems.
- Use of different types of basic gates
- Boolean algebraic  Laws
- How to reduce Boolean function
- Conversion of different number systems
- Methods of error detecting and correcting
- Comparison of logic families

# Signal:

It is physical quantity ,which contains information & a function of one or more independent variables.

-Two types of signal: analog & digital

1. Analog signal
2. Digital signals



# Analog signal versus digital signal

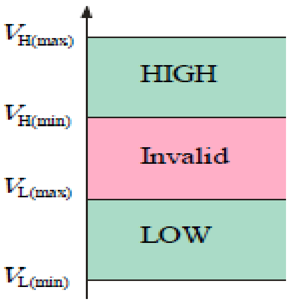
- Analog=continuous
- Digital=discrete
- Analog signals can have infinite no. of different signals. Example: Temperature, sound ,pressure, distant, sound.
- A digital Signal Not continuous signal. example: octal, hexadecimal.

**Advantage of digital signals**

- Digital signals can be processed & transmitted more efficiently and reliable.
- Can be store .
- Play back & further processing possible.
- noise effect is less.

# Binary Digits and Logic Levels

- Digital electronics uses circuits that have two states, which are represented by two different voltage levels called HIGH and LOW .The voltage  represent numbers in the binary systems.
- In binary a single number is called a bit(for binary digit).A bit can have the value of either a 0 or 1. depending on if the voltage is HIGH or LOW.



# Digital & Analog systems

**Digital systems**

Combination of device designed to manipulate logical information or physical quantity that are represented in digital form.

Example: digital calculator

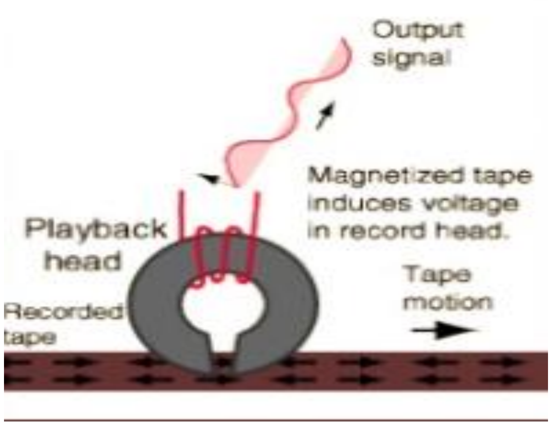
**Analog system**

Devices that manipulate physical quantities that are represented in analog form.

Example: magnetic tape recording, playback equipment

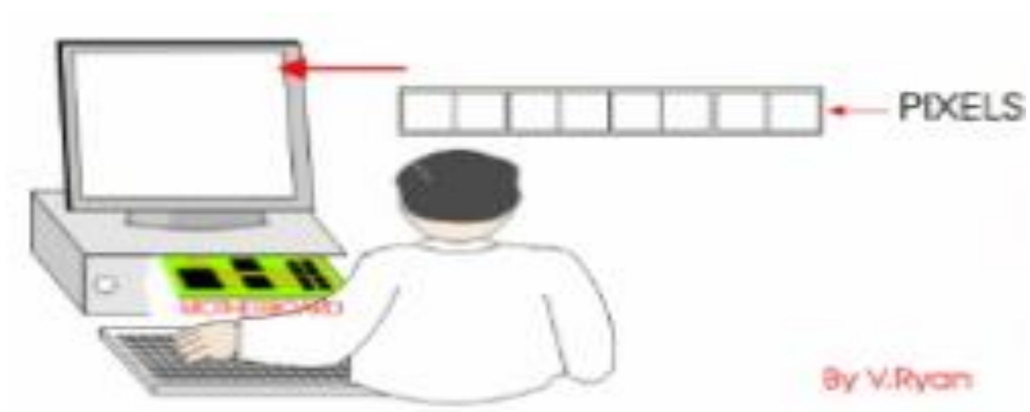
# Example 1 Tape

- During playback ,a magnetic material in the tape head is magnetized as the magnetic tape passes.
- Then ,the magnetic field penetrates a coil of wire is wrapped around it.
- Change in manganic field will induce a voltage in the coil. This induced voltage forms an electrical image of the signal which is recorded on the tape.



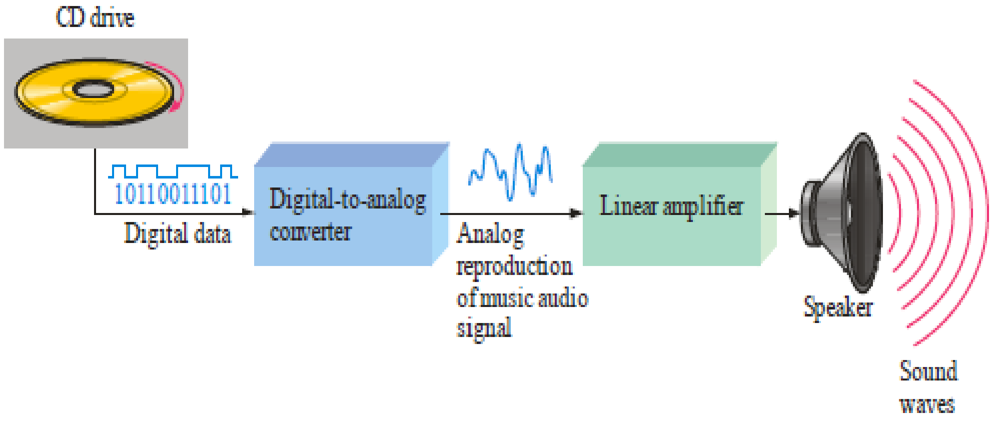
# Example2 computer

- All the stored and processed  data are in binary form. Why?
- Digital circuit/device only concerns about two operating states /logic levels.
- This  system allows computers to perform complex calculations very quickly and efficiently.



# Example 3 CD player

- CD player: Digital and analog parts co-exist together. Many systems use a mix of analog and digital electronics to take advantage of each technology . A typical CD player accepts digital data from the CD drive and convert it to an analog signal for amplification



# Advantage of digital Techniques

- Digital systems are easier to design.
- Information storage is easy.
- Accuracy & precision are easier to maintain throughout the system.
- They are more versatile.
- Digital circuit are less affected by noise
- Digital circuitry can be fabricated on IC chips

# Disadvantage of digital techniques

- The real world is analogue.
- Digital systems can be fragile.
- Processing digitized signals takes time.
- Digital circuits use more energy than analogue circuits & produce more heat.
- Digital circuits are made from analogue components –must make sure the digital behavior –must make sure the digital behavior is not a affected by the analogue.
- Digital circuit are sometimes more expensive

# Digital systems overcomes the drawback of analog systems

**When dealing with analogue inputs and outputs four steps must be followed:**

1. Convert the physical variable to an electrical signal(analogue)

2.Convert the electrical(analogue)signal into digital form- ADC (Analogue digital converter)

3.Process (operate on)the digital information

4. Convert the digital output back to real –world analogue form- DCA (Digital Analogue Converter)

# Gate

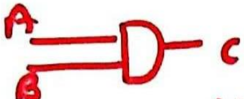
- The Most basic digital device are called gates.
- Gates got their name from their function of allowing or blocking (gating)the flow of digital information.
- A gate has one or more inputs and produces an output depending on the inputs
- A gate is called a combinational circuit
- Three most important gate are : AND,OR,NOT

# Logic gates

- Most basic logical unit of the digital system is gate circuit
- Types of gate circuit are as follows:

## ① AND Gate

- It has an o/p which is at logic level '0' and only goes 'HIGH' to logic level '1' when ALL of its i/p are at logic level '1'

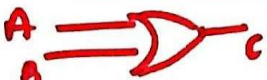



$C = A.B$

Truth table

| A | B | C |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## ② OR gate




$C = A + B$

truth table

| A | B | C |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## ③ NOT gate


$C = \bar{A}$ or $C = A'$

truth table

| A | C |
|---|---|
| 0 | 1 |
| 1 | 0 |

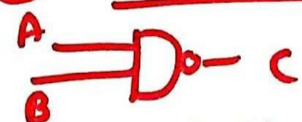
## ④ NOR gate


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

truth table

| A | B | C |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

## ⑤ NAND gate



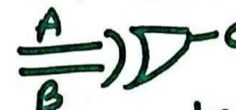

$C = (\overline{A.B})$

truth table

| A | B | C |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## ⑥ Ex-OR gate

- It has 1 state when one & only one of its two i/p assumes a logic 1 state & has 0 state when all its i/p are same.

