UNIT: - 3

Data Representation and Number Systems

Topic Covered :-

- Representation of Number, Binary, Octal, Hexadecimal number and its arithmetic.
- Representation of Integers, Representation of Fractions, Representation of Character, Characters codes (ASCII, EBCDIC, UNICODE)
- Binary arithmetic's: Binary addition and subtraction. Binary Multiplication and Division with the help of long-hand method.
- Conversion of Numbers: Conversation of number in Decimal, Binary, Octal, Hexadecimal.

REPRESENTATION OF INFORMATION
OR

❖ What is an Internal and External Data Representation?

- **Internal Representation** : "The data represented in the format that computer can understand is known as Internal data representation."
  For Example : Binary Representation

- **External Representation :** " The representation of data that user or human being can understand is external data representation"
  For Example : Text, Audio, Video, Graphics, Animation etc.

❖ WHAT ARE NUMBER SYSTEMS?

The information is represented in a computer is in the form of ***Binary digit*** popularly called bit.

A number system of base (also called radix) r is a system, which have r distinct symbols for digits. A string of these symbolic digits represent a number.

To determine the quantity that the number system represents, we multiply the number by an integer power of r depending on the place it is located and then find the sum of weighted digits. There are four number bases commonly used in programming, Binary, Octal Decimal and Hexadecimal. However most of the time we shall meet with Binary, Decimal and Hexadecimal number systems. These number systems have been differentiated according to their base number.

Every numbering system has its own base number and representation symbol. I have presented these four numbers in the following table:

Name of Number System	Base Number	Symbol Used for Representation
Binary	2	B
Octal	8	Q or O
Decimal	10	D or None
Hexadecimal	16	H

1. Binary Numbers

Today most of the modern computer systems operate using binary logic. The computer represents values using two voltage levels that indicate to either OFF or ON using 0 and 1.

They can also be represented as a string of these two – digits called bits. The base of binary or binary number system is 2.

For converting the value of binary number to decimal equivalent we have to find its quantity, which is found by multiplying a digit by its place value.

The following table shows the representation of binary number against the decimal numbers:

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

Example : binary number 101010 is equivalent to

$(1*2^{5)} + (0*2^{4)} + (1*2^{3)} + (0*2^{2)} + (1*2^{1)} + (0*2^{0)}$

$= (1*32) + (0*16) + (1*8) + (0*4) + (1*2) + (0*1)$

$= 32 + 8 + 2 = (42)_{10}$ in decimal system

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

2. Decimal Numbers

Decimal number systems has ten digits represented by 0,1,2,3,4,5,6,7,8 and 9. any decimal number can be represented as a string of these digits and since there are ten decimal digits, so the base or radix of this system is 10.

Example : a string of number 234 can be represented in quantity as

: $(2 * 10^2) + (3*10^1) + (4*10^0)$

3. Octal Numbers

An octal number system has eight digit represented as 0,1,2,3,4,5,6,7. The base of octal number system is 8.

For finding equivalent decimal number of an octal number, we have to find the quantity of the octal number.

Example : Octal number $(23.4)_8$ is equivalent to

$(2*8^1) + (3*8^0) + (4*8^{-1})$

$= (2*8) + (3*1) + (4 * 1/8)$

$= 16 + 3 + 4/8$

$= 16 + 3 + 0.5$

$= (19.5)$ in decimal system

$(23.4)_8 = (19.5)_{10}$

4. Hexadecimal Numbers

The hexadecimal system has 16 digits, which are represented as 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. The base of hexadecimal number system is 16.

For finding equivalent decimal number of an hexadecimal number, we have to find the quantity of the hexadecimal number.

