

# NUMBER SYSTEM

## 2.1 INTRODUCTION

We are familiar with the decimal number system which is used in our day-to-day work. In the decimal number system there are ten digits which are used to form decimal numbers. Ten separate symbols 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9 are used to represent ten decimal digits. A digital computer stores, understands and manipulates information composed of only zeros and ones. A programmer (or user) who works on a computer is allowed to use decimal digits; letters A, B, C, . . . Z, a, b, c, . . . z, usual special symbols, +, -, etc. for his convenience. The decimal digits, letters, symbols, etc. are converted to binary codes in the form of 0s and 1s within the computer. To understand the operation of a computer the knowledge of binary, octal and hexadecimal number system is essential. This chapter deals with these number systems.

## 2.2 DECIMAL NUMBER SYSTEM

As the ten fingers of our hands are the most convenient tools nature has given, human beings have always used them in counting. So the decimal number system followed naturally from their use. The base or radix of a number system is defined as the number of digits it uses to represent numbers in the system. Since the decimal number system uses ten digits, from 0 to 9, its base or radix is 10. The decimal number system is also called base-10 number system. The weight of each digit of a decimal number depends on its relative position within the number. This is explained by the following example.

**Example.** Take the decimal number 6498 as an example to explain the weight of each digit of the number.

$$\begin{aligned} 6498 &= 6000 + 400 + 90 + 8 \\ &= 6 \times 10^3 + 4 \times 10^2 + 9 \times 10^1 + 8 \times 10^0 \end{aligned}$$

The weight of each digit of a decimal number depends on its relative position within the number as explained below:

The weight of the 1st digit of the number from the right hand side = 1st digit  $\times 10^0$ .

The weight of the 2nd digit of the number from the right hand side = 2nd digit  $\times 10^1$ .

The weight of the 3rd digit of the number from the right hand side = 3rd digit  $\times 10^2$ .

The weight of the 4th digit of the number from the right hand side = 4th digit  $\times 10^3$ .

The above expressions can be written in general form as follows:  
The weight of the  $n$ th digit of the number from the right hand side

$$= n\text{th digit} \times 10^{n-1}$$

$$= n\text{th digit} \times (\text{Base})^{n-1}$$

The number system in which the weight of each digit depends on its relative position is called **positional number system**. The above form of general expression is true only for positional number system.

It is India that gave this positional method of expressing any number using ten symbols each symbol receiving a value of position as well as an absolute value. It was a profound idea. Its merit has been appreciated by a famous mathematician Marquis de L'Hopital.

## 2.3 BINARY NUMBER (OR BASE-2) SYSTEM

The base (or radix) of the binary number system is 2. It uses only two digits, 0 and 1. In short a binary digit is called a bit. The storing or computing electronic elements of a computer have only two stable states. The output of such an element at any time is either HIGH (5 volts) or LOW (0 volt). These are the only two stable states. There is no other state. These stable states can be represented by 1 and 0 respectively, that is HIGH is represented by 1 and LOW by 0. Due to this very limitation a computer can understand and manipulate information composed of only 0s and 1s. So all computers perform their internal operations and manipulations on binary digits. For the convenience of the users (programmers) they are allowed to use data and other information in the form of decimal digits, usual alphabetic special symbols. This information is converted to binary codes within the computer system. computer operates on binary bits. Thus we see that the knowledge of the binary number system is needed for those who want to understand the operating principle of a computer. It is not required by those who have to simply use a computer for their work.

In the decimal number system there is no difficulty in representing numbers up to 9. There is no symbol or digit to represent ten and therefore, it is represented by 10. It is a positional technique. Again, after 99 we have to represent hundred and utilizing positional technique it is written as 100. In the binary number system zero is represented by 0 and 100. In this way utilizing positional technique it is written as 10. Three is written as 11. Again four is represented by 100. In this way utilizing positional technique we proceed further. Thus it is seen that the binary number depends on its relative position within the number. The weight of each binary bit of a number depends on its relative position within the number. It has been explained by the following example.

**Example.** Take the binary number 1101 as an example to explain the weight of each bit of the number.

1101 (Binary Number)

$$= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$= 8 + 4 + 0 + 1 = 13 \text{ (Decimal Number)}$$

The weight of each bit of a binary number depends on its relative position within the number as explained below:  
The weight of the 1st bit of the binary number from the right hand side = 1st bit  $\times 2^0$   
The weight of the 2nd bit of the number from the right hand side = 2nd bit  $\times 2^1$

## NUMBER SYSTEM

The weight of the 3rd bit of the number from the right hand side = 3rd bit  $\times 2^2$ .  
The weight of the 4th bit of the number from the right hand side = 4th bit  $\times 2^3$ .  
The above expressions can be written in the form of a general expression given below.  
The weight of the  $n$ th bit of the number from the right hand side  
 $= n\text{th bit} \times 2^{n-1}$   
 $= n\text{th bit} \times (\text{Base})^{n-1}$

It is seen that this rule for a binary number is same as that for a decimal number. The above rule holds good for any other positional number system. The weight of a digit in any positional number system depends on its relative position within the number and the base of the number system. Table 2.1 shows binary equivalent of decimal numbers.

Table 2.1. Binary Equivalent of Decimal Numbers

Decimal Number	Binary Equivalent	Decimal Number	Binary Equivalent
0	0	11	1011
1	1	12	1100
2	10	13	1101
3	11	14	1110
4	100	15	1111
5	101	16	10000
6	110	31	11111
7	111	32	100000
8	1000	63	111111
9	1001	64	1000000
10	1010	128	10000000

## 2.4 CONVERSION OF A BINARY NUMBER TO DECIMAL NUMBER

To convert a binary number to its decimal equivalent we use the following expression:  
The weight of the  $n$ th bit of the number from the right hand side =  $n\text{th bit} \times 2^{n-1}$ .  
First we mark the bit position and then we give the weight of each bit of the number depending on its position. The sum of the weights of all bits gives the equivalent number. The following example illustrates the process.

**Example 1.** Convert the binary number 10 to its decimal equivalent.

$$\begin{aligned} \text{The first bit counting from the right hand side} &= 0 \\ &= 0 \times 2^{1-1} = 0 \times 2^0 \\ &= 0 \\ \text{The second bit from the right hand side} &= 1 \\ &= 1 \times 2^1 \\ &= 1 \times 2^1 + 0 \times 2^0 \\ &= 2 + 0 = 2 \end{aligned}$$

**Example 2.** Convert the binary number 101 to its decimal equivalent.

First we can write the binary number in the spread form and then show the bit position as shown below.