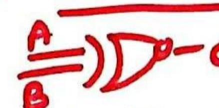



$C = A \oplus B$

truth table

| A | B | C |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## ⑦ Ex-NOR gate




$C = A \odot B$

truth table

| A | B | C |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Scanned with CamScanner

# Boolean Algebra

## AND laws

1. $A.0 = 0$ (Null law)
2. $A.1 = A$ (Identity law)
3. $A.A = A$
4. $A.\overline{A} = 0$

## OR laws

1. $A+0 = A$ (Null law)
2. $A+1 = 1$ (Identity law)
3. $A+A = A$
4. $A+\overline{A} = 1$

## Distributive laws

1. $A(B+C) = AB+AC$
2. $A+BC = (A+B)(A+C)$

## Redundant Literal laws

1. $A+A\overline{B} = A+B$
2. $A(\overline{A}+B) = AB$

## Commutative laws

1. $A+B = B+A$
2. $A.B = B.A$

## Associate laws

1. $(A+B)+C = A+(B+C)$
2. $(A.B).C = A(B.C)$

## Idempotent laws

1. $A.A = A$
2. $A+A = A$

## Absorption laws

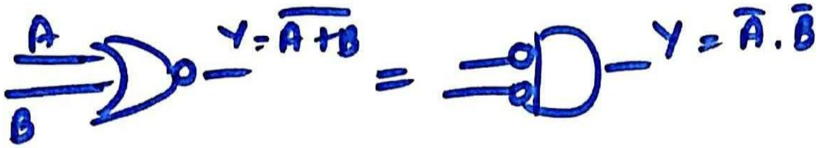
1. $A+AB = A$
2. $A(A+B) = A$

CS Scanned with CamScanner

# De- Morgen's Theorem

① Law 1:- $\overline{A+B} = \bar{A}.\bar{B}$

→ Complement of sum of variable is equal to the product of their individual complement



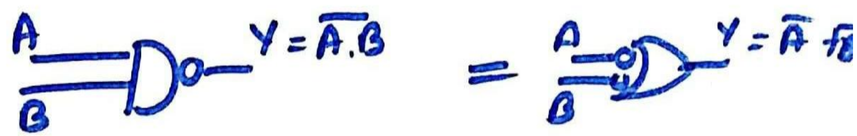

Truth table

| A | B | $\overline{A+B}$ | $\bar{A}$ | $\bar{B}$ | $\bar{A}.\bar{B}$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |

↑ L.H.S ↑ R.H.S

② Law 2:- $\overline{AB} = \bar{A}+\bar{B}$

→ Complement of a product of variable is equal to the sum of their individual complement.




Truth Table

| A | B | $\overline{A.B}$ | $\bar{A}$ | $\bar{B}$ | $\bar{A}+\bar{B}$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |

↑ L.H.S ↑ R.H.S

Scanned with CamScanner

# Universal Gate

## NAND as universal gate :-

(1) NOT using NAND



NAND

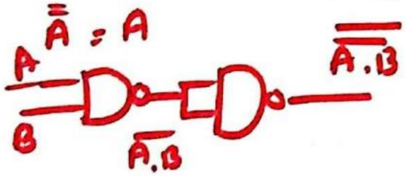
| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

→ NAND

(2) AND using NAND

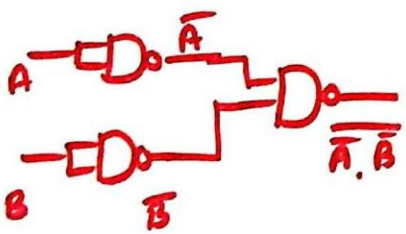
$Y = A.B = \overline{\overline{A.B}}$

$\bar{\bar{A}} = A$



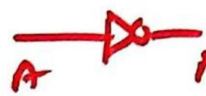

(3) OR using NAND

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



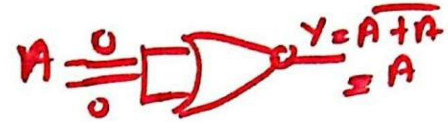

## NOR AS Universal NOR :-

(1) NOT using NOR



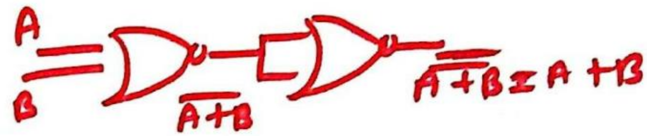

NOR

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |



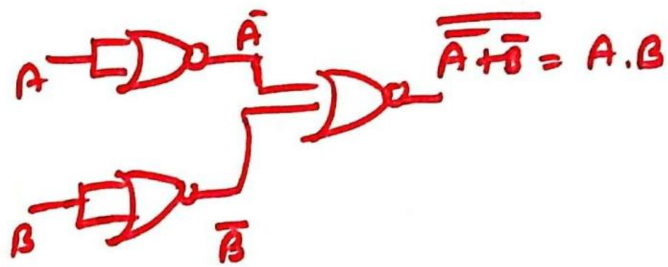

(2) OR gate using NOR

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




(3) AND gate using NOR

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




CS Scanned with CamScanner

# Reducing Boolean Expression

(1) $f = A + B[AC + (B+\bar{C})D]$

Ans $f = A + B[AC + BD + \bar{C}D]$ ← distributive

$= A + BAC + BBD + B\bar{C}D$ ← distributive

$= A + ABC + BD + B\bar{C}D$ ← $(A.A = A)$

$= A(1+BC) + BD(1+\bar{C})$ ← $A+1 = 1$

$\boxed{f = A + BD}$

(2) $f = A[B + \bar{C}(\overline{A\bar{B} + A\bar{C}})]$ (GTU 3/12/19)

Ans $f = A[B + \bar{C}(\overline{A\bar{B}} . \overline{A\bar{C}})]$ ← De-Morgan's

$= A[B + \bar{C}(\bar{A} + \bar{B})(\bar{A} + C)]$ ← De-Morgan's

$= A[B + \bar{C}(\bar{A}\bar{A} + \bar{A}C + \bar{B}\bar{A} + \bar{B}C)]$ ← distributive

$= A[B + \bar{C}\bar{A} + \bar{C}\bar{A}C + \bar{C}\bar{B}\bar{A} + \bar{C}\bar{B}C]$ ← distributive

$= A[B + \bar{C}\bar{A} + 0 + \bar{C}\bar{B}\bar{A} + 0]$ ← $A.A' = 0$

$= AB + A\bar{C}\bar{A} + A\bar{C}\bar{B}\bar{A}$ ← distributive

$= AB + 0 + 0$ ← $A.A' = 0$

$= AB$

(3) Reduce the expression $\cancel{A + B[AC + (B+\bar{C})D]}$ $(B+BC)(B+\bar{B}C)(B+D)$