The Hexadecimal Number System uses base 16 and includes only the digits 0 through 9 and the letters A, B, C, D, E, and F. We use H with the number to denote any hexadecimal number. The following table shows the representation of various number systems, differentiating them with each other

Binary (B)	Octal (Q)	Decimal	Hex (H)
0000	0	00	0
0001	1	01	1
0010	2	02	2
0011	3	03	3
0100	4	04	4
0101	5	05	5
0110	6	06	6
0111	7	07	7
1000	10	08	8
1001	11	09	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

Example : $(F2)_{16}$ is equivalent to

$(F*16^1) + (2*16^0)$

$= (15*16) + (2*1)$

$= 240 + 2$

= (242) in decimal system

$(F2)_{16} = (242)_{10}$

❖ CONVERSION

➢ Decimal to Others

1. Decimal to Binary Conversion

- In this method, the mantissa part of the number is repeatedly divided by 2 and noting the reminder, which will be either 0 or 1. This division is continued till the mantissa becomes zero.
- The reminders, which are noted down during the division is read in the reverse order to get the binary equivalent.
- **Example** : **114**

2	114	
2	7	0
2	3	1
2	1	1
	0	1

The number is written from below, that is 1110. So the Binary equivalent of 114 is 1110.

$(114)_{10} = (1110)_2$

- If the decimal number has a fractional part, then the fractional part is converted into binary by multiplying it with 2. Only the integer of the result is noted and the fraction is repeatedly multiplied by 2 until the fractional part becomes 0.
- **Example** : **0.125**

```
0.125
*   2
 0.25        0
*   2
  0.5       0
*   2
  1.0       1
```

The number is written from top. i.e., 0.001. So the Binary equivalent of 0.125 is 0.001

$(0.125)_{10} = (0.001)_2$

2. Decimal to Octal Conversion

- In this method, the mantissa part of the number is repeatedly divided by 8 and noting the reminder, which will be between 0 to 7.This division is continued till the mantissa becomes zero.
- The reminders, which are noted down during the division is read in the reverse order to get the octal equivalent.

- **Example: 888**

8	888	
8	111	0
8	13	7
8	1	5
	0	1

The number is written from below, that is 1110. So the Octal equivalent of 888 is 1570.

$$(888)_{10} = (1570)_8$$

- If the decimal number has a fractional part, then the fractional part is converted into octal by multiplying it with 8. Only the integer of the result is noted and the fraction is repeatedly multiplied by 8 until the fractional part becomes 0.

- **Example : 0.0625**

```
0.0625
*     8
  0.50    0
*    8
     0    4
```

The number is written from top. i.e., 0.04. So the octal equivalent of 0.0625 is 0.04

3. Decimal to Hexadecimal Conversion

- In this method, the mantissa part of the number is repeatedly divided by 16 and noting the reminder, which will be between 0 to 9,A,B,C,D,E,F. This division is continued till the mantissa becomes zero.
- The reminders, which are noted down during the division is read in the reverse order to get the Hexadecimal equivalent.

- **Example: 888**

16	888	
16	55	8
16	3	7
16	0	3

The number is written from below, that is 378. So the Hexadecimal equivalent of 888 is

$$378\ (888)_{10} = (378)_{16}$$

- If the decimal number has a fractional part, then the fractional part is converted into Hexadecimal by multiplying it with 16. Only the integer of the result is noted and the fraction is repeatedly multiplied by 16 until the fractional part becomes 0.

- **Example : 0.62**

0.62
* 16
0.92 9
* 16
0.72 14 → E
* 16
0.52 11 → B
* 16

The process will further continue. Therefore , the result has been taken up to 3 decimal places.

The number is written from top. i.e., 0.9EB. So the Hexadecimal equivalent of 0.62 is 0.9EB

$$(0.62)_{10} = (0.9EB)_{16}$$

Others to Decimal

4. Binary to Decimal Conversion

- To convert a binary number to its equivalent decimal we use the following expression. The weight of the nth bit of a number from the right hand side = nth bit * 2^{n-1}. After calculating the weight of each bit, they are added to get the decimal value.

- **Example : 101**

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

$(101)_2 = (5)_{10}$

- **Example : 1.001**

$= (1 * 2^0) + (0 * 2^{-1}) + (0 * 2^{-2}) + (1 * 2^{-3})$
$= (1 * 1) + (0 * 1/2) + (0 * 1/4) + (1 * 1/8)$
$= (1) + (0) + (0) + (1/8)$
$= 1 + 0.125$
$= 1.125$

$(1.001)_2 = (1.125)_{10}$

5. Octal to Decimal Conversion

- To convert a Octal number to its equivalent Decimal we use the following expression. The weight of the nth bit of a number from the right hand side = nth bit * 8^{n-1}. After calculating the weight of each bit, they are added to get the decimal value.

