Fundamentals of Digital Electronics

6

6.1 Introduction

In the modern age of electronics, signal and system play a vital role in everyday life. All of us are familiar with the impact of modern digital computers, display systems, smart phones, internet, 2G, 3G technologies etc. on the society. The operations of these systems and many other systems are based on the principle of digital techniques and these systems are termed as digital systems. The signals processed by these systems are input in the form of digital and are termed as digital signals. Therefore it is important that every design engineer and researcher must have thorough knowledge of digitization process and design.

6.2 Signals and Classification

The concept of signals and systems is applicable to wide range of areas of science and technology like Control Engineering, Circuit and Systems, Bio-medical Engineering, Digital Signals Processing, Digital Image Processing etc. Their physical nature and characteristics from different fields are drastically different. *"Any thing that carries information is called a signal"*. Few examples of signals are ECG signal, EEG signal, speech signal, electrical signal, radio signal, T.V. signal etc. *"A signal is also defined as a physical quantity that is a function of one, two or more than two dependent or independent variables"*. If the signal is dependent on only one variable, then it is termed as one-dimensional signal. For example, ac power supply, speech signal, sound produced by musical instruments are functions of one-variable namely–time. Hence these signals lie under one dimensional signal category. An image signal is a function of two-dimensions (x, y). Hence, it is a two-dimensional signal.

Similarly, the air pressure, wind speed lies under multi-dimensional signal category. Sometimes varying signals occur randomly and do not contain any useful information. Such signals are called noise signals. The response of a system to a signal or processing of a signal depends heavily on the characteristic attribute of the specific signal. Signals can be classified based on their nature and characteristics in the time domain.

They are broadly classified as :

- Continuous time signals
- Discrete time signals

6.2.1 Continuous Time Signals

The continuous time signals are defined for every value of time 't', and is represented by $x(t)$. A continuous time signal is also called an *analog signal*. Most of the signals encountered in practice are continuous time signals. Figure 6.1 shows an example of a continuous time signal whose amplitude varies continuously with time.

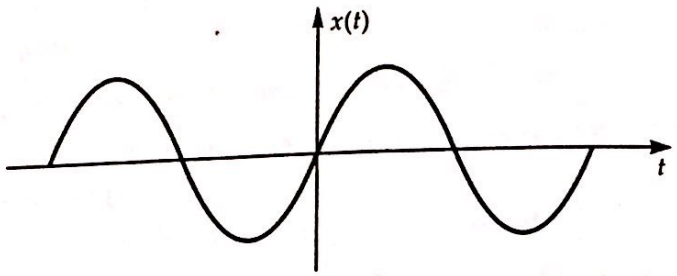


Figure 6.1 A continuous time signal

6.2.2 Discrete Time Signals

A discrete time signal is defined only at discrete instant of time and is represented by $x(n)$, where n is index. Thus in this case the independent variable has discrete values only, which are usually uniformly spaced. A discrete time signal is often derived from a continuous time signal by sampling it at a uniform rate. Sampling a continuous time signal $x(t)$, at time $t = n\tau$, (where n denotes an integer, and τ denotes the sampling period) yield a sample of value $x(n\tau)$. The discrete time signal can be denoted by

$$x(n) = x(n\tau) = x(t)\big|_{t=n\tau} \; ; \quad n = 0, \pm 1, \pm 2, \pm 3, \ldots\ldots \qquad \ldots(6.1)$$

$x(n)$ is sampled version of $x(t)$, and is termed as *discrete time sequence*. Figure 6.2 shows the discrete time signal derived from $x(t)$. Both continuous time and discrete time signals are further classified as follows :

(*i*) Even and odd signals
(*ii*) Deterministic and non-deterministic signals
(*iii*) Periodic and aperiodic signals
(*iv*) Energy and power signals
(*v*) Multichannel and multi-dimensional signals.

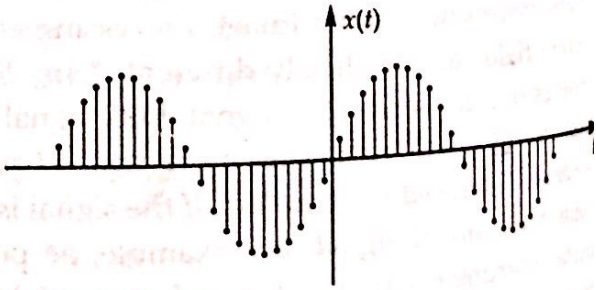


Figure 6.2 Representation of $x(t)$ as a discrete time signal $x(n)$.