The binary number in the spread form = 1 0 1  
 Bit position from right hand side: 3rd 2nd 1st

The weight of each bit is assigned and added together to give the decimal equivalent.  
 101 (Binary Number) =  $1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$$= 4 + 0 + 1$$

$$= 5 \text{ (Decimal Number)}$$

**Example 3.** Convert the binary number 1010 to its decimal equivalent.

The binary number in the spread form = 1 0 1 0

Bit position from the right hand side: 4th 3rd 2nd 1st

$$1010 \text{ (Binary Number)} = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

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

$$= 10 \text{ (Decimal Number)}$$

**Example 4.** Convert the binary number 11001 to its decimal equivalent.  
 The binary number in the spread form = 1 1 0 0 1

Bit position from the right hand side: 5th 4th 3rd 2nd 1st

$$11001 \text{ (Binary Number)} = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$= 16 + 8 + 0 + 0 + 1$$

$$= 25 \text{ (Decimal Number)}$$

To indicate the base of a number the following technique can be used:  
 $(25)_{10} = (11001)_2$

The suffix 10 indicates that the number 25 is a decimal number.  
 The suffix 2 indicates that the number 11001 is a binary number.

## 2.5 CONVERSION OF A DECIMAL NUMBER TO A BINARY NUMBER

In a decimal number the 1st position from the right hand side is for 1s, 2nd for 10s, 3rd for 100s and so on. Similarly, in a binary number the 1st position from the right hand side is for 1, 2nd for 2, 3rd for 4, 4th for 8, 5th for 16 and so on. This fact is utilized to determine which multiples of 2 are present in a given decimal number. A method is to be developed to determine the binary equivalent of a decimal number. For example, 5 does not have any 2. Therefore, its binary equivalent is 101. Based on this concept, the conversion of a decimal number to binary number, the decimal number is divided successively by 2. The quotient and remainder are noted down at each stage. The process is repeated until the quotient becomes zero. The binary number equivalent to the decimal number is given by the following expression:

Binary Number =  $R_0 R_1 R_2 \dots R_{n-1} R_n$   
 where  $R_0, R_1, R_2, \dots, R_n$  are the remainders at 1st, 2nd, ... and nth stage respectively.

## NUMBER SYSTEM

**Example 1.** Convert the decimal number 41 to its binary equivalent.

Quotient	Remainder	Remark
$41 \div 2 = 20$	1 (LSB)	There are 20 twos and one 1.
$20 \div 2 = 10$	0	There are 10 fours and no 2.
$10 \div 2 = 5$	0	This is equivalent to dividing by 4.
$5 \div 2 = 2$	1	There are 5 eights and no 4.
$2 \div 2 = 1$	0	Equivalent to dividing by 8.
$1 \div 2 = 0$	1 (MSB)	There are two 16s and one 8.
		Equivalent to dividing by 16.
		There is one 32 and no 16.
		Equivalent to dividing by 32.
		There is no 64 and one 32.
		Equivalent to dividing by 64.

Therefore, 41 (Decimal Number) = 101001 (Binary Number). The first remainder is the least significant bit (LSB) and the last remainder the most significant bit (MSB).

**Example 2.** Convert the decimal number 73 to binary.

Quotient	Remainder
$73 \div 2 = 36$	1 (LSB)
$36 \div 2 = 18$	0
$18 \div 2 = 9$	0
$9 \div 2 = 4$	1
$4 \div 2 = 2$	0
$2 \div 2 = 1$	0
$1 \div 2 = 0$	1 (MSB)

The decimal number 73 = 1001001 (Binary Number).

Checking of the answer.

The decimal equivalent of the above binary number is given by  
 $1001001 = 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$ 
 $= 64 + 0 + 0 + 8 + 0 + 0 + 1 = 73 \text{ (Decimal Number)}$

**Example 3.** Convert the decimal number 153 to binary.

Quotient	Remainder
$153 \div 2 = 76$	1 (LSB)
$76 \div 2 = 38$	0
$38 \div 2 = 19$	0
$19 \div 2 = 9$	1
$9 \div 2 = 4$	1
$4 \div 2 = 2$	0

0  
1 (MSB)

$$10011001 \text{ (Binary Number)}$$
$$10011001 = 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$= 128 + 0 + 0 + 16 + 8 + 0 + 0 + 1$$

## 2.6 ADDITION OF BINARY NUMBERS

In the binary number system  $1 + 0 = 1$ . When 1 is added to 1, the sum is 0 with a carry of 1. If the sum is written upto 2 bits, it is equal to 10 (2 decimal). The Table 2.2 shows the rules for binary addition.

### Table 2.2 Binary Addition

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	10

Carry

When 1 is added to 1, the sum is 0 with a carry 1. This carry is added to the sum of the adjacent bits.

### Example 1

$$\begin{array}{r} 1001 \text{ (9 decimal number)} \\ + 0101 \text{ (5 decimal number)} \\ \hline 1110 \text{ (14 decimal number)} \end{array}$$

### Example 2

$$\begin{array}{r} 0111 \text{ (7 decimal number)} \\ + 0011 \text{ (3 decimal number)} \\ \hline 1010 \text{ (10 decimal number)} \end{array}$$

### Example 3

$$\begin{array}{r} 1010 \quad (10 \text{ decimal number}) \\ + 1101 \quad (13 \text{ decimal number}) \\ \hline 10111 \quad (23 \text{ decimal number}) \\ \uparrow \\ \text{Carry} \end{array}$$

## 2.7 BINARY SUBTRACTION

The following examples will illustrate binary subtraction.

### Example 1

Bottom	
↓	
1110	(14 decimal number)
- 0101	(- 5 decimal number)
1001	(9 decimal number)

### Example 2

1010	(10 decimal number)
- 0101	(- 5 decimal number)
<u>0101</u>	(5 decimal number)

### Example 3

1010	(10 decimal number)
- 0011	(- 3 decimal number)
<u>0111</u>	(7 decimal number)

### Example 4

1101	(13 decimal number)
-0111	(-7 decimal number)
<hr/>	
0110	(6 decimal number)

In the above examples smaller number has been subtracted from a larger number. Let us see what happens if a larger number is subtracted from a smaller number.

### Example 1

BOTTOM	
↓	
0101	(5 decimal number)
0111	(- 7 decimal number)
1110	(- 2 decimal number)

The result is not a simple representation of  $-2$  i.e.,  $-0010$ . The result is the 2's complement of 2. This will be explained in the next section.

### Example 2

Borrow	
↓	
0111	(7 decimal number)
- 1000	(- 8 decimal number)
1111	(-1 decimal number)

The result is 2's complement of 1.

## 2.8 USE OF COMPLEMENTS TO REPRESENT NEGATIVE NUMBERS

Most of the computers now perform subtraction using complemented number. It is economical because subtraction can also be performed using the same electronic modes for the representation of decimal numbers. In the binary number system there are two types of complement: 1's complement and 2's complement. To represent a negative binary number 2's complement is most widely used at present. To understand 1's and 2's complement in the binary number system one should first understand 9's and 10's complements in the decimal number system.