- **Example : 32**

$= (3 * 8^1) + (2 * 8^0)$
$= (3 * 8) + (2 * 1)$
$= 24 + 2$
$= 26$

$(32)_8 = (26)_{10}$

- **Example : 30.05**

$= (3 * 8^{1}) + (0 * 8^{0}) + (0 * 8^{-1}) + (5 * 8^{-2})$
$= (3 * 8) + (0 * 1) + (0 * 1/8) + (5 * 1/64)$
$= (24) + 0 + 0 + 0.078125$
$= 24.078125$

$(30.05)_8 = (24.078125)_{10}$

6. Hexadecimal to Decimal Conversion

- To convert a Hexadecimal number to its equivalent Decimal we use the following expression. The weight of the nth bit of a number from the right hand side = nth bit * 16^{n-1}. After calculating the weight of each bit, they are added to get the decimal value.

- **Example : BA1**

B → 11 A → 10

$= (11 * 16^{2}) + (10 * 16^{1}) + (1 * 16^{0})$
$= (11 * 256) + (10 * 16) + (1 * 1)$
$= 2816 + 160 + 1$
$= 2977$

$(BA1)_{16} = (2977)_{10}$

- **Example : 123A.BA**

$= (1 * 16^{3}) + (2 * 16^{2}) + (3 * 16^{1}) + (10 * 16^{0}) + (11 * 16^{-1}) + (10 * 16^{-2})$
$= (1 * 4096) + (2 * 256) + (3 * 16) + (10 * 1) + (11 * 1/16) + (10 * 1/256)$
$= (4096) + (512) + (48) + (10) + (0.6875) + (0.0390635)$
$= 4666.7265$

$(123A.BA)_{16} = (4666.7265)_{10}$

7. Binary to Octal Conversion

- **To convert from an integer binary number to octal we follow the following two steps:**

1. First break the binary number into 3-bit sections from the LSB to the MSB.
2. And then convert the 3-bit binary number to its octal equivalent.

- **Example** : 11001011010001

3-bit Section of Binary Number	011	001	011	010	001
Equivalent number	3	1	3	2	1

The Octal equivalent of $(11001011010001)_2$ is $(31321)_8$

$(11001011010001)_2 = (31321)_8$

- **Example** : 10011001010.11001010

3-bit Section of Binary Number	010	011	001	010		110	010	100
Equivalent number	2	3	1	2	•	6	2	4

The Octal equivalent of $(10011001010.11001010)_2$ is $(2312.624)_8$

$(10011001010.11001010)_2 = (2312.624)_8$

8. Binary to Hexadecimal Conversion

- To convert a binary number into hexadecimal format, first of all pad the binary number with leading zeros on the left most side to make sure that the binary number contains multiples of four bits. After that follow the following two steps:
  1. First, break the binary number into 4-bit sections from the LSB to the MSB.
  2. And then convert the 4-bit binary number to its Hex equivalent.

- **Example :** 11001011010001

4-bit binary number section	0100	1110	1101	0011
Hexadecimal value	4	E	D	3

The Hexadecimal equivalent of $(1100101101001)_2$ is $(4ED3)_{16}$

- **Example :** 1000111.111000111

4-bit binary number section	0100	0111		1110	0011	1000
Hexadecimal value	4	7	•	E	3	8

The Hexadecimal equivalent of $(1000111.111000111)_2$ is $(47.E38)_{16}$

$(1000111.111000111)_2 = (47.E38)_{16}$

9. Octal to Binary Conversion

- To convert any integer octal number to its corresponding binary number we follow the following two steps:
  1. First convert the decimal number to its 3-bit binary equivalent.
  2. And then combine the 3-bit sections by removing the spaces.

- **Example :** 31321

Equivalent number	3	1	3	2	1

3-bit Section of Binary Number	011	001	011	010	001

Thus the binary equivalent for the octal number 31321(Q) is 011 0010 1101 0001.

$$(31321)_8 = (011\ 001\ 011\ 010\ 001)_2$$

- **Example : 777.**01

Equivalent number	7	7	7	•	0	1
3-bit Section of Binary Number	111	111	111		000	001

Thus the binary equivalent for the octal number 777.01 is 111111111.000001

$$(777.01)_8 = (111\ 111\ 111.000\ 001)_2$$

10. <u>Hexadecimal to Binary Conversion</u>

- To convert a hexadecimal number into a binary number we follow the following two steps:
  1. First, convert the Hexadecimal number to its 4-bit binary equivalent.
  2. And then combine the 4-bit sections by removing the spaces.

