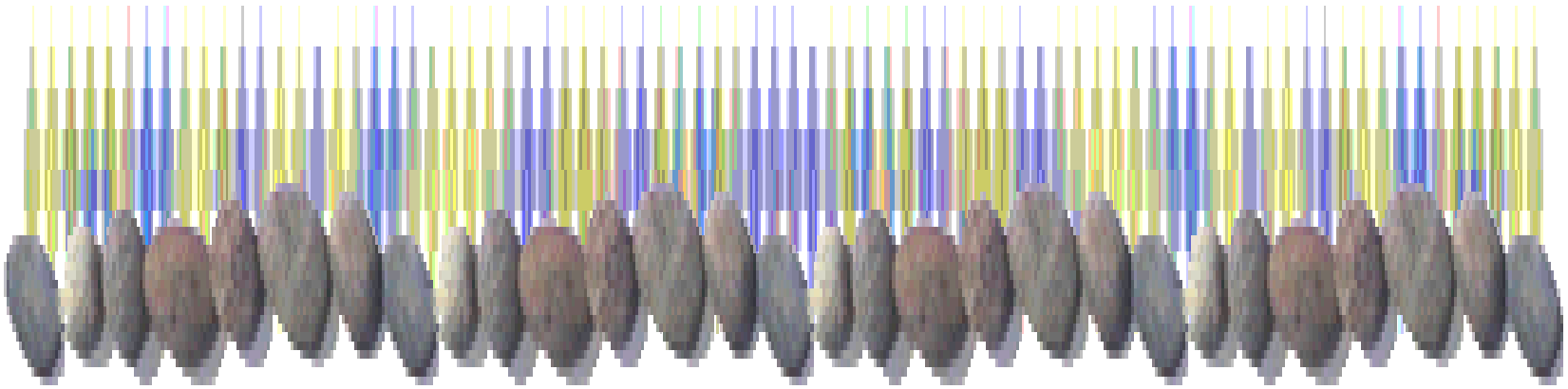
Chapter – 3
Data Representation



Introduction :

- We are all well conversant with the decimal number system as it is used in our daily life.
- But in computer system and other digital system different types of number system were used like binary, octal, hexadecimal etc.
- In this chapter we will study multiplication and division of binary numbers, floating point representation and error diction code.

Number System :

- In mathematics a **‘a number system’** is a set of numbers, (in the broadest sense of word), together one or more operations, such as addition or multiplications.
- Number systems are basically of two types.
  1) **Non Positional**
  2) **Positional**
- In this system each symbol represents the same value irrespective of its position.
- Hence it is difficult to perform different arithmetic operation with such a number system.
- However in positional number system arithmetic calculation can be easily performed because in this system each symbol represents different values depending on position they occupy in the number.

Number System :

- Following types of number systems are used commonly:

1) **Decimal number system**
2) **Binary number system**
3) **Octal number system**
4) **Hexadecimal number system**

1) Decimal number system (base 10)

- This system is used in out daily life. In this system base is 10, and there are 10 (ten) characters. 0,1,2,3,4,5,6,7,8,9, more than one characters are used to show number more than 9.
- Each position of number is given defining weight age,

  e.g. $1234 = 1000 + 200 + 30 + 4$

  $= 1*10^3 + 2*10^2 + 3*10^1 + 4*10^1$

  $= 1234.$

2) Binary number system (base 2)

- Binary number system is used in computer and other digital systems.
- In this system there are only two character 0 and 1. These shows two different conditions.
- This two condition can be represented by SWITCH, PUNCH CARD or TAPE or TRANSISTOR in cut off.
- For example open switch shows 0 state and close switch shows 1 state.
- In binary system values increases to the left of binary point as 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096…

3) Octal number system (base 8)

- In the octal number system base is 8 and there are 8 character : 0,1,2,3,4,5,6,7.
- In this number system values increase from right to left as 1, 8, 64, 512, 4096…
- Octal numbers with their equivalent decimal and binary number are mentioned in given table.

3) Octal number system (base 8)

Decimal	Octal	Binary
0	0	000
1	1	001
2	2	010
3	3	011
4	4	100
5	5	101
6	6	110
7	7	111
8	10	1000

4) Hexadecimal number system (base 16)

- This number system is very useful in microprocessor.
- In this system base is 16. These characters are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
- In this number system values increase to left of hexadecimal point as 1, 16, 256, 65536 and so on.
- Hexadecimal numbers with their decimal as well as binary and octal is shown.

4) Hexadecimal number system (base 16)

Decimal	Hexadecimal	Octal	Binary
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	8	10	1000
9	9	11	1001
10	A	12	1010
11	B	13	1011
12	C	14	1100
13	D	15	1101
14	E	16	1110
15	F	17	1111

❑ Rules of Binary addition & subtraction

❖ Binary Addition

Number		SUM A+B	Carry
A	B		
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

❖ For example :

```
  1 1 1 0 1
+ 0 0 1 0 1
-----------
=1 0 0 0 1 0
```

❑ Rules of Binary addition & subtraction

❖ Binary Subtraction

Number		SUM A-B	Borrow
A	B		
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

❖ For example :

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

❑ Perform the multiplication of following binary numbers.