### 2.8.1 9's Complement

To form the 9's complement of a decimal number each digit of a decimal number is subtracted from 9. The result so obtained is known as 9's complement of the number. For example, the 9's complement of 37 is  $(99 - 37) = 62$ . The 9's complement of 235 is  $(999 - 235) = 764$ .

### 2.8.2 10's Complement

The 10's complement of a decimal number is equal to the 9's complement of the number plus 1.

The 10's complement of a decimal number = Its 9's complement + 1.

The 10's complement of 37 =  $62 + 1 = 63$ .

The 10's complement of 235 =  $764 + 1 = 765$ .

Now let us examine the sum of a decimal number and its 10's complement.

**Example 1**

$$\begin{array}{r} 37 \quad \text{(decimal number)} \\ + 63 \quad \text{(its 10's complement)} \\ \hline 100 \end{array}$$

↑ Ignore carry

In the above example the given decimal number is of two digits. If the sum of a number and its 10's complement is considered only up to two digits, the sum is equal to zero. In other words the sum of a number and its 10's complement is equal to zero if the last stage is neglected.

**Example 2**

$$\begin{array}{r} 235 \quad \text{(decimal no.)} \\ + 765 \quad \text{(its 10's complement)} \\ \hline 1000 \end{array}$$

↑ Ignore carry

The decimal number 235 is of three digits. If the sum of a decimal number and its 10's complement is considered only up to three digits, the sum is equal to zero. Therefore, the 10's complement represents the negative value of the number. The 10's complement of a decimal number = - decimal number.

### 2.8.3 Addition in 10's Complement System

If 10's complement of a number is added to any number it is equivalent to its subtraction from that number. 10's complement subtraction is used in computers which employ BCD codes for the representation of decimal numbers.

**Example 1.** Add 86 and  $(-21)$ .

The 9's complement of 21 = 78

The 10's complement of 21 =  $78 + 1 = 79$

86

$$\begin{array}{r} + 79 \quad \text{(10's complement of 21)} \\ \hline 165 \end{array}$$

↑ Ignore carry

If the carry of the last stage is dropped, the result is correct.

**Example 2.** Add 59 and  $(-84)$

The 9's complement of 84 = 15

The 10's complement of 84 =  $15 + 1 = 16$

59

$$\begin{array}{r} + 16 \quad \text{(10's complement of 84)} \\ \hline 75 \quad \text{(10's complement of 25)} \end{array}$$

There is no carry and the result is in 10's complement. It is the 10's complement of 25. To get the correct result take 9's complement of the result, add 1 and put a - sign before.

The 9's complement of 75 = 24

The 10's complement of 75 =  $24 + 1 = 25$

Result = - 25, which is the correct result.

**Example 3.** Add  $(-26)$  and  $(-43)$

The 9's complement of 26 = 73

The 10's complement of 26 =  $73 + 1 = 74$

The 9's complement of 43 = 56

The 10's complement of 43 =  $56 + 1 = 57$

$$\begin{array}{r} 74 \\ + 57 \\ \hline 131 \end{array}$$

↑ Ignore carry

The 10's complement of 31 = 69. The correct result is - 69.

**Example 4.** Add 34 and 58

$$\begin{array}{r} 34 \\ + 58 \\ \hline 92 \end{array}$$

As the both numbers are positive, they are simply added. The 10's complement required. The result is positive and correct. There is no carry at the last stage.

### Conclusion

Whether the result is positive or negative the carry of the last stage is to be ignored. If the result is positive it is correct as usual. If the result is negative, it is in 10's complement form. Computer can represent signed numbers, and hence recognize whether the result is positive or negative. If the result is negative, computer understands that it is in the complement form. So it converts it to get the result in the usual form. This has been explained by discussing 2's complement which is similar to 10's complement in the decimal system.

### 2.8.4 1's Complement

The 1's complement in the binary number system is similar to 9's complement in the decimal system. To obtain 1's complement of a binary number each bit of the binary number is subtracted from 1. For example, the 1's complement of the binary number 010 is 101. Thus 1's complement of 1110 is 0001. Thus 1's complement of a binary number may be found simply changing each 1 to a 0 and each 0 to a 1.

**Example 1.** Find 1's complement of the binary number 101100.

The 1's complement of the binary number 101100.

**Example 2.** Find 1's complement of the binary number 0000.

The 1's complement of the binary number 0000.

**Example 3.** Find 1's complement of the binary number 1111.

The 1's complement of the binary number 1111.

### 2.8.5 2's Complement

The 2's complement in the binary number system is similar to 10's complement in the decimal number system. The 2's complement of a binary number is equal to the 1's complement of the number plus one.

The 2's complement of a binary number = 1's complement + 1.

**Example 1.** Find 2's complement of the binary number 1011.

The 2's complement of the binary number 1011.

**Example 2.** Find 2's complement of the binary number 101100.

The 2's complement of the binary number 101100.

**Example 3.** Find 2's complement of the binary number 0000.

The 2's complement of the binary number 0000.

Let us examine the sum of a binary number and its 2's complement.

**Example 1.** Add the binary number 1100 and its 2's complement.

The binary number = 1100

Its 1's complement = 0011

Its 2's complement = 0011 + 1 = 0100

The number + its 2's complement = 1100

$$\begin{array}{r} 1100 \\ + 0100 \\ \hline 10000 \end{array}$$

↑ Ignore carry

If the carry of the last stage is neglected the sum of a binary number and its 2's complement is equal to zero. In the above example the binary number is of 4 bits. If a 4-bit processor is used and the last carry is neglected, the sum will be considered only upto 4 bits. For an 8-bit processor the binary number and its 2's complement is written upto 8 bits as shown below in the examples 2 and 3.

**Example 2.** Add the binary number 1011 and its 2's complement.

$$\begin{array}{r} \text{The binary number} \\ 1011 \\ \text{Its 1's complement} \\ 0100 \\ \text{Its 2's complement} \\ 0100 + 1 \\ = 0101 \end{array}$$

$$\begin{array}{r} \text{The given binary number} \\ 1011 \\ + \text{Its 2's complement} \\ 0101 \\ \hline 10000000 \end{array}$$

↑ Ignore carry

The sum only upto 8 bits is equal to zero.

**Example 3.** Add 5 and (-5).

5 (decimal) = 00000101 (binary)

-5 = Its 2's complement = 1111010 + 1 = 1111011

$$\begin{array}{r} 5 = 00000101 \\ + (-5) = 1111011 \\ \hline 100000000 \end{array}$$

↑ Ignore carry

If the sum is considered only upto 8 bits, it is equal to zero.

### 2.8.6 Binary Subtraction Using 2's Complement

The addition of 2's complement of a number is equivalent to its subtraction. This will be clear from the following example.