- **Example :** 4ED3

Hexadecimal value	4	E	D	3
4-bit binary number section	0100	1110	1101	0011

Thus the binary equivalent for the Hexadecimal number 4ED3 is 0100111011010011

$$(4ED3)_{16} = (0100\ 1110\ 1101\ 0011)_2$$

- **Example :** 123A.BA

Hexadecimal value	1	2	3	A	•	B	A
4-bit binary number section	0001	0010	0011	0100		1011	1010

Thus the binary equivalent for the Hexadecimal number 123A.BA is 0001 0010 0011 0100.1011 1010

$$(123A.BA)_{16} = (0001\ 0010\ 0011\ 0100\ .\ 1011\ 1010)_2$$

11. Octal to Hexadecimal Conversion

- To convert a Octal number into a Hexadecimal number we follow the following steps:
- First of all convert the Octal to Binary.
- Now Convert from binary to Hexadecimal
- **Example : 2534**

Convert from Octal to Binary

Equivalent number	2	5	3	4
3-bit Section of Binary Number	010	101	011	100

Convert from Binary to Hexadecimal

4-bit binary number section	0101	0101	1100
Hexadecimal value	5	5	C

Thus the Hexadecimal equivalent for the octal number 2537 is 55C

$(2537)_8 = (55C)_{16}$

- **Example : 234.23**

Convert from Octal to Binary

Equivalent number	2	3	4	•	2	3
3-bit Section of Binary Number	010	011	100		010	011

Convert From Binary to Hexadecimal

4-bit binary number section	0000	1001	1100	•	0100	1100
Hexadecimal value	0	9	C		4	C

$(234.23)_8 = (09C.4C)_{16}$

12. Hexadecimal to Octal Conversion

- To convert from Hexadecimal number into a Octal number we follow the following steps:
- First of all convert the Hexadecimal to Binary.
- Now Convert from binary to Octal
- **Example :** 123A

Hexadecimal to Binary

Hexadecimal value	1	2	3	A
4-bit binary number section	0001	0010	0011	1010

Binary to Octal

3-bit Section of Binary Number	000	001	001	000	111	010
Equivalent number	0	1	1	0	7	2

$(123A)_{16} = (0111072)_8$

- **Example :** 0.0ABC

Hexadecimal to Binary

Hexadecimal value	0	•	0	A	B	C
4-bit binary number section	0000		0000	1010	1010	1100

Binary to Octal

3-bit Section of Binary Number	000	000	•	000	010	101	011	110	000
Equivalent number	0	0		0	2	5	3	6	0

$(0.0ABC)_{16} = (00.025360)_8$

BINARY ARITHMETIC

Arithmetic is at the heart of the digital computer, and the majority of arithmetic performed by computers is binary arithmetic, that is, arithmetic on base two numbers.

❖ **Binary Addition**

- The rules for binary addition are the same as those for any positional number System.

Adding numbers in binary is quite easy. Addition is done exactly like adding decimal numbers, except that you have only two digits (0 and 1).

- **The only number facts to remember are that**

0+0 = 0, with no carry,
1+0 = 1, with no carry,

0+1 = 1, with no carry,
1+1 = 0, and you carry a 1.

- Binary addition works on the same principle, but the numerals are different. Begin with one-bit binary addition:

```
  0     0    1
 +0    +1   +0
___   ___  ___
  0     1    1
```

1+1 carries us into the next column. In decimal form, 1+1=2. In binary, any digit higher than 1 puts us a column to the left (as would 10 in decimal notation). The decimal number "2" is written in binary notation as "10" (1*2^1)+(0*2^0). Record the 0 in the ones column, and carry the 1 to the twos column to get an answer of "10." In our vertical notation,

```
  1
 +1
___

 10
```

Example : The process is the same for multiple-bit binary numbers:

```
  1010
+1111
-----
 11001
```