6.3 Number Systems

All of us are familiar with the decimal numbering system which we have been using for ages. This numbering system utilizes ten symbols or digits *i.e.*, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, for representing any quantity. This is why this system is named as *"Decimal Number System"*. Since using only ten symbols we can represent any quantity. Hence the base or radix for decimal number system is ten. In general, any number with radix R, having m digits to the left (integers) and n digits to the right (fractions) of radix point can be expressed as

$$\alpha_m(R)^{m-1} + \alpha_{m-1}(R)^{m-2} + \ldots\ldots + \alpha_2(R)^1 + \alpha_1(R)^0 + \beta_1(R)^{-1} + \beta_2(R)^{-2} + \ldots\ldots + \beta_n(R)^{-n} \quad \ldots(6.2)$$

where $(R)^r$ $(r = m-1, m-2, \ldots, 0, -1, -2, \ldots, -n)$ represents the positional weights. α_i and β_i are the symbol values.

The decimal number system is not suitable to use in digital circuits, because for this we need ten different voltage levels, which has to be communicated through ten different wires to the processing circuit. Also it was very difficult to maintain these voltage levels free from noise interference, resulting unwanted change in voltage levels, which will affect the circuit operation. The *binary number system*, which consists of only two digits 0 and 1, is the simplest numbering system for processing by electrical circuits. 1 simply represents 'ON' or conducting state of electrical circuit and 0 simply represents the 'OFF' on non-conducting state of electrical circuit. As a matter of fact the modern computers communicate and operate with binary numbers. In order to deal with large binary strings the other number systems are designed and used widely now a days like, *Octal number systems, hexadecimal number systems.*

The most commonly used number systems, with their radix and symbols are given in Table 6.1.

Table 6.1 *Characteristics of commonly used number systems.*

Number System	*Radix*	*Symbols*
Binary	2	0, 1
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *A, B, C, D, E, F*

6.3.1 Decimal Number System

We are familiar with the decimal number system which we have been using for the ages. This number system utilizes ten symbols (digits) *i.e.*, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. The radix (base) for decimal number system is ten (10) and the radix point is termed as decimal point. The left side digits of the decimal point are termed as *integer parts* and the right side digits of the decimal point are termed as *fraction part*. Any number in decimal number system can be formed by multiplying each decimal number with its positional weight and then adding up all the digit values. Since there are ten digits in the decimal number system, thus the positional weights are in powers of ten.

The leftmost digit has the maximum value hence it is known as the *most significant digit* (MSD) and the rightmost digit has the minimum value, hence it is termed as *least significant digit* (LSD). For example,

$$(5307.105)_{10} = 5\times10^{3} + 3\times10^{2} + 0\times10^{1} + 7\times10^{0} \bullet 1\times10^{-1} + 0\times10^{-2} + 5\times10^{-3}$$

⟵ Integer part ⟶ ↑ ⟵ Fraction part ⟶

Decimal point

6.3.2 Binary Number System

Binary number system consists of only two symbols (digits) *i.e.*, 0 and 1. The radix (base) for binary number system is 2, and the radix point is termed as binary point. In binary system each binary digit is termed as *bit*, having its own positional weight. In any binary sequence the leftmost binary digit is termed as *most significant bit* (MSB) and the rightmost binary digit is termed as *least significant bit* (LSB). The smallest unit of data is defined as a bit. In digital system, the information is stored and processed as a group of bits. A group of four bits is known as *nibble* and a group of eight bits is known as a *byte*. Similarly, a group of two bytes *i.e.*, sixteen bit is termed as a *word*. Since one byte consists of eight bits and each bit may have two values either 1 or 0, hence one byte can represent maximum 2^8 *i.e.*, 256 distinct values.