Ans = B

Ans $F = A + B[AC + (B+\bar{C})D]$

$= A + B(AC + BD + \bar{C}D]$ ← distributive

$= A + BAC + BBD + B\bar{C}D$ ← distributive

$= A + ABC + BD + BD\bar{C}$ ← In order

$= A(1 + BC) + BD(1+\bar{C})$ ← factor

$= A(1) + BD.1$ ← $1+\bar{C} = 1$

$= A + BD$

(4) Reduce the expression

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

Ans = 0

(5) Show that,

$AB + A\bar{B}C + B\bar{C} = AC + B\bar{C}$

CS Scanned with CamScanner

(1) $(B + BC)(B + \bar{B}C)(B + D)$

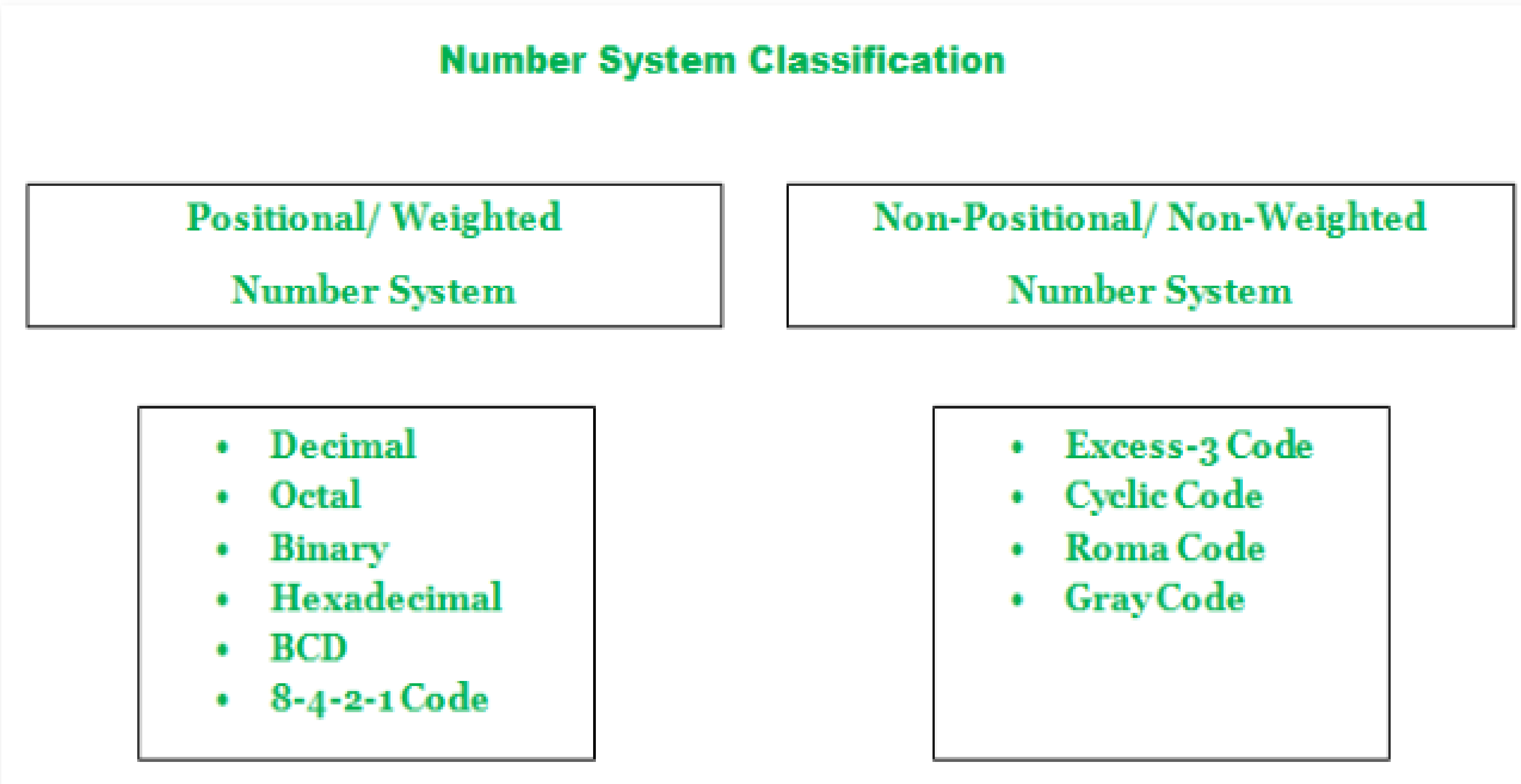
(2) $\left(\overline{A + \overline{BC}}\right)(A\bar{B} + ABC)$

(3) Show that $AB + A\bar{B}C + B\bar{C} = B\bar{C} + AC$

(2) Show that $A\bar{B}C + B + B\bar{D} + AB\bar{D} + \bar{A}C$

$= B + C$

# Classification of Number systems



# Non-positional Number Systems

**Characteristics**

- Use symbols such as I for 1, II for 2, III for 3, IIII for 4, IIIII for 5, etc.
- Each symbol represents the same value regardless of its position in the number.
- The symbols are simply added to find out the value of a particular number.

## Difficulty

- It is difficult to perform arithmetic with such a number system.

# Positional Number Systems

**Characteristics**

- Use only a few symbols called digits
- These symbols represent different values depending on the position they occupy in the number.
- The value of each digit is determined by:

  1. The digit itself
  2. The position of the digit in the number
  3. The base of the number system

- **Number system**: It is set of values used to represent a quantity.
- **Radix/Base:** No. of value that a digit can have is equal to the systems.
- **Weight :**each position represent a different multiple of base this multiplier called weight.
- **MSD:** Leftmost digit having the highest Weight  known as most significant digit
- **LSD:** Rightmost digit having the highest Weight  known as least significant digit

Weight

$$
\begin{array}{llrr}
125_{10} \Rightarrow & 5 \times 10^{0} & = & 5 \\
 & 2 \times 10^{1} & = & 20 \\
 & 1 \times 10^{2} & = & \underline{100} \\
 & & & 125
\end{array}
$$

Base

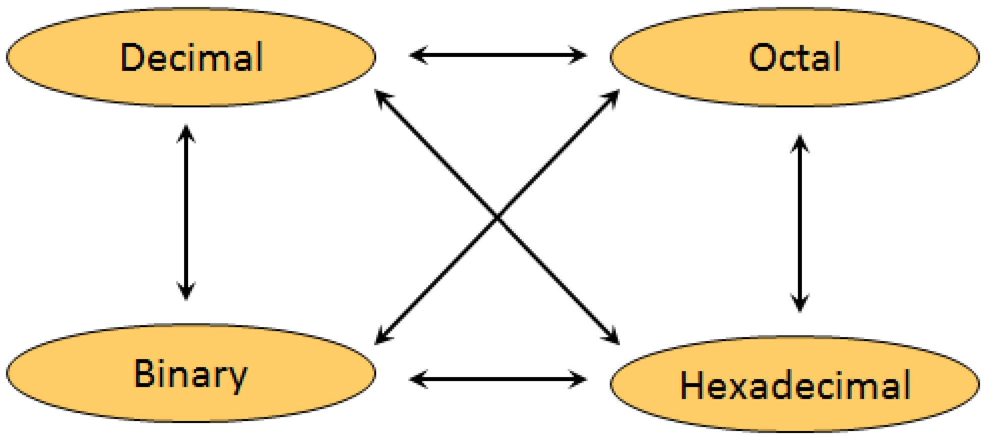
# Common number systems

| System | Base | Symbols | Used by humans? | Used in computers? |
|---|---|---|---|---|
| Decimal | 10 | 0, 1, … 9 | Yes | No |
| Binary | 2 | 0, 1 | No | Yes |
| Octal | 8 | 0, 1, … 7 | No | No |
| Hexa-decimal | 16 | 0, 1, … 9, A, B, … F | No | No |

- The maximum value of a single digit is always equal to one less than the value of the base.

# Conversion among Bases

- Possibilities




- Example

$$25_{10} = 11001_2 = 31_8 = 19_{16}$$

Base

# Binary Number System

**Characteristics**

- A positional number system
- Has only 2 symbols or digits (0 and 1). Hence its

base = 2.

- The maximum value of a single digit is 1 (one less than the value of the base).
- Each position of a digit represents a specific power of the base (2)
- This number system is used in computers

# Decimal To Binary

- **Successive division for inter part:**

1.Divide the integer part of given decimal no. by the base & note down remainder.

2. Continue to divide the quotient by base until there is nothing left. Note remainder from each step.

3.List the remainder in reverse order from bottom to top .

$125_{10} = ?_2$

| | | |
|---|---|---|
| 2 | 125 | 1 |
| 2 | 62 | 0 |
| 2 | 31 | 1 |
| 2 | 15 | 1 |
| 2 | 7 | 1 |
| 2 | 3 | 1 |
| 2 | 1 | 1 |
| | 0 | |

$125_{10} = 1111101_2$

- **Successive multiplication for fractional part**

1. Multiply given no. by base
2. Note down carry generated in this multiplication as MSD
3. Multiply only fractional no. of the product in step 2 by the base ,& note down carry as the next bit to MSD
4. Repeat Step 2 & 3 Upto The End.