STEPS

Step 1:
Column 2^0: 0+1=1.
Record the 1.
Temporary Result: 1; Carry: 0
Step 2:
Column 2^1: 1+1=10.
Record the 0, carry the 1.
Temporary Result: 01; Carry: 1
Step 3:
Column 2^2: 1+0=1 Add 1 from carry: 1+1=10.
Record the 0, carry the 1.
Temporary Result: 001; Carry: 1
Step 4:
Column 2^3: 1+1=10. Add 1 from carry: 10+1=11.
Record the 11.
Final result: 11001

❖ **Binary Subtraction**

- Subtraction is generally simpler than addition since only two numbers are involved and the upper value representation is greater than the lower value representation.
- The problem of "borrow" is similar in binary subtraction to that in decimal. We can construct a subtraction table that has two parts - the three cases of subtracting without borrow, and the one case of the involvement of a borrow digit, no matter how far to the left is the next available binary digit.

A	B	A-B
0	0	0
1	0	1
1	1	0
0	1	1 (borrow 10)

- **Example :**

```
   1001010
-    10100
  ---------
   0110110
```

Answer = 0110110

❖ **Binary Multiplication**

- Multiplication in the binary system works the same way as in the decimal system:

1*1=1
1*0=0
0*1=0

- **Example : 101 * 11**

```
  101
* 11
 ____
  101
 1010
 _____
 1111
```

Answer = 1111

❖ **Binary Division**

- Follow the same rules as in decimal division. For the sake of simplicity, throw away the remainder.
- **Example**: **111011/11**

```
    10011
   _______
11)111011
  -11
  ______
    0101
     -11
    _____
      101
       11
    _____
       10
```

Answer : 10011 Reminder : 10

❖ **COMPLEMENTS METHOD**

- Computers use complemented numbers or complements to perform subtraction. In the binary number system there are two types of complements – 1"s Complement and 2"s Complement.
- Similarly in the decimal number system also there are two types of complements – the 9"scomplement and the 10"s complement.

❖ **1's Complement**

- The 1"s complement in the binary system is similar to the 9"s complement in the decimal system. To get the 1"s complement of a number, each bit is subtracted from 1.

Example : The 1's complement of 1010 is 0101

```
  1111
- 1010
  0101
```

1's Complement : 0101

❖ **2's Complement**

- The 2"s complement in the binary number system is similar to the 10"s complement in the decimal system. To get the 2"s complement of a number, add 1 to the 1"s complement of the number.

- **Example : 2's complement of 1010 is**
  0110 1's Complement : 0101
  2' Complement = 1's Complement + 1

  0101 +
  1 0110

- **9's Complement**

- The 9"s complement of a decimal number is obtained by subtracting each digit of the number from 9.
- **Example : 9's Complement of 2 is 7.**

  9
  - 2
  7

9's Complement of 123 is 876

999
- 123
876

❖ **10's Complement**

- The 10"s complement of decimal number is obtained by adding one to the 9"s complement of that number.
- **Example : 10's complement of 2 is 8**

  9's complement + 1 = 7 + 1 = 8
- **10's complement of 123 is 877**

  10's Complement = 9's complement + 1

  = **(999 – 123) + 1**

  = 876 + 1

  = **877**

BITS, BYTES AND WORDS

A byte is basic grouping of bits(binary digits) that the computer operates on as a single unit. It consists of 8 bits and is used to represent a character by the ASCII and EBCDIC coding system. A word is a grouping of bits (usually larger than byte) that is transferred as a unit between primary storage and the registers of the ALU and Control Unit.

Character Codes(ASCII & EBCDIC)

1. **ASCII** Stands for **AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE**

2. **EBCDIC** Stands for **EXTENDED BINARY CODED DECIMAL INTERCHANGE CODE**

The mapping of characters onto integers is called a **character code**.

Each computer has a set of characters that it use. This set includes letters A-Z, a-z ,digits 0-9 and a set of special symbols such as space, period, minus sign, comma , carriage return.

To transfer these characters into the computer, each one is assigned a integer number. For example a = 1, b=2 ,…, z =26, + = 27.

For communication, computers use the same code or they will not be able to understand one another. For that purpose standards have been developed.