For example,

$$(1111.010)_2 = 1\times2^{3} + 1\times2^{2} + 1\times2^{1} + 1\times2^{0} \bullet 0\times2^{-1} + 1\times2^{-2} + 0\times2^{-3}$$

⟵ Integer part ⟶ ↑ ⟵ Fractional part ⟶

Binary point

6.3.3 Octal Number System

Octal number system utilizes eight symbols (digits) *i.e.*, 0, 1, 2, 3, 4, 5, 6, 7. The radix (base) for octal number system is 8 and the radix point is termed as octal point. It is also a weighted number system. In any octal sequence each digit has its value defined by its position with base 8. For example,

$$(532.14)_8 = 5\times8^{2} + 3\times8^{1} + 2\times8^{0} \bullet 1\times8^{-1} + 4\times8^{-2}$$

⟵ Integer part ⟶ ↑ ← Fractional part →

Binary point

6.3.4 Hexadecimal Number System

The base (radix) of hexadecimal number system is 16. The hexadecimal number system utilizes sixteen digits to represent any number. These symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *A, B, C, D, E, F*. In the hexadecimal number system, the radix point is termed as *hexadecimal point*. In any hexadecimal sequence, each digit has its value defined by its position with weight 16.

For example,

$$(B26.DE)_{16} = B\times16^{2} + 2\times16^{1} + 6\times16^{0} \bullet D\times16^{-1} + F\times16^{-2}$$

⟵ Integer part ⟶ ↑ ← Fractional part →

Hexadecimal point

The relationship between the various number systems as discussed above are listed in Table 6.2.

Table 6.2 *Relationship between various number systems.*

S. No.	Decimal Number System	Binary Number System	Octal Number System	Hexadecimal Number System
1	0	0000	0	0
2	1	0001	1	1
3	2	0010	2	2
4	3	0011	3	3
5	4	0100	4	4
6	5	0101	5	5
7	6	0110	6	6
8	7	0111	7	7
9	8	1000	10	8
10	9	1001	11	9
11	10	1010	12	A
12	11	1011	13	B
13	12	1100	14	C
14	13	1101	15	D
15	14	1110	16	E
16	15	1111	17	F

6.4 Conversion of Numbers

The human being uses decimal number system, for normal mathematical operations, while the digital computer uses binary data for processing. In order to deal with large quantity of binary numbers of many bits, octal number system and hexadecimal number system are extensively used. Hence, it is essential to understand the conversion process of numbers from one number system to another.

6.4.1 Decimal to Binary Conversion

Any decimal number can be converted into its equivalent binary number using following steps :

Step 1 : *Separation*

Separate the given decimal number into its integer and fractional parts. The decimal number lying left hand side of decimal point is integer decimal number and the decimal number lying right hand side of the decimal point is fractional decimal number.

For example,

$(76.625)_{10}$ = 76 • 625

Integer decimal number — Decimal point — Fractional decimal number

Step 2 : *Integer Conversion*

In order to convert the integer decimal number into its equivalent binary number, the *repeated division method* is used. In this method, divide the decimal number by 2, producing a quotient and a remainder. This remainder is *least significant bit* (LSB) of the desired binary pattern.

Divide the quotient obtained by the previous division by two, which produces another quotient and remainder. This remainder is next digit of the binary number. This division process continues until the quotient becomes zero *i.e.*, 0. The remainder obtained in the final division is termed as *most significant bit* (MSB).

For example,

$$(76)_{10} = (?)_2$$

Integer Conversion

	Division		*Remainder*	
2	76			
2	38	→	0	(LSB)
2	19	→	0	
2	9	→	1	
2	4	→	1	
2	2	→	0	
2	1	→	0	
	0	→	1	(MSB)

$\Rightarrow \quad (76)_{10} = (1001100)_2$

Step 3 : *Fractional Conversion*

In order to convert the fractional part of the decimal number to its equivalent binary number, the *repeated multiplication method* is used. In this method the fractional decimal number is multiplied by 2, producing decimal number, consisting both integer part and fractional part. This integer part is termed as MSB of the desired binary number. Now again repeat the same process discussed above with the fractional portion obtained in the previous stage, producing next binary bit. This process is repeated until the fractional part of a decimal number becomes zero. The integer part produced as the result of last multiplication is LSB of the desired binary number.