$0.6875_{10} = ?_2$

| | *integer* | | *fraction* |
|---|---|---|---|
| $0.6875 \times 2 = 1.3750$ | 1 | + | 0.3750 |
| $0.3750 \times 2 = 0.7500$ | 0 | + | 0.7500 |
| $0.7500 \times 2 = 1.5000$ | 1 | + | 0.5000 |
| $0.5000 \times 2 = 1.0000$ | 1 | + | 0.0000 |

$0.6875_{10} = 0.1011_2$

# Exercise

1. $(163.875)_{10}$ =
2. $(52)_{10}$ =
3. $(0.75)_{10}$ =
4. $(105.15)_{10}$ =

# Answer

1. 10100011
2. 110100
3. 0.11
4. 1101001.001001

# Binary to Decimal

1. Write down weight corresponding to different position.
2. Multiply each digit in the given no. with corresponding weight to obtain product no.
3. Add all the product no. to get decimal equivalent.

1 0 1 0 1 1

$$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$32 + 0 + 8 + 0 + 2 + 1$$

$$43_{10}$$

$$101011_2 = 43_{10}$$

1 1 . 1 1

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

$$2 + 1 + 0.5 + 0.25$$

$$3.75_{10}$$

# Exercise

1. $(10101)_2$
2. $(11011.101)_2$
3. $(1001011)_2$
4. $(1011.01)_2$

# Answer

1. $21_{10}$
2. $27.625_{10}$
3. $75_{10}$
4. $11.25_{10}$

# Decimal to octal conversion

1. Divide by 8
2. Keep track of remainder

$125_{10} = ?_8$

| | | |
|---|---|---|
| 8 | 125 | 5 |
| 8 | 15 | 7 |
| 8 | 1 | 1 |
| | 0 | |

$0.6875_{10} = ?_8$

| | *integer* | | *fraction* |
|---|---|---|---|
| $0.6875 \times 8 = 5.5000$ | 5 | + | 0.5000 |
| $0.5000 \times 8 = 4.0000$ | 4 | + | 0.0000 |

$125_{10} = 175_8$

$0.6875_{10} = 0.54_8$

# exercise

1. $(3000.45)_{10}$ =
2. $(378.93)_{10}$ =
3. $(5497)_{10}$ =
4. $(3025)_{10}$ =

# Answer

1. 5870.3463
2. 572
3. 12571
4. 5721

# Octal to Decimal

1. Multiply each bit by $8^n$, where n is the weight of the bit
2. The weight is the position of the bit, starting from 0 on the right.
3. Add the result.

$$4 \quad 3 \quad . \quad 2 \quad 5$$

$$4 \times 8^1 + 3 \times 8^0 + 2 \times 8^{-1} + 5 \times 8^{-2}$$

$$32 + 3 + 0.25 + 0.0781$$

$$35.3281_{10}$$

$$43.25_8 = 35.3281_{10}$$

# Exercise

1. $(314)_8$
2. $(4057.06)_8$
3. $(5721)_8$
4. $(630.4)_8$

# Answer

1. $204_{10}$
2. $2095.0937_{10}$
3. $3025_{10}$
4. $408.5_{10}$

# Decimal to Hexadecimal

1. Divide by 16
2. Keep track of remainder

$1234_{10} = ?_{16}$

| | | |
|---|---|---|
| 16 | 1234 | 2 |
| 16 | 77 | 13=D |
| 16 | 4 | 4 |
| | 0 | |

$1234_{10} = 4D2_{16}$

# Exercise

1. $(2003.31)_{10} =$
2. $(2598.675)_{10} =$
3. $(49056)_{10} =$
4. $(46687)_{10} =$

# Answer

1. 7D3.4F5C2
2. A26.ACCC
3. BFA0
4. B65F

# Hexadecimal To Decimal

1. Multiply each bit by $16^n$, where n is the weight of the bit
2. The weight is the position of the bit, starting from 0 on the right.
3. Add the result.

A B C

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

$$10 \times 16^2 + 11 \times 16^1 + 12 \times 16^0$$

$$2560 + 176 + 12$$

$$2748_{10}$$

$$ABC_{16} = 2748_{10}$$

# Exercise

1. $(4CB.2)_{16}$
2. $(A0F9.0EB)_{16}$
3. $(5C7)_{16}$
4. $(B65F)_{16}$

# Answer

1. $1224.125_{10}$
2. $41209.0572_{10}$
3. $1479_{10}$
4. $46687_{10}$

# Octal to Binary

- Convert each octal digit to a 3-bit equivalent binary representation.

$705_8 = ?_2$

$$\begin{array}{ccc} 7 & 0 & 5 \\ \downarrow & \downarrow & \downarrow \\ 111 & 000 & 101 \end{array}$$

$705_8 = 111000101_2$

# Exercise

1. $(364)_8$
2. $(5721)_8$
3. $(12571)_8$
4. $(26153.7406)_8$

# Answer

1. $011110100_2$
2. $101111010001_2$
3. $001010101111001_2$
4. $1011000110101 1.11110000110_2$

# Binary to Octal

- Group bits in 3 starting from LSB

$1011010111_2 = ?_8$

$$
\begin{array}{cccc}
001 & 011 & 010 & 111 \\
\downarrow & \downarrow & \downarrow & \downarrow \\
1 & 3 & 2 & 7
\end{array}
$$

$1011010111_2 = 1327_8$

# Exercise

1. $(11010010)_2$
2. $(110101.101010)_2$
3. $(10101111001.0111)_2$
4. $(1100000110.1101)_2$

# Answer

1. $322_8$
2. $65.52_8$
3. $2571.34_8$
4. $306.D_8$

# Hexadecimal to Binary

- Convert each hexadecimal digit to a 4 bits equivalent binary representation.

## Hexa-Decimal to Binary

| Hexa-Decimal | Binary | Hexa-Decimal | Binary |
|---|---|---|---|
| 0 | 0000 | 8 | 1000 |
| 1 | 0001 | 9 | 1001 |
| 2 | 0010 | A | 1010 |
| 3 | 0011 | B | 1011 |
| 4 | 0100 | C | 1100 |
| 5 | 0101 | D | 1101 |
| 6 | 0110 | E | 1110 |
| 7 | 0111 | F | 1111 |

$10AF_{16} = ?_2$

| 1 | 0 | A | F |
|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ |
| 0001 | 0000 | 1010 | 1111 |

$$10AF_{16} = 1000010101111_2$$

# Exercise

- $(FA8)_{16} = (\quad)_2$
- $(9AC3)_{16} = (\quad)_2$
- $(1A74D)_{16} = (\quad)_2$
- $(1AC.9A)_{16} = (\quad)_2$
- $(ABC.5AC)_{16} = (\quad)_2$

# Binary to Hexadecimal

- Group bits in 4 starting from LSB

$1011010111_2 = ?_{16}$

$$\begin{array}{ccc} 0010 & 1101 & 0111 \\ \downarrow & \downarrow & \downarrow \\ 2 & D & 7 \end{array}$$

$1011010111_2 = 2D7_{16}$

# Exercise

- $(11011)_2 = (\quad)_{16}$
- $(101101)_2 = (\quad)_{16}$
- $(11101111)_2 = (\quad)_{16}$
- $(110.011)_2 = (\quad)_{16}$
- $(1001.0010)_2 = (\quad)_{16}$

# Octal To Hexadecimal

- Convert octal to binary
- Regroup bits in 4 from LSB
- Convert Binary To hexadecimal

$1076_8 = ?_{16}$

| 1 | 0 | 7 | 6 |
|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ |
| 001 | 000 | 111 | 110 |

| 0010 | 0011 | 1110 |
|---|---|---|
| ↓ | ↓ | ↓ |
| 2 | 3 | E |

# Exercise

- $(463)_8 = (\quad)_{16}$
- $(2056)_8 = (\quad)_{16}$
- $(2057.64)_8 = (\quad)_{16}$
- $(6543.04)_8 = (\quad)_{16}$
- $(7476.47)_8 = (\quad)_{16}$

# Hexadecimal to Octal

- Convert hexadecimal to binary
- Regroup bits in 3 from LSB
- Convert Binary To Octal

$1F0C_{16} = ?_8$

| | | | | | |
|---|---|---|---|---|---|
| | 1 | F | | 0 | C |
| | ↓ | ↓ | | ↓ | ↓ |
| | 0001 | 1111 | | 0000 | 1100 |
| 000 | 001 | 111 | 100 | 001 | 100 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 0 | 1 | 7 | 4 | 1 | 4 |

$1F0C_{16} = 17414_8$

# Exercise

$(FA8)_{16} = (\quad)_8$

$(9AC3)_{16} = (\quad)_8$

$(1A74D)_{16} = (\quad)_8$

$(1AC.9A)_{16} = (\quad)_8$

$(ABC.5AC)_{16} = (\quad)_8$

# Sign binary Number

- Two ways of representing signed numbers:
  1. sign magnitude form
  2. complement form
- Most of computer use complement form for negative number notation.
- 1’s and 2’s complement are two different method this type.