**Example 1.** Subtract 2 from 6.

Simple binary subtraction

$$\begin{array}{r} 0110 \text{ (6 decimal)} \\ - 0010 \text{ (-2 decimal)} \\ \hline 0100 \text{ (4 decimal)} \end{array}$$

### Subtraction using 2's complement

1's complement of 0010 (2 decimal) = 1101

2's complement of 0010 (2 decimal) = 1101 + 1 = 1110

$$\begin{array}{r}
 0110 \quad (6 \text{ decimal}) \\
 + 1110 \quad (+ 2's \text{ complement of } 2) \\
 \hline
 1\ 0100 \quad (4 \text{ decimal}) \\
 \leftarrow \text{Ignore carry}
 \end{array}$$

The carry of the last stage is to be neglected if we are using 2's complement notation.

**Example 2.** Subtract 3 from 5

$$\begin{array}{l}
 1's \text{ complement of } 0011 \text{ (3 decimal)} = 1100 \\
 2's \text{ complement of } 0011 \text{ (3 decimal)} = 1100 + 1 \\
 = 1101
 \end{array}$$

$$\begin{array}{r}
 0101 \quad (5 \text{ decimal}) \\
 + 1101 \quad (+ 2's \text{ complement of } 3) \\
 \hline
 1\ 0010 \quad (2 \text{ decimal}) \\
 \leftarrow \text{Ignore carry}
 \end{array}$$

The carry of the last stage is neglected.

## 2.8.7 Representation of Signed and Unsigned Numbers

In the decimal system we put + or - sign before a number to represent its sign. In the binary system, such notation can not be employed and therefore, a different technique has been adopted. To represent positive sign a 0 is placed before the binary number. For example, 9 is represented by 0 1001. To represent negative number a 1 is placed before the number. For example, -9 will be represented as 1 1001. There is only one way to represent a positive number. But there are three different ways to represent a negative number. These are:

- (1) Signed-Magnitude Representation
- (2) Signed-1's Complement Representation
- (3) Signed-2's Complement Representation

The representation of -9 is shown below:

**Signed-magnitude representation.** 1 1001

**Signed-1's complement representation.** 1 0110

**Signed-2's complement representation.** 1 0111

Hence, 9 is represented by 4 binary bits and a separate bit is used to represent sign. In a computer the MSB can be used to represent the sign of the number. For example, as the computer will represent -9 as shown below, 7 bits are used to represent the number. The MSB is used to represent the sign of the number.

**Signed-magnitude representation** 1 0001001

**Signed-1's complement representation** 1 1110110

**Signed-2's complement representation** 1 1110111

When all the bits of the computer word are used to represent the number and no bit is used for sign representation it will be called unsigned representation of the number.

## 2.8.8 Addition in Signed 2's Complement

The 2's complement in the binary number system is similar to 10's complement in the decimal system. We have already discussed the rules for addition in 10's complement. Those rules also hold good for 2's complement. A sign bit adjacent to the most significant bit is used to distinguish a positive number from a negative number. A 1 in the sign bit is used to represent a negative number and a 0 in the sign bit a positive number. The use of sign bit clearly indicates whether the result is positive or negative. The following examples will show the different situations in 2's complement addition.

**Example 1.** Addition of two positive numbers.

Add 5 and 3.

Normal Notation	Computer Notation
+ 0101 (+ 5 decimal)	0 0101 (+ 5 decimal with sign bit)
+ 0011 (+ 3 decimal)	0 0011 (+ 3 decimal with sign bit)
+ 1000 (+ 8 decimal)	0 1000 (+ 8 decimal)

In computer representation sign bit has also been included. The first bit from the left or fifth bit from the right is the sign bit. The addition is also performed on the sign bit. In the above example the sum 1000 is correct in the binary form and it is equal to 8 decimal. A 0 in the sign bit indicates that the result is positive. There is no carry at the last stage.

**Example 2.** Addition of a positive and a negative number, the positive number being greater than the negative number.

Add 9 and (-4).

The 1's complement of 0100 (4 decimal) = 1011

The 2's complement of 0100 = 1011 + 1 = 1100

Normal Notation	Computer Notation
+ 1001 (+ 9 decimal)	0 1001 (+ 9 decimal with sign bit)
- 0100 (- 4 decimal)	1 1100 (2's decimal of 4 with sign bit)
+ 0101 (5 decimal)	1 0101 (+ 5 decimal)

← Neglect carry

The sum 0101 is correct in the binary form and equal to 5 decimal. The sign bit 0 indicates the sum is positive. There is no carry at the last stage. The carry is to be dropped.

**Example 3.** Addition of a positive and negative number, the negative number being greater than the positive number.

Add (-9) and 3.

The 1's complement of 1001 (9 decimal) = 0110

The 2's complement of 1001 = 0110 + 1 = 0111

- 9	1 0111 (2's complement of 9 with sign bit)
+ 3	0 0011 (+ 3 with sign bit)
- 6	1 1010 (2's complement of 6 with sign bit)

The sum is negative as indicated by the sign bit. It is in 2's complement. There is no carry at the last stage.

Take 1's complement of the result, add 1 and put a - sign before it. In the above example, the result in 2's complement is 1010. The correct result will be:

$$\frac{1}{-0.110} \quad (+ \text{ of decimal})$$

The 1's complement of 1100 (12 decimal) = 0011

0010 =

The 1's complement of 0010 (2 decimal) = 1101

The 2's complement of 0010

12 10100 (2's complement of 10 with 4 bits)

complement of 12 with sign bit

- 14	1 1 0010	(2 <sup>nd</sup> complement of 14 with sign bit)
------	----------	--

Carry = 1

The sum is negative and it is in  $2^n$  complement. There is a carry through the end. It is to be dropped.

The above result is in  $2^{\text{nd}}$  complement. The correct result can be obtained as  $2^{\text{nd}}$  complement of 6010 (result) = 1110

14 (decimal)

The carry of the last stage is to be dropped. When the sum is positive, it is in the binary form. When the sum is negative it is in  $2^n$ 's complement. The sign bit indicates whether the sum is positive or negative. When the sum is in  $2^n$ 's complement, the sign bit indicates the correct form for display by the computer very easily. This is true only when there is no overflow. The case of overflow has been discussed later on in this chapter in Sec. 2.10.

In the above examples we have taken a separate sign bit. In a computer the most significant bit may be reserved for sign bit; and the rest of the bits of the data word will represent the magnitude of a number. Consider an 8-bit computer. The magnitude of the number is represented by 7 bits. The MSB represents the sign of the number. If the sign bit is 0, the number is positive; if the sign bit is 1, the number is negative. The negative numbers are represented in  $2^n$ 's complemented form. The following example will illustrate the addition of signed binary numbers.

**Example 5.** Add +14 and +9

**Example 5.** Add  $+14$  and  $+9$

$$+ 14 \quad 00001110$$
$$\frac{00001001}{6} + \frac{\quad}{6}$$

+ 23 00010111

Sign bit is 0, so the sum is **positive**.