For example,

$$(0.625)_{10} = (\ ?\ )_2$$

Fractional Conversion

Multiplication		*Integer*	*Fraction*
$0.625 \times 2 = 1.25$	$\longrightarrow$	1 (MSB)	0.25
$0.25 \times 2 = 0.50$	$\longrightarrow$	0	0.50
$0.50 \times 2 = 1.00$	$\longrightarrow$	1 (LSB)	0.00

$$(0.625)_{10} = (.101)_2$$

$\Rightarrow$

Step 4 : Combination

Combine the binary number obtained in step 2 and step 3, separated by binary point *i.e.,*

$$(76 \cdot 625)_{10} = (1001100 \cdot 101)_2$$

Decimal point Binary point

Example 6.1 *Convert the decimal number* $(3289.625)_{10}$ *into an equivalent binary number.*

Solution. *Step (i).* : ***Separation.*** Separate the given decimal number into integer part and fractional part

Decimal number = Integer part + Fractional part

$$(3289.625)_{10} = (3289)_{10} + (0.625)_{10}$$

Step (ii) : ***Integer conversion***

	Division		*Remainder*	
2	3289			
2	1644	$\longrightarrow$	1	(LSB)
2	822	$\longrightarrow$	0	
2	411	$\longrightarrow$	0	
2	205	$\longrightarrow$	1	
2	102	$\longrightarrow$	1	
2	51	$\longrightarrow$	0	
2	25	$\longrightarrow$	1	
2	12	$\longrightarrow$	1	
2	6	$\longrightarrow$	0	
2	3	$\longrightarrow$	0	
2	1	$\longrightarrow$	0	
	0	$\longrightarrow$	1	(MSB)

Now taking the remainders from MSB to LSB (*i.e.,* from bottom to top) gives the binary equivalent. Thus

$$(3289)_{10} = (1000\ 1101\ 1001)_2$$

Step (iii) : ***Fractional conversion***

Multiplication	*Integer*		*Fraction*
$0.625 \times 2 = 1.250$	1	(MSB)	0.250
$0.250 \times 2 = 0.500$	0		0.500
$0.500 \times 2 = 1.000$	1		0.000
$0.000 \times 2 = 0.000$	0	(LSB)	0.000

After multiplication the fractional part by 2, and considering the integer part generated each time, the fractional part is forwarded for the same process repeatedly. Now taking the integers generated from MSB to LSB (*i.e.*, from top to bottom) gives the binary equivalent *i.e.*,

$$(0.625)_{10} = (1010)_2$$

Step (iv) ***Combination***

$$(3289.625)_{10} = (1000\ 1101\ 1001.1010)_2$$

Example 6.2 *Convert the following decimal numbers into their binary equivalents :*

(a) $(25.53125)_{10}$ *(b)* $(37.50)_{10}$ *(c)* $(10.6875)_{10}$

Solution. *(a)* $(25.53125)_{10} = (?)_2$

Step (i) : ***Separation***

Integer part + Fractional part

$$(25.53125)_{10} = (25)_{10} + (0.53125)_{10}$$

Step (ii) : ***Integer conversion***

	Division		*Remainder*	
2	25			
2	12	→	1	(LSB)
2	6	→	0	
2	3	→	0	
2	1	→	1	
2	0	→	1	(MSB)

$$\Rightarrow \quad (25)_{10} = (11001)_2$$

Step (iii) : ***Fractional conversion***

Multiplication	*Integer*		*Fraction*
$0.53125 \times 2 = 1.6250$	1	(MSB)	0.625
$0.625 \times 2 = 1.250$	1		0.250
$0.250 \times 2 = 0.500$	0		0.500
$0.500 \times 2 = 1.000$	1	(LSB)	0.000

$$\Rightarrow \quad (0.53125)_{10} = (0.1101)_2$$

Step (iv) : ***Combination***

$$(25.53125)_{10} = (11001.1101)_2$$

(b) $(37.50)_{10} = (?)_2$

Step (i) : ***Separation***

Decimal number = Integer part + Fractional part

$$(37.50)_{10} = 37 + 0.50$$

Step (ii) : ***Integer conversion***

	Division		*Remainder*	
2	37			
2	18	→	1	(LSB)
2	9	→	0	
2	4	→	1	
2	2	→	0	
2	1	→	0	
	0	→	1	(MSB)

$\Rightarrow \quad (37)_{10} = (100101)_2$

Step (iii) : ***Fractional conversion***

Multiplication		*Integer part*	*Fraction part*
$0.50 \times 2 = 1.00$	→	1 (MSB)	00