# 1's complement

- It is obtained by subtracting each digit of that binary no. from 1.
- Or, change 1's to 0's  and 0's to 1's. in binary
- 
  - Example

|   |   |   |   |   |
|---|---|---|---|---|
|   | 1 | 1 | 1 | 1 |
| - | 1 | 1 | 0 | 1 |
|   | 0 | 0 | 1 | 0 |

(1's complement of 1101)

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
|   | 1 | 1 | 1 | . | 1 | 1 |
| - | 1 | 0 | 1 | . | 0 | 1 |
|   | 0 | 1 | 0 | . | 1 | 0 |

(1's complement of 101.01)

# 2's complement

- It is obtained by adding 1 to 1's complement.
- **Or**, copy all zeros ,working from LSB toward the MSB , until the 1st 1 is reached, copy that 1 & flip all the remaining bits.
    - Example

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

(2's complement of 1100)

$$\begin{array}{rcccccc} & 1 & 1 & 1 & . & 1 & 1 \\ - & 1 & 0 & 1 & . & 0 & 1 \\ \hline & 0 & 1 & 0 & . & 1 & 0 \\ + & & & & & & 1 \\ \hline & 0 & 1 & 0 & . & 1 & 1 \end{array}$$

(2's complement of 101.01)

# 9's complement

- It's obtained by subtracting each digit of that decimal no. from 9

  - Example

$$\begin{array}{r} 9\ 9\ 9\ 9 \\ -\ 3\ 4\ 6\ 5 \\ \hline 6\ 5\ 3\ 4 \end{array}$$

(9's complement of 3465)

$$\begin{array}{r} 9\ 9\ 9\ .\ 9\ 9 \\ -\ 7\ 8\ 2\ .\ 5\ 4 \\ \hline 2\ 1\ 7\ .\ 4\ 5 \end{array}$$

(9's complement of 782.54)

# 10’s complement

- It’s obtained by adding 1’s to 9’s complement.

$$
\begin{array}{rccccccc}
 & 9 & 9 & 9 & . & 9 & 9 \\
- & 7 & 8 & 2 & . & 5 & 4 \\
\hline
 & 2 & 1 & 7 & . & 4 & 5 \\
+ & & & & & & 1 \\
\hline
 & 2 & 1 & 7 & . & 4 & 6
\end{array}
$$

(10’s complement of 782.54)

# Sign number representation

- If no. is positive ,the magnitude is represented in its true binary form and a sign bit 0 is placed in front of the MSB
- If no. is negative ,magnitude is represented in its 2's complement form and sign bit 1 is placed in front of MSB
    - Express -65.5 in 12 bit 2's complement form.

| | | |
|---|---|---|
| 2 | 65 | 1 |
| 2 | 32 | 0 |
| 2 | 16 | 0 |
| 2 | 8 | 0 |
| 2 | 4 | 0 |
| 2 | 2 | 0 |
| 2 | 1 | 1 |
| | 0 | |

$0.5 \times 2 = 1.0$

So, result in 12-bit binary is as follows:

$65.5_{10} = 01000001.1000_2$

For negative number, we have to convert this into 2's complement form

$-65.5_{10} = 10111110.1000_2$

# Exercise

1. Express -45 in 8 bit 2’s complement form.
2. Express -73.25 in 12 bits 2’s complement form.

# Answer

1. 11010011
2. 10110110.0100

# Subtraction using complement form

## 1. Using 9'S Complement

- obtain 9's complement of subtrahend

-Add the result to minuend and call it intermediate result.

If carry is generated then answer is positive and add the carry to LSD

-If There is no Carry Then Answer Is Negative And Take 9's Complement Of Intermediate Result And Place Negative Sign To The Result

1) 745.81 – 436.62

$$\begin{array}{r} 745.81 \\ -\ 436.62 \\ \hline 309.19 \end{array} \xrightarrow{\text{9's complement}} \begin{array}{r} 745.81 \\ +\ 563.37 \\ \hline 1\ 309.18 \\ +\ \ \ 1 \\ \hline 309.19 \end{array}$$

# Example

2) 436.62 - 745.81

$$
\begin{array}{r}
436.62 \\
-\ 745.81 \\
\hline
-\ 309.19
\end{array}
\xrightarrow{\text{9's complement}}
\begin{array}{r}
436.6 \\
+\ 254.1 \\
\hline
690.8
\end{array}
$$

$$
690.8 \xrightarrow{\text{9's complement}} -\ 309.1
$$

As carry is not generated, so take 9's complement of the intermediate result and add ' – ' sign to the result

## 2. Using 10'S Complement

- obtain 10's complement of subtrahend
- Add the result to minuend.
- If carry is generated then ignore it and result itself is answer.
- If There is no Carry Then Answer Is Negative And Take 10's Complement Of Result And Place Negative Sign To The Result .

### Example

1) 745.81 – 436.62

$$\begin{array}{r} 745.81 \\ -\ 436.62 \\ \hline 309.19 \end{array} \quad \xrightarrow{\text{10's complement}} \quad \begin{array}{r} 745.81 \\ +\ 563.38 \\ \hline 1\ 309.19 \end{array}$$

Ignore the carry

# Example

2) 436.62 - 745.81

$$
\begin{array}{rr}
 & 436.62 \\
- & 745.81 \\
\hline
- & 309.19
\end{array}
\xrightarrow{\text{10's complement}}
\begin{array}{rr}
 & 436.62 \\
+ & 254.19 \\
\hline
 & 690.81
\end{array}
$$

10's complement

$$- \; 309.19$$

As carry is not generated, so take 10's complement of the intermediate result and add ' – ' sign to the result

# Binary addition

- Rules for binary addition

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | | 1 | 1 | |
| | 1 | 1 | 0 | 1 | . | 1 | 0 | 1 |
| + | 0 | 1 | 1 | 1 | . | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | . | 0 | 0 | 0 |