1. Binary Coded Decimal (BCD)

The BCD code is the simplest binary code that is used to represent a decimal number. In the BCD code four bits represent are decimal number.

For example 2 is represented as 0010.

If a decimal number is having more than 1 digit, each decimal digit is represented individually.

Example : 123 is represented as 000100100011.

There is a difference between the binary equivalent of a decimal number and in BCD code. For e.g : the binary equivalent of 45 is 101101 and its BCD code is 01000101.

Computers perform subtraction using compliment method and it is difficult to find compliments when numbers are represented using BCD code.

For e.g : 1"s compliment of 2 (0010) is 1101 which is = 13 in the decimal system and it is not valid BCD code.

To overcome this difficulty other BCD codes such as "ACCESS – 3" are use.

2. ASCII (American Standard Code for Information Interchange)

ASCII code stands for American Standard Code for Information Interchange.

ASCII is the most popular standard or most common alphanumeric code used to represent information in a computer like A,B,C, numerals,1,2…etc is for letters numbers and other symbols.

This code allows manufacturers to standardize computer hardware such as keyboard, printers and video displays.

- **ASCII – 7**
  - Each ASCII character has 7 bits, so it is known as "ASCII-7". it allows to store 128 characters. It also defined codes to convey information such as end of line, end of page, etc… to the computer.
  - These codes are said to be control characters which are not printable. For example CR for carriage returns.

- It is a 7 it code so it is also known as ASCII – 7.
- Micro computers use 8 bit word length uses 7 bit to represent the basic code.
- The 8^{th} bit is used for parity(error checking) or it may be permanently 0 or 1.
- With 7 bits up to 128 characters can be coded.
- A letter, digit, special symbols etc… are known as characters.
- It includes upper and lower case alphabets, numbers, punctuation marks, special and control characters.
- The 7 bits in ASCII code format can be represented as

$$X_6X_5X_4X_3X_2X_1X_0$$

Where each X is either 0 or 1. for example upper case letter A can be coded as 1000001

And lower case letter „a" can be coded as

1100001

Similarly there are ASCII code for special characters. Some examples are 010 0100 ($)
010 1011(+)
011 1101(=)

➢ **ASCII – 8**

- The newer version of ASCII is "ASCII – 8" code
- It is an eight(8) bit code.
- It uses 8 bits to represent basic code and 9^{th} bit can be added for parity checking.
- Using 8 bit up to 256 characters can be represented.

3. EBCDIC (Extended Binary Coded Decimsal Interchange Code)

- It is the standard character code for large computer. It is an 8 bit code without parity.
- 9th bit can be added for parity.
- With 8 bits up to 256 characters can be coded.
- In ASCII-8 and EBCDIC code, the first 4 bits are known as "Zone bit" and remaining 4 bits represent "digit value"
- In ASCII – 7, the first 3 bits are zone bits and remaining 4 bits are digit value.

REPRESENTATION OF INTEGERS

Decimal digits are considered as characters and codes are assign to them. These codes are digits are primarily use when digits are used only as symbol with no value.

For e.g : "091001", is the identification number that has no value for calculation purpose, so we can use ASICII code to represent this text.

If we want to sore decimal number in a computer and perform arithmetic operation, the representation must have a value assign to the number.

In order to this we convert decimal number to binary number that has a value equal to the given decimal number.

The value of each digit in the number is determined by following 3 things. Digit itself

- The position of the digit in the number
- The base of the number system.

For e.g : the decimal number given is 123. The value of the number is calculated as the following :

1 * 100 2 * 10 3*1

[Hundred‟s position] [Ten‟s position] [One‟s position]

The notation used to express numbers given above is known as positional system.

When the number system is binary only two symbols – 0 and 1 are there and the base of a number is 2.

When a number is given like 1011000 , the right most bit is known as “Least Significant Bit” (LSB) and the left most bit is known as “ Most significant Bit”(MSB).