$\Rightarrow \quad (0.50)_{10} = (0.1)_2$

Step (iv) : ***Combination***

$$(37.50)_{10} = (100101.1)_2$$

(c) $(10.6875)_{10} = (?)_2$

Step (i) : ***Separation***

Decimal number = Integer part + Fractional part

$$(10.6875)_{10} = (10)_{10} + (0.6875)_{10}$$

Step (ii) : ***Integer conversion***

	Division		*Remainder*	
2	10			
2	5	→	0	(LSB)
2	2	→	1	
2	1	→	0	
	0	→	1	(MSB)

$\Rightarrow \quad (10)_{10} = (1010)_2$

Step (iii) : ***Fractional conversion***

Multiplication	*Integer part*	*Fraction part*
$0.6875 \times 2 = 1.3750$	1 (MSB)	0.3750
$0.3750 \times 2 = 1.7500$	1	0.7500
$0.75 \times 2 = 1.50$	1	0.50
$0.50 \times 2 = 1.00$	1 (LSB)	0.00

$$\Rightarrow \quad (0.6875)_{10} = (0.1111)_2$$

Step (iv) : ***Combination***

$$(10.6875)_{10} = (1010.1111)_2$$

6.4.2 Binary to Decimal Conversion

We know that the base of the binary number is 2. To convert the binary number into its equivalent decimal number following steps can be adopted :

Step 1 *Separation* : Separate the given binary number into its integer and fractional part.

Step 2 *Conversion* : Multiply each binary digit by its positional weight.

Step 3 *Combination* : Add up the product to provide the equivalent decimal number.

Example 6.3 *Convert the binary number* $(1011.1010)_2$ *into its decimal equivalent.*

Solution. Step (i) : ***Separation***

$$\begin{array}{ccccc} \text{Binary number} & = & \text{Integer part} & + & \text{Fractional part} \\ \downarrow & & \downarrow & & \downarrow \\ (1011.1010)_2 & = & (1011)_2 & + & (0.1010)_2 \end{array}$$

Step (ii) : ***Integer conversion***

Binary number	1	0	1	1
Digit position	3	2	1	0
Positional weight	2^3	2^2	2^1	2^0
Positional value	8	4	2	1
Integer value				
	$8 = 1 \times 8$			
	$0 = 0 \times 4$			
	$2 = 1 \times 2$			
	$1 = 1 \times 1$			
Total	11			

$$\Rightarrow \quad (1011)_2 = (11)_{10}$$

Step (iii) : ***Fraction conversion***

Binary point					
Binary number	.	1	0	1	0
Digit position		−1	−2	−3	−4
Positional weight		2^{-1}	2^{-2}	2^{-3}	2^{-4}
Positional value		0.5	0.25	0.125	0.0625
Integer value					
		0.500 ← 1 × 0.5			
		0.000 ← 1 × 0.25			
		0.125 ← 1 × 0.125			
		0.000 ← 1 × 0.0625			
		Total 0.625			

$\Rightarrow \quad (\cdot 1010)_2 = (0.625)_{10}$

Step (iv) : ***Combination***

Therefore, $(1011.1010)_2$ is equal to

$$(1011.1010)_2 = (11.625)_{10}$$

Example 6.4 *Convert the following binary numbers into decimal number :*

(*a*) 010011 (*b*) 100110.1011 (*c*) 1111.1111 (*d*) 1101101.001

Solution. (*a*) Given $(010011)_2$

$$\Rightarrow \quad 0\times2^5+1\times2^4+0\times2^3+0\times2^2+1\times2^1+1\times2^0$$

$$\Rightarrow \quad 0+16+0+0+2+0=(18)_{10}$$

(*b*) $(100110.1011)_2$

$$\Rightarrow \quad 1\times2^5+0\times2^4+0\times2^3+1\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}+1\times2^{-4}$$

$$\Rightarrow \quad 32+0+0+4+2+0+0.500+0+0.125+0.0625=(38.6875)_{10}$$

(*c*) $(1111\cdot1111)_2$

$$\Rightarrow \quad 1\times2^3+1\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}+1\times2^{-3}+1\times2^{-4}$$

$$\Rightarrow \quad 8+4+2+1+0.500+0.250+0.125+0.625=(15.9375)_{10}$$

(*d*) $(110110.001)_2$

$$\Rightarrow \quad 1\times2^5+1\times2^4+0\times2^3+1\times2^2+1\times2^1+0\times2^0+0\times2^{-1}+0\times2^{-2}+1\times2^{-3}$$

$$\Rightarrow \quad 32+16+0+4+2+0+0+0+0.125=(54.125)_{10}$$

6.4.3 Octal Conversion

As we already discussed in the previous section, the base of octal number system is 8. The complete representation of octal number with digit position, positional weights and value is given in Fig. 6.3.