11100000 14

**0 111011** (2's complement of 0 with sign bit)

10000101 (+ b decimal)      g +

Sign bit is 0, so the result is positive  
Carry = 1, it is neglected.

**Example 7.** Add  $(-14)$  and  $+9$ 

1110010 (2's complement of 14 with sign bit)

000001001 + 0

**IIIb1** (2's complement of **b** with sign bit)

Flag bit is 1, so the result is negative.  
00000100 It's complement of the result.

**1 Add 1**

00000101 (5 decimal).

Prefix = 1s to show the result in usual standard binary form

**Example 8.** Add  $(-14)$  and  $(-9)$ .

14 11110010 (2's complement of 14 with sign bit)

**1110111** (2's complement of 0 with sign bit).

111101001 (2's complement of 23 with sign bit)

Sign bit is 1, so the result is negative.  
Carry is 1, it is neglected.

00010110 It's complement of the result.

**† 1 Add 1**

— 00010111 ( = 23 decimal)

**Prefix** = is to show the correct result in usual standard binary form.

The above examples show that the addition and subtraction in  $2^n$ 's complement is versatile. The sign bit clearly indicates whether the result is positive or negative. If it is negative it means that it is in  $2^n$ 's complement. Then the computer converts it to a standard form to show it on the screen. The case of overflow is discussed in Sec. 2.10.

## 2.9 CONVERSION OF A BINARY FRACTION TO A DECIMAL FRACTION

In the decimal number system the weights of the digits which come after the decimal point, are represented as:

$$0.635 = 0.6 + 0.03 + 0.005$$

$$= 6 \times \frac{1}{10} + 3 \times \frac{1}{100} + 6 \times \frac{1}{1000}$$

$$= 6 \times 10^{-1} + 3 \times 10^{-2} + 5 \times$$

Similarly in the binary number system the weights of the binary bits which the binary point can be expressed as

$$\begin{aligned} 0.1101 &= 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} \\ &= 1 \times \frac{1}{2} + 1 \times \frac{1}{4} + 0 \times \frac{1}{8} + 1 \times \frac{1}{16} \\ &= 0.5 + 0.25 + 0 + 0.0625 \\ &= 0.8125 \text{ (decimal)} \end{aligned}$$

**Example 1.** Convert the binary fraction 0.10111 to decimal fraction.

$$\begin{aligned} 0.10111 &= 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-5} \\ &= 1 \times \frac{1}{2} + 0 \times \frac{1}{4} + 1 \times \frac{1}{8} + 1 \times \frac{1}{16} + 1 \times \frac{1}{32} \\ &= 0.5 + 0 + 0.125 + 0.0625 + 0.03125 \\ &= 0.71875 \text{ (decimal)} \end{aligned}$$

**Example 2.** Convert the binary real number 1101.1010 to a decimal real number. A binary real number consists of two parts an integer and a fraction. The equivalent is determined for both integer as well as fraction and these are added to give equivalent decimal real number.

$$\begin{aligned} 1101.1010 &= (1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0) + (1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4}) \\ &= (8 + 4 + 0 + 1) + (0.5 + 0 + 0.125 + 0) \\ &= 13.625 \text{ (decimal)} \end{aligned}$$

## 2.10 CONVERSION OF A DECIMAL FRACTION TO A BINARY FRACTION

To convert a decimal fraction to its binary equivalent a technique of successive multiplication by 2 is used. The integer part is noted down after the multiplication by 2 at each step and the remainder new fraction is used for the multiplication by 2 at each step. Following example will illustrate the procedure.

**Example 1.** Convert the decimal fraction 0.6125 to an equivalent binary fraction.

Fraction	Fraction $\times 2$	Integer Part	Remainder New Fraction	Integer
0.6125	1.225	1	0.225	1 (MSB)
0.6125	1.25	1	0.25	
0.25	0.50	0	0.50	
0.50	1.00	1	0.00	
				1 (LSB)

0.6125 (decimal) = 0.1101 (Binary)

**Example 2.** Convert the decimal fraction 0.635 to its binary equivalent.

Fraction	Fraction $\times 2$	Integer Part	Remainder New Fraction	Integer
0.635	1.27	1	0.27	1 (MSB)
0.27	0.54	0	0.54	
0.54	1.08	1	0.08	
				1 (LSB)

0.08	0.16	0.16	0
0.16	0.32	0.32	0
0.32	0.64	0.64	0
0.64	1.28	0.28	1 (LSB)

In this example it is seen that the fraction has not become zero, and the process will continue further. For such a case an approximation is made. For this example, we may take the result up to 6th binary bit after the binary point.

$$0.635 \text{ (decimal)} \approx 0.1010001 \text{ (binary)}$$

**Example 3.** Convert the decimal real number 12.625 to an equivalent binary real number.

For the above decimal real number the binary equivalent is obtained for both integer as well as fraction separately.

12 (decimal) is first converted to its binary equivalent.

$$\begin{aligned} \text{Quotient} & & \text{Remainder} \\ 12 \div 2 &= 6 & 0 \text{ (LSB)} \\ 6 \div 2 &= 3 & 0 \\ 3 \div 2 &= 1 & 1 \\ 1 \div 2 &= 0 & 1 \text{ (MSB)} \end{aligned}$$

$$12 \text{ (decimal)} = 1100 \text{ (binary)}$$

Then the decimal fraction 0.625 is converted to its binary equivalent.

Fraction	2 $\times$ Fraction	Remainder New Fraction	Integer
0.625	1.25	0.25	1 (MSB)
0.25	0.50	0.50	0
0.50	1.00	0.00	1 (LSB)

$$0.625 \text{ (decimal)} = 0.101 \text{ (binary)}$$

Now adding the binary equivalents of 12 and 0.625, we get

$$12.625 \text{ (decimal)} = 1100.101 \text{ (binary)}$$

## 2.11 BINARY CODED DECIMAL (BCD CODES)

The BCD is the simplest binary code to represent a decimal number. In BCD code a decimal number is represented by four binary bits. For example, 3 is represented by 0011. If a decimal number consists of two or more than two digits, each decimal digit is individually represented by its 4-bit binary equivalent. For example, 66 is represented by 0101 0110.

Numbers are usually represented by some sort of binary codes. There is a difference between a binary equivalent of a decimal number and the binary code of a decimal number. For example, the binary equivalent of the decimal number 43 is 101011, but it is represented in BCD code as 0100 0011.

Table 2.3 shows the standard BCD codes for one-digit and two-digit decimal numbers.