We require 3 bits to represent a decimal number from 0 to 7 and 4 bits represent 8 and 9.
So the average bits needed to represent the decimal digit keys :

[(8 * 3) + (2 * 4)]/10 = (24+8)/10 = 32/10 = 3.2 (Approximate value)

REPRESENTATION OF FRACTIONS

Decimal fractions are represented as the following :

0.235 = 2 * 10(-1) + 3 * 10 (-2) + 5 * 10 (-3)

Decimal Point [one – tenth(1/10) Position] [one – hundredth(1/100) Position] [one – thousand(1/1000) Position]

Observe that negative power of 10 is used as weight to multiply the digit in the fractional part of the number.

Binary fractions are represented by 1s and 0s. The right side of a binary point the bits are multiply by negative power of two to get the decimal value of binary fraction as shown here.

$(0.1011) = 1 * 2^{(-1)} + 0 * 2^{(-2)} + 1 * 2^{(-3)} + 1 * 2^{(-4)}$

Binary Point	[1/1st Position]	[1/2nd Position]	[1/3rd Position]	[1/4th Position]

Notice that we have used subscript 2 to indicate that the number is binary. This notation to represent the base of a number is useful to prevent miss – interpretation of members.

REPRESENTATION OF HEXA DECIMAL NUMBERS

The average number of bits needed to represent a decimal digit is 3.2. so the binary equivalent of a 10 digit decimal number will be a approximately 32 bit long.

It is difficult to write such a long number of 1s and 0s and convert them to equivalent decimal number without making mistake.

The hexadecimal system uses 16 as a base which is useful notation to express binary numbers. These system uses 16 symbols 0 to 9 and A to F.

Here, in hexadecimal number system the alphabets are treated as numbers and not as characters.

16 is a power of 2 namely 24 . there is one to one relationship between a hexadecimal and its equivalent binary digit means that we need 4 bits to represent a hexadecimal digit.

Binary numbers can be quickly converted into its hexadecimal equivalent by grouping least significant bit and replacing each 4 bit group with its hexadecimal equivalent.

For example :
000100011111000000101100
Hexadecimal number is 11F0.2C

Observe that groups are created from left to right for fractional part and from right to left for integer part.

If the number of bits in the integer part is not multiple of 4, we insert leading 0s because leading 0s have no significance for the integer part.

If the number of bits in the fractional part is not a multiple of 4 then we need to add trailing 0s that have no significance in the fractional part.

[Hamming's algorithm for 16 bit word]

Given the code is 16 bit code as follows : 1111000010101110
Codeword(m + r) = 001011100000101101110
Received codeword is = 00100111000000101101110

Original string : 1 1 1 1 0 0 0 0 1 0 1 0 1 1 1 0

Code Word : 001011100000101101110
Received Code : 001001110000010110 1110

The parity check bit will be checked with the following results.

Code Word

0	0	1	0	1	1	1	0	0	0	0	0	1	0	1	1	0	1	1	1	0
1	**2**	3	**4**	5	6	7	**8**	9	10	11	12	13	14	15	**16**	17	18	19	20	21

Received Code

0	0	1	0	0	1	1	0	0	0	0	0	1	0	1	1	0	1	1	1	0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

Bit 1 [Incorrect] = 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21
[1 1 1 1 1] = 5
Bit 2 [Correct] = 2, 3, 6, 7, 10, 11, 14, 15, 18, 19
[1 1 1 1 1 1] = 6
Bit 4 [Correct] = 4, 5, 6, 7, 12, 13, 14, 15,20, 21
[1 1 1 1 1 1] = 5
Bit 8 [Correct] = 8, 9, 10, 11, 12 ,13, 14, 15
[1 1] = 2
Bit 16[Correct] = 16, 17, 18, 19, 20, 21
[1 1 1 1] = 4

[Example of Hamming‟s Algorithm]

The total number of 1‟s in the bit position checked by bit 1 should be even because even parity check method is used. But it is odd so incorrect bit must be one of the bit checked by parity bit 1.

Parity bit 4 is also incorrect means that the bits checked by parity bit 4 are also incorrect. The error must be one of the bits common in the list of checked bit by 1 and 4

5, 7, 13, 15, 21

But bit 2 is correct so eliminates 7 and 15.

Same way bit 8 is also correct, eliminates 13 and bit 16 is also correct , eliminate 21. Only one bit is remaining is bit 5 which is the one with error. As it was read as 0 it should be 1.

In this manner error can be corrected using hamming‟s algorithm.