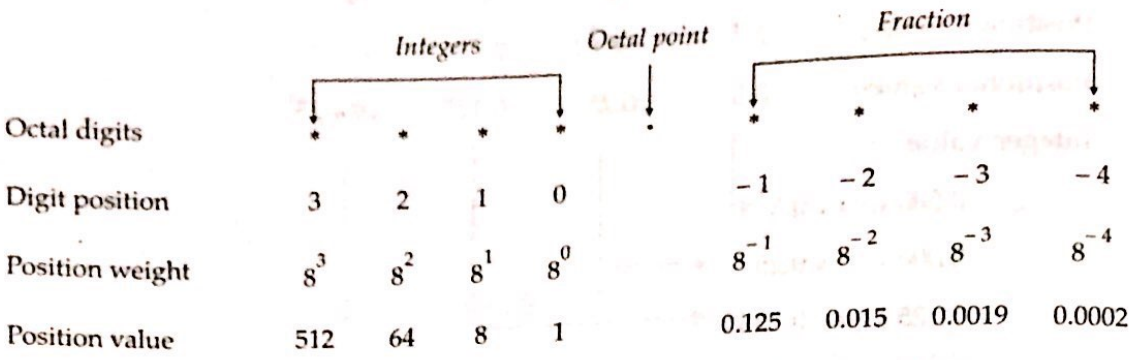


Figure 6.3 Positional weights and values of octal numbers

Hence it is clear that octal number system is also a positional number system, where each digit carries weights depending on its position relative to the octal point. In order to understand the representation of octal numbers in other number system and vice-versa we need to understand the conversion process discussed as under.

6.4.3A Decimal to Octal Conversion

For conversion of decimal number into its octal equivalent we follow the same procedure as we discussed for decimal to binary conversion. Since the base of the octal number system is 8,

Step (iii) : Fractional conversion

Multiplication	Integer		Fraction
$0.35 \times 8 = 2.8$	2	(MSB)	0.8
$0.8 \times 8 = 6.4$	6		0.4
$0.4 \times 8 = 3.2$	3		0.2
$0.2 \times 8 = 1.6$	1	(LSB)	0.6

$$(0.35)_{10} = (.2631)_8$$

$\Rightarrow$

Step (iv) Combination

$$(127.35)_{10} = (177.2631)_8$$

6.4.3B Octal to Decimal Conversion

To convert the octal number into its equivalent decimal number following steps can be adopted :

Step 1 *Separation* : Separate the given octal number into its integer and fractional part.

Step 2 *Conversion* : Multiply each digit of octal number by its positional weight.

Step 3 *Combination* : Add the product and combine the result obtained in integer and fractional conversion to provide the equivalent decimal number.

Example 6.6 *Convert the octal number* $(10110.101)_8$ *into its decimal equivalent.*

Solution. Given $(10110.101)_8$

Step (i) : Separation

Octal number = Integer part + Fractional part

$$(10110.101)_8 = (10110)_8 + (0.101)_8$$

Step (ii) : Conversion

(a) Integer conversion

$$(10110)_8 = 1\times8^4 + 0\times8^3 + 1\times8^2 + 1\times8^1 + 0\times8^0$$
$$= 4096 + 0 + 64 + 8 + 0 = (4104)_{10}$$

(b) Fractional conversion

$$(0.101)_8 = 1\times8^{-1} + 0\times8^{-2} + 1\times8^{-3}$$
$$= 0.125 + 0 + 0.0019 = (0.1269)_{10}$$

Step (iii) : Combination

$\Rightarrow$

$$(10110.101)_8 = (4104.1269)_{10}$$

6.4.3C Octal to Binary Conversion

In order to convert the octal number into its equivalent binary number, for each octal digit write the corresponding binary digit. The same procedure is adopted for both integer and fractional parts.