❖ For example :

1) 1 1 0 1 * 1 0 0

1101

* 100

= 110100

+ 00000

+ 0000

= 110100

❑ Perform the multiplication of following binary numbers.

❖ For example :

2) 1 1 1 1 * 0 1 1

1111

* 011

= 000000

+ 11110

+ 1111

= 101101

❑ Perform the multiplication of following binary numbers.

❖ For example :

3) 1 0 1 0 . 1 * 1 1 0

1010.1

* 1 1 0

= 1010100

\+ 101010

\+ 00000

= 111111.0

❑ Perform the multiplication of following binary numbers.

❖ For example :

4) 1 1 0 * 1 1 0

110

* 110

= 11000

+ 1100

+ 000

= 100100

❑ Perform the multiplication of following binary numbers.

❖ For example :

5) 1 1 . 10 * 1 1

11.10

* 11

= 11100

+ 1110

= 1010.10

❑ Perform Division of following binary Number.

- For example :

1) 10110 / 10 =1011
2) 10000 / 101=111011

❑ Floating point representation of a number:

- In computing floating point is describes a system for numeric representation in which a string of a digit (or bits) represent a relational number.
- It is divide into two parts : which are first part: **signed** (નિશાની) and second part : **mantissa** (સખ્યા).
- The term floating point refers to the fact that the radix point (decimal point or more commonly in computers, binary point) can “float”. That is it can be placed anywhere relative to the significant digits of the number.

❑ Floating point representation of a number:

- E.g. Decimal Number : 1234.567

Fraction	Exponent
+0.1234567	+04

- This representation equivalent to the scientific notation $+0.1234567 * 10^4$
- General form of floating point is : $n * k^e$
- Where n is mantissa, e is exponent while k is radix. Which is shown in storage layout form.

❑ Floating point representation of a number:

❖ Storage Layout

- ➢ Floating point structure for single and double precision shown below:
- ➢ The following figure shows the layout for single (32-bit) and double (64 bit) precision floating point values.

Type	Sign	Exponent	Mantissa	Bits	Range
Single	1	8	23	32	127
Double	1	11	52	64	1023

❑ Floating point representation of a number:

- **The sign bit**
- It represent sign bit if its value is 1 number is negative other wise number is positive.
- **The Exponent**
- The exponent field needs to represent both positive and negative exponent.
- For example precision exponent range is 8 bit means -128 to +127. for double precision range is 11 bit means -1024 to +1023

❑ Floating point representation of a number:

❖ The Mantissa

- ➢ The mantissa also known as the significant, represents the precision bits of the number.

❑ Normalization :

- A floating point number is said to be normalized if the most significant digit of the mantissa is nonzero.
- Similarly in case of binary number

  $0.01111 * 2^4$ is non normalized number. And $0.111011 * 2^5$ is normalized.
- In some computers only normalized numbers are stored.
- The process of shifting of mantissa to the left to make the most significant bit to be non-zero is called normalization.

❑ Fixed Point Representation :

- In a fixed point representation of numbers the binary or decimal point is assigned to be at the right or left of the number.
- This method is not widely used.

❖ **Parity Bit**

- A parity bit is a bit that is added to ensure that the number of bits with the value one in a set of bits is even or odd.
- Parity bits are used as simplest form of error detection code.
- There are two variants of parity bit :even parity bit and odd parity bit.

❑ Error Detection Code :

- **Definition of error detection & correction code :**
- **Error detection:** error detection is ability to detect the presence of errors cause by noise or other impairments during transmission from the transmitter to the receiver.
- **Error correction:** error correction is the additional ability to reconstruct the original error free data.
- There are two basic ways to design the channel code and protocol for an error correcting system.

❑ Error Detection Code :

- ❑ **Automatic Repeat reQuest : (ARQ)**
  - ➢ The transmitter sends the data and also an error detection code, which the receiver uses to check for errors, and requests retransmission of erroneous data.
- ❑ **Forward Error Correction : (FEC)**
  - ➢ The transmitter encodes the data with an error correcting code (ECC) and sends the coded message.
  - ➢ The receiver never sends any messages back to the transmitter.
  - ➢ All error detection code transmit more bit than original data.

❑ Error Detection Code :

❑ Cyclic Redundancy Check (CRC) :

- ➢ More complex error detection and correction methods make use of the properties of fields and polynomials (error code) over such fields.
- ➢ The cyclic redundancy check consider a block of data as the coefficients to a error code and then divides by a fixed, predetermined error code.

❑ Error Detection Code :

- ❑ **Hamming distance based checks :**
  - ➢ If we want to detect d bit errors in an n bit word we can map every n bit word into a bigger n+d+1 bit word so that the minimum hamming distance between each valid mapping is d+1.
- ❑ **Hash function :**
  - ➢ Any hash function can be used as a integrity check.
- ❑ **Horizontal and Vertical redundancy check :**
  - ➢ Other typed of redundancy check include horizontal redundancy check, vertical redundancy check, and double or dual parity used.