Decimal Number	Standard BCD code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	0001 0000
11	0001 0001
12	0001 0010
13	0001 0011
14	0101 0110
15	0101 0111
16	1000 0100
17	1001 0001

In the standard BCD code the weights of four binary bits which represent an individual digit are 8, 4, 2, 1. At present modern computers perform subtraction using complement 101. There is difficulty in forming complements when numbers are represented by single digits, each digit is represented individually by four binary bits. For example, 5 is represented by 0101 1110, 3E7 by 0011 1011 0111. Table 2.4 shows hexadecimal numbers and their binary representations. The binary representation of a hexadecimal number is also called binary coded hexadecimal number. The reason for using 4 binary bits to write a hexadecimal digit is that the largest hexadecimal digit 'F' requires only 4 binary bits to write. For example, 5 (decimal) = 1000 (8 decimal) in excess-3 code. 253 (decimal) = 0101 1000 0110. The drawback of this code is that it is not a well-defined code, that is the sum of weights of binary bits is not equal to the corresponding decimal digit.

Another BCD code is 2, 4, 2, 1 code. It is a weighted code and it has complements into or out of a digital system. Electronic calculators, digital voltmeters, frequency counters used in early computers. Modern computers do not use BCD numbers as they have to process names and other nonnumeric data.

In addition to difficulty in forming complements for binary subtraction there is difficulty in performing addition in standard BCD. If the result lies in the range 10 to 15 there is carry from the 4th bit of any BCD digit, a correction of +6 has to be made to get the correct result. This will be clear from the following example.

## NUMBER SYSTEM

Example 1. Add 8 and 5.

	BCD
8	1000
+ 5	0101
13	1101
	Incorrect BCD
	+ 0110
	10010011
	Correct BCD 13

Example 2. Add 8 and 9.

	BCD
8	1000
+ 9	1001
17	00010001
	Incorrect BCD
	+ 0110
	00010111
	Correct BCD 17

## 2.12 HEXADECIMAL NUMBER SYSTEM

The hexadecimal number system is now extensively used in computer industry. The radix (base) is 16. Its digits from 0 to 9 are same as those used by decimal number system. Its hexadecimal number system 10 is represented by A, 11 by B, 12 by C, 13 by D, 14 by E, and 15 by F. The decimal number 16 is represented by 10, 17 by 11 and so on.

A hexadecimal digit is represented by four binary bits. For example, 5 is represented by 0101, A by 1010 and D by 1101. If a hexadecimal number consists of two or more than two digits, each digit is represented individually by four binary bits. For example, 5B is represented by 0101 1011, 3E7 by 0011 1011 0111. Table 2.4 shows hexadecimal numbers and their binary representations. The binary representation of a hexadecimal number is also called binary coded hexadecimal number. The reason for using 4 binary bits to write a hexadecimal digit is that the largest hexadecimal digit 'F' requires only 4 binary bits for its representation.

Let us examine the binary equivalent of a decimal number and its hexadecimal representation. The binary equivalent of the decimal number 94 is 1011110. The hexadecimal equivalent of 94 (decimal) is 5E. The binary representation of 5E is 0101 1110. If we neglect the leading zero, that is zero in MSB position, both binary representations are identical.

Table 2.4. Hexadecimal Numbers and their Binary Representations

Decimal Number	Hexadecimal Number	Binary Coded Hexadecimal Number
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101

6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

## 2.12.1 Hexadecimal Versus BCD

From Table 2.4 it is clear that the hexadecimal system utilizes the full capacity of binary bits, whereas BCD codes do not utilize the same. The BCD codes do not utilize binary codes from 1010 to 1111. In the hexadecimal system an 8-bit word can represent the hexadecimal is a compact form of representation, and it occupies less memory thereby reducing the hardware cost. The arithmetic operations are also simpler in hexadecimal system.

When a computer gives hardware or software error it may be required to examine the information stored in the memory. The memory stores information in binary form which is very difficult to examine. Therefore, the stored information is printed out in hexadecimal form which is very compact and easy to examine. Computers which work in binary form dump the information in hexadecimal as printouts that are very compact. Listings are also printed in hexadecimal.

Usually, the input data and the result (output) are in the decimal form. Before a processor (hexadecimal or octal) performs arithmetic operations the input data have to be converted into binary form (hexadecimal or octal). Again results which are in the binary form are converted to decimal form. The binary system will be efficient for small input requirements and a large number of computations. A processor working with small input requirements are required. The computers using BCD codes perform arithmetic operations on decimal data (BCD codes) and eliminate the requirement for conversion to binary and back to decimal.

Many decimal fractions do not have exact binary equivalents. In some business applications dealing with accounting precise calculations are desired. Therefore, BCD codes are desired for the exact representation of decimal numbers including the digits after the decimal point. But BCD arithmetic takes more time for execution and hence it is slower than binary arithmetic.

arithmetic. Thus the price paid for precise decimal fractions is less efficient use of memory and slower processing.

Some computers and all calculators employ BCD codes. Most of the modern computers use hexadecimal system. Some large computers may have the hardware facility for arithmetic calculations with both binary (hex) as well as BCD.

## 2.12.2 Conversion of a Hexadecimal Number to a Decimal Number

For the conversion of a hexadecimal number to an equivalent decimal number the following well known expression for the weights of digits of the number will be used:

The weight of  $n$ th digit of the number from the right hand side =  $n$ th digit  $\times$  (Base) $^{n-1}$ .

The following examples will illustrate the procedure.

**Example 1.** Convert the hexadecimal number 4B8 to its equivalent decimal number.

8 is the 1st digit of the number from the right hand side, therefore, its weight is  $8 \times 16^0$ .

$8 \times 16^0$ .

B is the 2nd digit of the number from the right hand side, therefore, its weight is  $B \times 16^1$ .

$B \times 16^1$ .

4 is the 3rd digit of the number from the right hand side, therefore, its weight is  $4 \times 16^2$ .

$4 \times 16^2$ .

Therefore,  $4B8$  (hex) =  $4 \times 16^2 + B \times 16^1 + 8 \times 16^0$

=  $4 \times 256 + 11 \times 16 + 8 \times 1$

=  $1024 + 176 + 8$

=  $1208$  (decimal).

**Example 2.** Convert the hexadecimal number 6E to its decimal equivalent.

6E (hex)

=  $6 \times 16^1 + E \times 16^0$

=  $6 \times 16 + 14 \times 1$

=  $96 + 14$

=  $110$  (decimal).

**Example 3.** Convert the hexadecimal number 2B6D to its equivalent decimal number.

2B6D (hex)

=  $2 \times 16^3 + B \times 16^2 + 6 \times 16^1 + D \times 16^0$

=  $2 \times 4096 + 11 \times 256 + 6 \times 16 + 13 \times 1$

=  $8192 + 2816 + 96 + 13$

=  $11117$  (decimal).

## 2.12.3 Conversion of a Hexadecimal Fraction to a Decimal Fraction

In the hexadecimal system the weights of the hexadecimal digits after the hexadecimal point are as follows:

0.5A6B

=  $5 \times 16^{-1} + A \times 16^{-2} + 6 \times 16^{-3} + B \times 16^{-4}$

=  $0.3125 + 0.0390625 + 0.0014648437 + 0.00016784667$

=  $0.35319519037$  (hexadecimal).