Example 6.7 *Convert the octal number* $(725.31)_8$ *into binary form.*

Solution. Given $(725.31)_8$

$$\Rightarrow \quad \begin{array}{cccccc} 7 & 2 & 5 & . & 0 & 0 \\ \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow \\ 111 & 010 & 101 & . & 011 & 001 \end{array}$$

$$\Rightarrow \quad (111010101.011001)_2.$$

Example 6.8 *Convert the octal number* $(425)_8$ *into binary form.*

Solution. Given $(425)_8$

$$\Rightarrow \quad \begin{array}{ccc} 4 & 2 & 5 \\ \downarrow & \downarrow & \downarrow \\ 100 & 010 & 101 \end{array}$$

$$\Rightarrow \quad (100010101)_2$$

6.4.3D Binary to Octal Conversion

To convert the binary number into equivalent octal number following steps are used :

Step 1 Starting from the binary point separate the binary number into groups of three digits for both towards the integer part as well as fractional part.

Step 2 Write the equivalent octal number corresponding to each group.

NOTE *Add binary digit 0 at each end, if necessary to complete a group of three digits.*

Example 6.9 *Convert* $(110110101.110)_2$ *to octal equivalent.*

Solution. Given $(110110101.110)_2$

$$\Rightarrow \quad \begin{array}{ccccc} 110 & 110 & 101 & . & 110 \end{array}$$

$$\Rightarrow \quad \begin{array}{ccccc} 6 & 6 & 5 & . & 6 \end{array}$$

Therefore, $(110110101.110)_2 = (665.6)_8$.

6.4.4 Hexadecimal Conversion

We already discussed in previous section, the base of hexadecimal number system is 16. The complete representation of hexadecimal number with digit positions, positional weights and values is given in Fig. 6.4.

Hexadecimal digits								
Digit position	3	2	1	0	−1	−2	−3	−4
Position weight	16^3	16^2	16^1	16^0	16^{-1}	16^{-2}	16^{-3}	16^{-4}
Position value	4096	256	16	1	0.062	0.039	0.0002	0.00024

Figure 6.4 Positional weights and values of hexadecimal numbers

Hence it is clear that hexadecimal number system is also a positional number system, where each digit carries weights depending on its position relative to the hexadecimal point. In order to understand the representation of hexadecimal numbers in other number system and vice-versa we need to understand the conversion process discussed as under.

6.4.4A Hexadecimal to Binary Conversion

A hexadecimal number is also termed as 4-bit binary number, since each digit of a hexadecimal number can be represented by a 4-bit binary number.

Hence for conversion of any hexadecimal number to its equivalent binary number following steps may be adopted :

Step 1 Convert each hexadecimal number to its equivalent 4-bit binary number.

Step 2 The groups of bits are combined to form the binary number.

Example 6.10 *Convert the following hexadecimal numbers to binary equivalents :*

(*a*) AB3C.FF (*b*) 1FE8.DC

Solution. (*a*)

A	B	3	C	.	F	F
↓	↓	↓	↓		↓	↓
1010	1011	0011	1100	.	1111	1111

$\Rightarrow \quad (AB3C.FF)_{16} = (1010\ 1011\ 0011\ 1100\ .\ 1111\ 1111)_2$

(*b*)

1	F	E	8	.	D	C
↓	↓	↓	↓		↓	↓
0001	1111	1110	1000	.	1101	1100

$\Rightarrow (1FE8.DC)_{16} = (0001\ 1111\ 1110\ 1000\ .\ 1101\ 1100)_2$

$= (1111\ 1111\ 01000\ .\ 110\ 111)_2$

6.4.4B Binary to Hexadecimal Number

To convert the binary number into equivalent hexadecimal number following steps are used :

Step 1 Starting from the hexadecimal point separate the hexadecimal number into groups of four bits (4-bits) for both toward the integer part as well as for fraction part.

Step 2 Write the equivalent hexadecimal number corresponding to each group.

NOTE *Add binary digit 0 to each end if necessary to complete a group of 4-bits.*

Example 6.11 *Convert the following binary numbers to their hexadecimal equivalents :*

(*a*) $(110010)_2$ (*b*) $(110\ 110)_2$ (*c*) $(1011\ \ 1101\ \ 1011)_2$

(*d*) $(111\ \ 101\ \ 010\ .\ \ 10101)_2$ (*e*) $(1100\ \ 10101\ .\ \ 1010\ \ 111)_2$