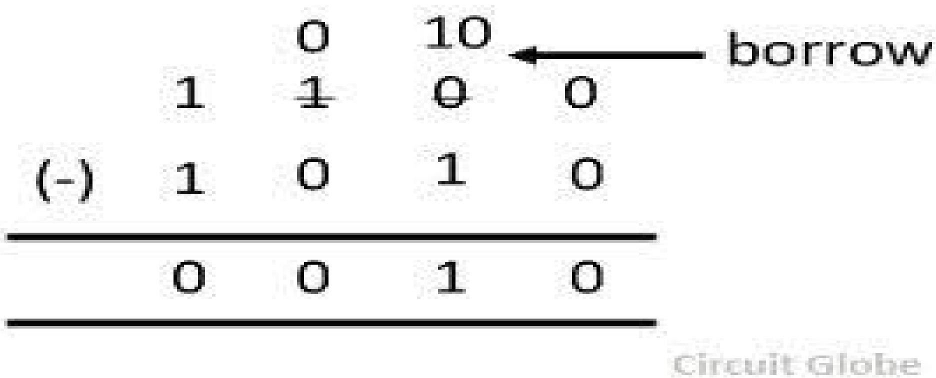
0 + 0 = 0

0 + 1 = 1

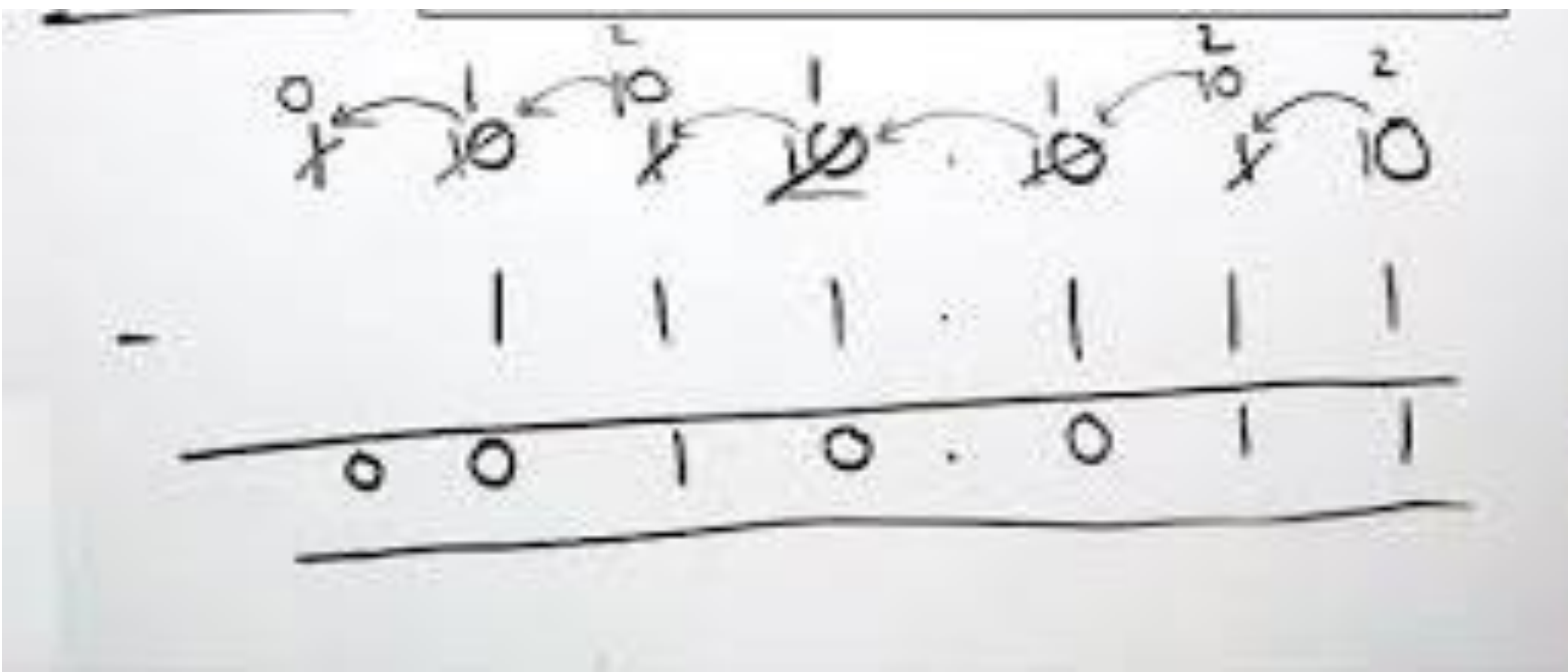
1 + 0 = 1

1 + 1 = 10 i.e. 0 with a carry of 1

# Binary Subtraction

- 


0 – 0 = 0
1 – 1 = 0
1 – 0 = 1
0 – 1 = 1, with a borrow 1



# 3.Subtraction using 1’s complement

-obtain 1’s complement of subtrahend.

-Add the result to  minuend and call it intermediate result.

-If carry is generated then answer is positive and add the carry to LSD

-If There Is No Carry Then Answer Is Negative And Take 1’s Complement Of Intermediate Result  And Place Negative Sign To The Result.

## Example

1) 68.75 – 27.50

$$
\begin{array}{rcr}
68.75 & & 01000100.1100 \\
-\ 27.50 & \xrightarrow{\text{1's complement}} & +\ 11100100.0111 \\
\hline
+\ 41.25 & & 100101001.0011 \\
 & & +1 \\
\hline
 & & 00101001.0100
\end{array}
$$

# Example

2) 43.25 - 89.75

$$
\begin{array}{r}
43.25 \\
-\ 89.75 \\
\hline
-\ 46.50
\end{array}
\qquad
\xrightarrow{\text{1's complement}}
\qquad
\begin{array}{r}
00101011.0100 \\
+\ 10100110.0011 \\
\hline
11010001.0111
\end{array}
$$

$$
11010001.0111 \xrightarrow{\text{1's complement}} 00101110.1000
$$

As carry is not generated, so take 1's complement of the intermediate result and add ' – ' sign to the result

# 3.Subtraction using 1’s complement

-obtain 2’s complement of subtrahend.

-Add the result to  minuend.

-If carry is generated then ignore it and result is answer.

-If There Is No Carry Then Answer Is Negative And Take 2’s Complement Of Result  And Place Negative Sign To The Result.

## Example

1) 68.75 – 27.50

| | | | |
|---|---|---|---|
| | 68.75 | | 01000100.1100 |
| - | 27.50 | 2’s complement → | + 11100100.1000 |
| + | 41.25 | | $\boxed{1}$ 00101001.0100 |
| | | Ignore Carry bit | |
| | | | 00101001.0100 |

# Example

2) 43.25 - 89.75

| | | | |
|---|---|---|---|
| 43.25 | | | 00101011.0100 |
| - 89.75 | $\xrightarrow{\text{2's complement}}$ | + | 10100110.0100 |
| - 46.50 | | | 11010001.1000 |
| | 2's complement | | 00101110.1000 |

As carry is not generated, so take 2's complement of the intermediate result and add ' – ' sign to the result

# Binary code

- Group of symbol called code.

**Advantages:**

1.They are suitable For computer application.

2.Analysis and designing of digital circuit become easy .if we use binary code.

3. As only 1 & 0 are being used ,implementation of binary code become easy.

# CLASSIFICATION OF BINARY CODE

Binary code

- Weighted code
  - Binary
    Ex: 0s1
  - BCD
    Ex:
    8421
    2421
    5211
    4221
- Non weighted code
  - Example:
    Gray
    Excess-3
    Five bit BCD
- Reflective code
  - Example:
    5211
    2421
    Excess-3
- Sequential code
  - Example:
    8421
    Excess-3
- Error detecting and correcting code
  - Example:
    Hamming
    Parity

# BCD code

- **8421 BCD Code:**
- Each decimal digit , 0 through 9, is coded by 4 –bit binary number.
- They are weighted code.
- 1010 to 1111 are invalid in BCD.
- Less efficient than pure binary.
- Arithmetic operation are more complex than in pure binary,

| | | |
|---|---|---|
| Decimal | 1 | 4 |
| BCD | 0001 | 0100 |
| Binary | 1110 | |

# Question- Comparison of BCD & Binary.

## Packed BCD:

BCD number corresponding to decimal no. beyond are called packed BCD.

### BCD Addition:

if there is no carry & sum term is not an illegal code, no correction Is needed.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 5 | | | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| + | 1 | 3 | | + | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| | 3 | 8 | | | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

No carry, no illegal code. So, this is the correct sum.