## 2.12.4 Conversion of a Decimal Number to a Hexadecimal Number

For the Conversion of a decimal number to an equivalent hexadecimal number, the decimal number is divided by 16 successively.

**Example 1.** Convert the decimal number 67 to an equivalent hexadecimal number.

**Quotient**  
 $67 \div 16 = 4$   
 $4 \div 16 = 0$   
 The decimal number 67 = 43 (hexadecimal).

**Example 2.** Convert the decimal number 952 to its hexadecimal equivalent.

**Quotient**  
 $952 \div 16 = 59$   
 $59 \div 16 = 3$   
 $3 \div 16 = 0$   
 The decimal number 952 = 3B8 (hexadecimal).

## 2.12.5 Conversion of a Decimal Fraction to a Hexadecimal Fraction

For the conversion of decimal fraction to its equivalent hexadecimal fraction, the technique of repeated multiplication by 16 is used. The integer part is noted down after multiplication and the new remainder fraction is used for multiplication at the next step.

**Example.** Convert the decimal fraction 0.62 to its equivalent hexadecimal fraction.

Fraction	Fraction $\times 16$	Remainder	Integer
0.62	9.92	0.92	9 (MSD)
0.92	14.72	0.72	14 = E
0.72	11.52	0.52	11 = B
0.52	8.32	0.32	8
0.32	5.12	0.12	5
0.12	1.92	0.92	1 (LSD)

The process will further continue. Therefore, the result has been taken upto 6 hexadecimal point.

The decimal fraction 0.62 = 0.9EB851 (hex.) approximately.

## 2.12.6 Conversion of a Binary Number to a Hexadecimal Number

The conversion of binary number to hexadecimal number uses a very simple technique.

The base of the hexadecimal number system is 16. To convert a binary number to hexadecimal, the binary number is divided into groups of 4 bits each. Each group of 4 bits is converted to its hexadecimal equivalent.

**Example 1.** Convert the binary number 01101110 to its equivalent hexadecimal number.

$= (0110)(1110)$   
 $= 6E$  (hex.).

**Example 2.** Convert the binary number 110101101 to its equivalent hexadecimal number.

The formation of groups of 4 bits each in an integer binary number is made from right side to left.  
 $= (11)(0100)(1101)$

## 2.13 OCTAL NUMBER SYSTEM

In the above expression we see that the group of the most significant binary bits contains only 2 binary bits. This can be extended to 4 binary bits by adding zeros in MSB positions. MSBs are extended by adding zeros the number remains unaffected. Therefore, the given number may be grouped as follows:

$(1101001101)_2$   
 $= (0011)(0100)(1101)$   
 $= 34D$  (hex.).

**Example 3.** Convert the binary real number 1011100 . 1000101 to its equivalent hexadecimal real number.

In case of binary fraction, the formation of grouping of binary bits which are after the binary point, is made from left to right. For the integer part group is made from right to left.  
 $1011100.1000101$   
 $= (101)(1100)(1000)(101)$   
 $= (0101)(1100)(1000)(101)$   
 $= 5C.8A$  (hex.).

## 2.12.7 Conversion of a Hexadecimal Number to a Binary Number

To convert a hexadecimal number to its equivalent binary number each digit of the given hexadecimal number is converted to its 4-bit binary equivalent.

**Example 1.** Convert the hexadecimal number 6B9 to its equivalent binary number.

$(6B9)_{16}$   
 $= (0110)(1011)(1001)$   
 $= (011010111001)_2$   
 $= (11010111001)_2$

**Example 2.** Convert the real hexadecimal number 6D.3A to its equivalent binary number.

$(6D.3A)_{16}$   
 $= (0110)(1101)(0011)(1010)$   
 $= (0110110100111010)_2$   
 $= (11011010011101)_2$

## 2.13 OCTAL NUMBER SYSTEM

The base of the octal number system is 8. This system is also used in computer industry. It uses eight digits 0, 1, 2, 3, 4, 5, 6 and 7. The decimal number 8 is represented by 10, 9 by 11, 10 by 12 and so on in the octal number system. An octal number is represented by a group of three binary bits. For example, 4 is represented by 100, 6 by 110 and 7 by 111. The reason for using 3 binary bits to represent an octal number is that the largest octal digit 7 which can be represented by only 3 binary bits i.e.,  $7 = 111$ . If an octal number contains 10 or more than two digits, each digit is individually represented by a group of three binary bits. For example, 48 is represented by 100 110 and 884 by 011 101 100. Table 2.6 shows octal numbers and their binary representations. The binary representation of an octal number is an called binary coded octal number.

Let us examine the binary equivalent and binary representations of a decimal number (octal and hexadecimal). The binary equivalent of the decimal number 48 is 101111. The octal number 48 is 60 (octal) and 60 (octal) respectively. The 1 can be represented by binary as 00101101, and 0 (octal) as 101101. If we neglect zeros in MSB positions, all the three binary representations are same.

Table 2.5 Octal Number and their Binary Representations

Decimal Number	Octal Number	Binary Coded Octal Number
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	001 000
9	11	001 001
10	12	001 010
11	13	001 011
12	14	001 100
13	15	001 101
14	16	001 110
15	17	001 111
16	20	010 000
17	21	010 001
18	22	010 010
19	23	010 011
20	24	010 100
21	25	010 101
22	26	010 110
23	27	010 111
24	30	011 000
25	31	011 001
26	32	011 010
27	33	011 011
28	34	011 100
29	35	011 101
30	36	011 110
31	37	011 111
32	40	100 000
33	41	100 001
34	42	100 010
35	43	100 011
36	44	100 100
37	45	100 101
38	46	100 110
39	47	100 111

### 2.13.1 Conversion of an Octal Number to a Decimal Number

For the conversion of a number from any number system to decimal number system make the use of the following well known expression:

The weight of the  $n$ -th digit of the number from right hand side =  $n$ th digit  $\times$  (Base) <sup>$n$</sup>

**Example 1.** Convert the octal number 56 to its equivalent decimal number.  
As the base of the octal number system is 8, applying the general rule of conversion we get

$$\begin{aligned} 56 (\text{octal}) &= 5 \times 8^1 + 6 \times 8^0 \\ &= 40 + 6 \\ &= 46 (\text{decimal}) \end{aligned}$$

**Example 2.** Convert the octal number 364 to its decimal equivalent.

$$\begin{aligned} 364 (\text{octal}) &= 3 \times 8^2 + 6 \times 8^1 + 4 \times 8^0 \\ &= 3 \times 64 + 6 \times 8 + 4 \times 1 \\ &= 192 + 48 + 4 \\ &= 244 (\text{decimal}) \end{aligned}$$

### 2.13.2 Conversion of an Octal Fraction to a Decimal Fraction

In the octal system the weight of the octal digits after the octal point are as follows:

$$0.563 (\text{octal}) = 5 \times 8^{-1} + 6 \times 8^{-2} + 3 \times 8^{-3}$$

### NUMBER SYSTEM

$$\begin{aligned} &= 0.6925 + 0.09375 + 0.005859375 \\ &= 0.724609375 (\text{decimal}) \end{aligned}$$

### 2.13.3 Conversion of a Decimal to an Octal Number

For the conversion of a decimal number to an equivalent octal number, the technique of repeated division by 8 is used.