- If there is a carry out of one group to the next group, or if the sum tem is illegal code, then 6 (0110)is added to the sum term of that group & the resulting carry is added to the next group

```
    679.6        0110  0111  1001  .0110
+   536.8      + 0101  0011  0110  .1000
---------      -------------------------
   1216.4        1011  1010  1111  .1110   All are illegal codes
               +0110 +0110 +0110 +.0110    Add 0110 to each
               -------------------------
               10001 10000 10101 1.0100    Propagate carry
           +1     +1    +1    +1
           -----------------------------
           0001  0010  0001  0110  .0100   Corrected Sum
```

# Binary Subtraction

- If there is no borrow from the next higher group then no correction is required.
- If there is a borrow from the next group , then 6(0110) is subtracted from the difference term of this group.

| | | |
|---|---|---|
| | 3 8 | 0 0 1 1 1 0 0 0 |
| - | 1 5 | - 0 0 0 1 0 1 0 1 |
| | 2 3 | 0 0 1 0 0 0 1 1 |

No borrow. So, this is the correct difference.

| | | | |
|---|---|---|---|
| | 206.7 | 0010 0000 0110 .0111 | |
| - | 147.8 | - 0001 0100 0111 .1000 | |
| | 58.9 | 0000 1011 1110 .1111 | Borrows are present |
| | | -0110 -0110 -.0110 | Subtract 0110 |
| | | 0101 1000 .1001 | Corrected difference |

# Excess-3 Code

- Excess -3 code=8421 BCD+0011(3)
- It is a self- complementing code.
- 0000,0001,0010,1101,1110,1111 are illegal codes.

  - Example

| | | |
|---|---|---|
| Decimal | 1 | 4 |
| BCD | 0001 | 0100 |
| XS-3 | 0100 | 0111 |

# Excess-3 addition

- If there is no carry out from the addition of any of the 4 bit group, subtract 0011 from the sum term of these group .If carry out, add 0011 to the sum term of those group.

| | | |
|---|---|---|
| | 3 | 7 |
| + | 2 | 8 |
| | 6 | 5 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 1 | 0 | | 1 | 0 | 1 | 0 |
| + | 0 | 1 | 0 | 1 | | 1 | 0 | 1 | 1 |
| | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| | | | | +1 | | | | | |
| | 1 | 1 | 0 | 0 | | 0 | 1 | 0 | 1 |
| - | 0 | 0 | 1 | 1 | + | 0 | 0 | 1 | 1 |
| | 1 | 0 | 0 | 1 | | 1 | 0 | 0 | 0 |

Propagate carry to next group

- Add 0011 to group which generated carry
- Subtract 0011 to group which do not generated carry

| | | | | | |
|---|---|---|---|---|---|
| | 247.6 | 0101 | 0111 | 1010 | .1001 |
| + | 359.4 | + 0110 | 1000 | 1100 | .0111 |
| | 607.0 | 1011 | 1111 | 1 0110 | 1 .0000 |
| | | | +1 | +1 | |
| | | 1011 | 1 0000 | 0111 | .0000 |
| | | +1 | | | |
| | | 1100 | 0000 | 0111 | .0000 |
| | | - 0011 | +0011 | +0011 | +.0011 |
| | | 1001 | 0011 | 1010 | .0011 |

Carry generated

Propagate carry

Corrected Sum in XS-3

- Add 0011 to group which generated carry
- Subtract 0011 to group which do not generated carry

# Excess-3 Subtraction

- If there is no borrow from the next 4 – bit group & add 0011 to the difference term of such group.
- If there is a borrow ,subtract 0011 from the difference term.

| | | | | |
|---|---|---|---|---|
| 267 | | 0101 | 1001 | 1010 |
| - 175 | - | 0100 | 1010 | 1000 |
| 092 | | 0000 | 1111 | 0010 |
| | | +0011 | -0011 | +0011 |
| | | 0011 | 1100 | 0101 |

- Subtract 0011 to group which generated borrow
- Add 0011 to group which do not generated borrow

| | | | | |
|---|---|---|---|---|
| 57.6 | | 1000 | 1010 | .1001 |
| - 27.8 | - | 0101 | 1010 | .1011 |
| 29.8 | | 0010 | 1111 | .1110 |
| | | +0011 | -0011 | -.0011 |
| | | 0101 | 1100 | .1011 |

- Subtract 0011 to group which generated borrow
- Add 0011 to group which do not generated borrow

# Gray code

- The gray code is a non-weighted code.
- It is a cyclic code because successive code words in this code differ in one bit position only, i.e. it is a unit distance code.
- It is also a reflective code.
- The n least significant bits for $2^n$ through $2^{n+1}-1$ are the mirror images of those for 0 through $2^n-1$.
- An N-bit gray code can be obtained by reflecting an N-1 bit code about an axis at the end of the code, and putting the MSB of 0 above the axis and the MSB of 1 below the axis.
- One reason for the popularity of the gray code is its ease of conversion to and from binary.
- Reflection of gray code is shown in table.

| Gray Code | | | | Decimal | 4-bit binary |
| --- | --- | --- | --- | --- | --- |
| 1-bit | 2-bit | 3-bit | 4-bit | | |
| 0 | 00 | 000 | 0000 | 0 | 0000 |
| 1 | 01 | 001 | 0001 | 1 | 0001 |
| | 11 | 011 | 0011 | 2 | 0010 |
| | 10 | 010 | 0010 | 3 | 0011 |
| | | 110 | 0110 | 4 | 0100 |
| | | 111 | 0111 | 5 | 0101 |
| | | 101 | 0101 | 6 | 0110 |
| | | 100 | 0100 | 7 | 0111 |
| | | | 1100 | 8 | 1000 |
| | | | 1101 | 9 | 1001 |
| | | | 1111 | 10 | 1010 |
| | | | 1110 | 11 | 1011 |
| | | | 1010 | 12 | 1100 |
| | | | 1011 | 13 | 1101 |
| | | | 1001 | 14 | 1110 |
| | | | 1000 | 15 | 1111 |

| Decimal | Binary Code | Gray Code |
|---|---|---|
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0011 |
| 3 | 0011 | 0010 |
| 4 | 0100 | 0110 |
| 5 | 0101 | 0111 |
| 6 | 0110 | 0101 |
| 7 | 0111 | 0100 |
| 8 | 1000 | 1100 |
| 9 | 1001 | 1101 |
| 10 | 1010 | 1111 |
| 11 | 1011 | 1110 |
| 12 | 1100 | 1010 |
| 13 | 1101 | 1011 |
| 14 | 1110 | 1001 |
| 15 | 1111 | 1000 |

electronicsarea.com

# Binary to grey code conversion

- If an n-bit binary number is represented by $B_n\ B_{n-1} \ldots B_1$ and its gray code equivalent $G_n\ G_{n-1} \ldots G_1$, where $B_n$ and $G_n$ are the MSBs, then the gray code bits are obtained from the binary code as follows:

| | | | | |
|---|---|---|---|---|
| $G_n = B_n$ | $G_{n-1} = B_n \oplus B_{n-1}$ | $G_{n-2} = B_{n-1} \oplus B_{n-2}$ | ........ | $G_1 = B_2 \oplus B_1$ |

The conversion procedure is as follows:

1. Record the MSB of the binary as the MSB of the gray code.
2. Perform X-ORing between the MSB of the binary and the next bit in binary. This answer is the next bit of the gray code.
3. Perform X-ORing between $2^{nd}$ bit of the binary and $3^{rd}$ bit of the binary, the $3^{rd}$ bit with the $4^{th}$ bit, and so on.
4. Record the successive answer bits as the successive bits of the gray code until all the bits of the binary number are exhausted.

Example:- Convert the binary 1001 to Gray code.

Ans: $1 \oplus 0 \oplus 0 \oplus 1$

$$\begin{array}{ccccccc} 1 & \rightarrow\oplus\rightarrow & 0 & \rightarrow\oplus\rightarrow & 0 & \rightarrow\oplus\rightarrow & 1 \\ \downarrow & & \downarrow & & \downarrow & & \downarrow \\ 1 & & 1 & & 0 & & 1 \end{array}$$

# Grey To Binary code conversion

If an n-bit gray number is represented by $G_n\ G_{n-1}\ \dots\ G_1$ and its binary code equivalent $B_n\ B_{n-1}\ \dots\ B_1$, where $G_n$ and $B_n$ are the MSBs, then the binary bits are obtained from the gray bits as follows:

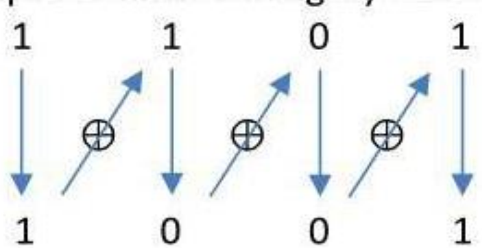
| | | | | |
|---|---|---|---|---|
| $B_n = G_n$ | $B_{n-1} = B_n \oplus G_{n-1}$ | $B_{n-2} = B_{n-1} \oplus G_{n-2}$ | ……… | $B_1 = B_2 \oplus G_1$ |

The conversion procedure is as follows:

1. The MSB of the binary number is the same as the MSB of the gray code number.
2. Perform X-ORing between the MSB of the binary and next significant bit of gray code. This answer is the next bit of binary.
3. Perform X-ORing between the 2nd bit of the binary and 3rd bit of the gray code, the 3rd bit of the binary with the 4th bit of gray code, and so on.
4. Record the successive answers as the successive bits of the binary until all the bits of the gray code are exhausted.

Example:- Convert the gray code 1101 to binary.

Ans.



# Error detecting code

- Noise can alter or distort the data in transmission.
- The 1s may get changed to 0s and 0s to 1s.
- Because digital systems must be accurate to the digit, errors can pose a serious problem.
- Single bit error should be detect & correct by different schemes.
- Parity, Check Sums and Block Parity are the examples of error detecting code.

# Parity

- Parity bit is the simplest technique.
- There are two types of parity – Odd parity and Even parity.
- For odd parity, the parity is set to a 0 or a 1 at the transmitter such that the total number of 1 bits in the word including the parity bit is an odd number.
- For even parity, the parity is set to a 0 or a 1 at the transmitter such that the total number of 1 bits in the word including the parity bit is an even number.

- Detect a single-bit error but can not detect two or more errors within the same word.
- In any practical system, there is always a finite probability of the occurrence of single error.
- E.g. In an even-parity scheme, code 10111001 is erroneous because number of 1s is odd(5), while code 11110110 is error free because number of 1s is even(6).

# Check sum

- Simple parity can not detect two errors within the same word.
- Added to the sum of the previously transmitted words
- At the transmission, the *check sum* up to that time is sent to the receiver.
- The receiver can check its sum with the transmitted sum.
- If the two sums are the same, then no errors were detected at the receiver end.
- If there is an error, the receiving location can ask for retransmission of the entire data.
- This type of transmission is used in teleprocessing system.

# Block parity

- When several binary words are transmitted or stored in succession, the resulting collection of bits can be regarded as a block of data, having rows and columns.
- Parity bits can then be assigned to both rows and columns.
- This scheme makes it possible to *correct* any single error occurring in a data word and to *detect* any two errors in a word.
- This technique also called word parity, is widely used for data stored on magnetic tapes.
- For example, six 8-bit words in succession can be formed into a 6x8 block for transmission.
- Parity bits are added so that odd parity is maintained both row-wise and column-wise and the block is transmitted as a 7x9 block as shown in Figure 1.
- At the receiving end, parity is checked both row-wise and column-wise and suppose errors are detected as shown in Figure 2.
- These single-bit errors detected can be corrected by complementing the error bit.
- In Figure 2, parity errors in the $3^{rd}$ row and $5^{th}$ column mean that the $5^{th}$ bit in $3^{rd}$ row is in error.
- It can be corrected by complementing it.
- Two errors as shown in Figure 3 can only be detected but not corrected.
- In Figure 3, parity errors are observed in both columns 2 and 4.
- It indicated that in one row there two errors.

| | |
|---|---|
| 01011011 | 0 |
| 10010101 | 1 |
| 01101110 | 0 |
| 11010011 | 0 |
| 10001101 | 1 |
| 01110111 | 1 |
| 01110110 | 0 |

Parity Row

Parity Column

Figure 1

| | |
|---|---|
| 01011011 | 0 |
| 10010101 | 1 |
| 01100110 | 0 |
| 11010011 | 0 |
| 10001101 | 1 |
| 01110111 | 1 |
| 01110110 | 0 |

Parity error in 3rd Row

Parity error in 5th Column

Figure 2

| | |
|---|---|
| 01011011 | 0 |
| 10010101 | 1 |
| 01101110 | 0 |
| 10000011 | 0 |
| 10001101 | 1 |
| 01110111 | 1 |
| 01110110 | 0 |

Parity errors in 2nd & 4th Column

Figure 3

# Error correcting code

- 7-bit Hamming Code is widely used error correcting code, containing 4 bits of data and 3 bits of even parity.
- Pattern: $P_1\ P_2\ D_3\ P_4\ D_5\ D_6\ D_7$
- Group-1: $P_1D_3D_5D_7$, Group-2: $P_2D_3D_6D_7$, Group-3: $P_4D_5D_6D_7$

- Example: Data = 1101

  $P_1\ P_2\ D_3\ P_4\ D_5\ D_6\ D_7 = P_1\ P_2\ 1\ P_4\ 1\ 0\ 1$

  $P_1\ D_3\ D_5\ D_7 = 1\ 1\ 1\ 1$

  $P_2\ D_3\ D_6\ D_7 = 0\ 1\ 0\ 1$

  $P_4\ D_5\ D_6\ D_7 = 0\ 1\ 0\ 1$

- 7-bit Hamming Code is 1 0 1 0 1 0 1

# Error correcting code

- How to detect error?
- Example: Received data = 1001001

  $P_1\ P_2\ D_3\ P_4\ D_5\ D_6\ D_7 = 1\ 0\ 0\ 1\ 0\ 0\ 1$

  $P_1\ D_3\ D_5\ D_7 = 1\ 0\ 0\ 1$ (No Error)

  $P_2\ D_3\ D_6\ D_7 = 0\ 0\ 0\ 1$ (Error)

  $P_4\ D_5\ D_6\ D_7 = 1\ 0\ 0\ 1$ (No Error)

- The error word is 0 1 0 = $2_{10}$.
- Complement the $2^{nd}$ bit (from left).
- Correct code is 1 1 0 1 0 0 1

# Digital IC specification

- Threshold voltage
- Propagation Delay
- Power dissipation
- Fan-in
- Fan-out
- Voltage & Current parameters
- Noise Margin
- Operating Temperatures
- Speed power products

# Comparison of logic families

| Characteristic | TTL | CMOS | ECL |
|---|---|---|---|
| Power Input | Moderate | Low | Moderate-High |
| Frequency limit | High | Moderate | Very high |
| Circuit density | Moderate-high | High-very high | Moderate |
| Circuit types per family | High | High | Moderate |

| Logic Family | Propagation delay time (ns) | Power dissipation per gate (mW) | Noise Margin (V) | Fan-in | Fan-out | Cost |
|---|---|---|---|---|---|---|
| TTL | 9 | 10 | 0.4 | 8 | 10 | Low |
| CMOS | <50 | 0.01 | 5 | 10 | 50 | Low |
| ECL | 1 | 50 | 0.25 | 5 | 10 | High |

# Transistor transistor logic

- Dependence on transistors alone to perform basic logic operations.
- Most popular logic family.
- Most widely useful bipolar digital IC family.
- The TTL uses transistors operating in saturated mode.
- It is the fastest of the saturated logic families.
- Good speed, low manufacturing cost, wide range of circuits, and the availability in SSI and MSI are its merits.

# SchottKy TTL

- When a transistor is saturated, excess charge carries will be stored in the base region and they must be removed before the transistor can be turned off.
- So, owing to storage time delay, the speed is reduced.
- The Schottky TTL series reduces this storage time delay by not allowing the transistor to go into full saturation.
- This is accomplished by using a Schottky barrier diode(SBD) between the base and the collector of each transistor.
- More than three times the switching speed of standard TTL, at the expense of approximately doubling the power consumption.

# Tri state TTL

- It utilizes the advantage of the high speed of operation of the totem-pole configuration and wire ANDing of the open-collector configuration.
- It is called the tri-state TTL, because it allows three possible output states: HIGH, LOW, and HIGH Impedance (Hi-Z).
- In the Hi-Z state, both the transistors in the totem-pole arrangement are turned off, so that the output terminal is a HIGH impedance to ground or $V_{CC}$.
- In fact, the output is an open or floating terminal, that is, neither a LOW nor a HIGH.