**Example 1.** Convert the decimal number 62 to its equivalent octal number.

Quotient	Remainder
$62 \div 8 = 7$	6 (LSD)
$7 \div 8 = 0$	7 (MSD)
62 (decimal)	= 76 (octal)

**Example 2.** Convert the decimal number 958 to its equivalent octal number.

Quotient	Remainder
$958 \div 8 = 119$	6 (LSD)
$119 \div 8 = 14$	7
$14 \div 8 = 1$	6
$1 \div 8 = 0$	1 (MSD)
958 (decimal)	= 1676 (octal)

Checking the answer:

$$\begin{aligned} 1676 (\text{Octal}) &= 1 \times 8^3 + 6 \times 8^2 + 7 \times 8^1 + 6 \times 8^0 \\ &= 512 + 384 + 56 + 6 \\ &= 958 (\text{decimal}) \end{aligned}$$

### 2.13.4 Conversion of a Decimal Fraction to an Octal Fraction

For the conversion of a decimal fraction to its equivalent octal fraction the technique of repeated multiplication by 8 is used. The integer part is noted down and new remainder is used for the multiplication at the next stage.

**Example 1.** Convert the decimal fraction 0.96 to its equivalent octal fraction.

Fraction	Fraction $\times 8$	Remainder Fraction	New Integer
0.96	7.68	0.68	7 (MSD)
0.68	5.44	0.44	5
0.44	3.52	0.52	3
0.52	4.16	0.16	4
0.16	1.28	0.28	1 (LSD)

The process will continue further so we may take the result upto 5 places of octal point.

$$0.96 (\text{decimal}) = 0.75341 (\text{octal}) \text{ approximately}$$

### 2.13.5 Conversion of a Binary Number to an Octal Number

The octal number system is a base-8 system. For binary to octal conversion, groups of three binary bits each are formed in the binary number. After forming the groups, each group of 3 binary bits is converted to its octal equivalent.

**Example 1.** Convert the binary number 101110 to its equivalent octal number. The formation of groups of 3 bits each in an integer binary number is made from left to left.

$$(101110)_2 = (101)(110) \\ = 56 \text{ (octal)}$$

**Example 2.** Convert the binary number 1101011 to its equivalent octal number.

$$(1101011)_2 = (1)(101)(011)$$

As the leftmost group consists of only one binary bit, this group is extended by zeros in MSBs.

$$(1101011)_2 = (001)(101)(011) \\ = 153 \text{ (octal)}$$

**Example 3.** Convert the binary real number 1011.1011 to its equivalent octal number. In the integer part of the binary number the group of 3 bits is formed from right to left. In the binary fraction the group of 3 bits is formed from left to right.

$$(1011.1001)_2 = (1)(011)(101)(1) \\ = (001)(011)(101)(100) \\ = 13.54 \text{ (octal)}$$

## 2.13.8 Conversion of an Octal Number to a Binary Number

To convert an octal number to its equivalent binary number each digit of the number is converted to its 3-bit binary equivalent.

**Example 1.** Convert the octal number 376 to its equivalent binary number.

$$(376)_8 = (011)(111)(110) \\ = (01111110)_2 \\ = (11111110)_2$$

**Example 2.** Convert the real octal number 56.34 to its equivalent binary number.

$$(56.34)_8 = (101)(110)(011)(100) \\ = (101110.011100)_2 \\ = (101110.0111)_2$$

## 2.13.7 Conversion of an Octal Number to a Hexadecimal Number and Vice-Versa

The conversion of a hexadecimal number to its equivalent octal number or vice-versa can easily be done through binary as illustrated by the example given below.

**Example 1.** Convert the hexadecimal number 3DE to its equivalent octal number. The hexadecimal number 3DE is first converted to its equivalent binary number.

$$3DE \text{ (hex)} = (0011)(1101)(1110) \\ = (0011101110110)_2$$

Now the above binary equivalent is divided into groups of 3 bits to obtain its octal equivalent.

$$0011101110110 = (001)(111)(011)(110) \\ = 1736 \text{ (octal)}$$

**Example 2.** Convert the real hexadecimal number 5B.34 to its equivalent octal number. The number 5B.34 is first converted to its binary equivalent.

$$5B.34 = (0101)(1011)(0011)(0100) \\ = 01011011.00110100$$

Now forming the groups of 3 binary bits to obtain its octal equivalent we have,

$$01011011.00110100 = (01)(011)(011)(001)(110)(100) \\ = (001)(011)(011)(001)(110)(100) \\ = 133.184 \text{ (octal)}$$

**Example 3.** Convert the octal number 538 to its equivalent hexadecimal number. Converting 538 (octal) first to its binary equivalent, we get,

$$(538)_8 = (101)(011)(110) \\ = (101011110)_2$$

Now forming the groups of 4 binary bits to obtain its hexadecimal equivalent we get,

$$(101011110) = (1)(0101)(1110) \\ = (0001)(0101)(1110) \\ = 15E \text{ (hex)}$$

**Example 4.** Convert the real octal number 46.57 to its equivalent hexadecimal number. Converting 46.57 (octal) first to its binary equivalent we get,

$$46.57 = (100)(110)(101)(111) \\ = (100110.101111)_2$$

Now forming the groups of 4 binary bits to obtain its hexadecimal equivalent we have,

$$100110.101111 = (10)(0110)(1011)(11) \\ = (0010)(0110)(1011)(1100) \\ = 26.FC \text{ (hex)}$$

## 2.14 ASCII AND ISCI CODES

ASCII is pronounced as "ask-ee". It stands for American Standard Code for Information Interchange. ASCII code is used extensively in small computers, peripherals, instruments and communication devices. It has replaced many of the special codes that were previously used by manufacturers. It is a 7-bit code. Microcomputers using 8-bit word length use 7 bits to present the basic code. The 8th bit is used for parity or it may be permanently 1 or 0. Table 2.1 shows ASCII codes. With 7 bits up to 128 characters can be coded. A letter, digit or special character is called a character. It includes upper and lower case alphabets, numbers, punctuation marks, special characters and control characters. Table 2.7 shows the definitions of control characters. Some control characters which are used in serial communications are: ACK, NAK, etc. ENQ is used for enquiry. ACK is for acknowledgement. It is used to indicate successful reception after completing error checking. NAK is negative acknowledgement, used to indicate errors in reception.

ISCI stands for Indian Standard Code for Information Interchange. It is an 8-bit code for Indian languages.