Solution. (*a*) Given binary number

$$(110010)_2$$

$\Rightarrow$ 11 0010 = 0011 0010 $\Rightarrow$ $(32)_{16}$

(*b*) Given binary number

$$(110110)_2$$

$\Rightarrow$ 11 0110 = 0011 0110 $\Rightarrow$ $(36)_{16}$

(*c*) Given binary number

$$(1011\ \ 1101\ \ 1011)_2$$

$\Rightarrow$ 1011 1101 1011 $\Rightarrow$ $(B\ D\ B)_{16}$

(*d*) Given binary number

$$(111101010\ .\ 10101)_2$$

$\Rightarrow$ 1 1110 1010 . 1010 1

$\Rightarrow$ 0001 1110 1010 . 1010 1000

$\Rightarrow$ $(1\ E\ A\ .\ A\ 8)_{16}$

(*e*) Given binary number

$$(110010101.1010111)_2$$

$\Rightarrow$ 1 1001 0101 . 1010 111

$\Rightarrow$ 0001 1001 0101 . 1010 1110

$\Rightarrow$ $(1\ 95\ .\ A\ E)_{16}$

6.4.4C Decimal To Hexadecimal Conversion

There are different methods to convert the integer and fraction part of decimal number into its hexadecimal equivalent.

Method 1 *The simplest method of conversion for any decimal number to its equivalent hexadecimal equivalent includes following steps :*

Step 1 Convert decimal number to its equivalent binary number.

Step 2 Convert the binary number formed to its equivalent Hexadecimal number.

Example 6.12 *Convert* $(26.53125)_{10}$ *to its equivalent hexadecimal number.*

Solution. Given the decimal number

Decimal number = Integer part + Fractional part

$$(26.53125)_{10} = (26)_{10} + (0.53125)_{10}$$

Decimal to binary conversion

$\Rightarrow$ $(26)_{10}$

	Division		*Remainder*	
2	26	$\longrightarrow$	0	(LSB)
2	13	$\longrightarrow$	1	
2	6	$\longrightarrow$	0	
2	3	$\longrightarrow$	1	
2	1	$\longrightarrow$	1	(MSB)
	0			

$\Rightarrow$ $(26)_{10} = (11010)_2$

Similarly, $(0.53125)_{10}$

Multiplication	*Integer*	*Fraction*
$0.53125 \times 2 = 1.06250$	1 (MSB)	0.6250
$0.06250 \times 2 = 1.2500$	0	0.1250
$0.1250 \times 2 = 0.2500$	0	0.2500
$0.2500 \times 2 = 0.5000$	0	0.5000
$0.5000 \times 2 = 1.0000$	1 (LSB)	0.0000

$\Rightarrow$ $(0.53125)_{10} = (.10001)_2$

Hence $(26.53125)_{10} = (11010.10001)_2$

Binary to Hexadecimal Conversion

Since from previous section

$$(26.53125)_{10} = (11010.10001)_2$$

$\Rightarrow$ 1 1010 . 1000 1

$\Rightarrow$ 0001 1010 . 1000 1000

↓ ↓ ↓ ↓

$\Rightarrow$ 1 A . 8 8

Hence, $(26.53125)_{10} = (1A.88)_{16}$

Method 2 *Alternatively we can follow the same procedure as we have discussed for decimal to binary conversion, for the conversion of decimal number into its equivalent hexadecimal number. Since the base of the hexadecimal number system is 16, hence for conversion of the integer part of the decimal number we use the method of successive division by 16. Similarly, for conversion of fraction part of the decimal number, repeated multiplication method is used. Here the decimal number is to be multiplied by 16.*

Example 6.13 *Convert decimal number* $(26.53125)_{10}$ *into its equivalent hexadecimal number.*

Solution. Step (i). Given the decimal number

Decimal number = Integer part + Fractional part

↓ ↓ ↓

$(26.53125)_{10} = (26)_{10} + (0.53125)_{10}$

Step (ii). ***Integer conversion***

	Division		*Remainder*	
16	26			
16	1	→	*A*	(LSB) ↑
	0	→	1	(MSB) ↑

$\Rightarrow (26)_{10} = (1A)_{16}$

Step (iii) : ***Fraction conversion***

Multiplication	*Integer Part*	*Fraction Part*
$0.53125 \times 16 = 8.5$	8 (MSB)	0.5
$0.50 \times 16 = 8.0$	8 (LSB) ↓	0.0

$\Rightarrow (0.53125)_{10} = (0.88)_{16}$

Step (iv) : ***Combination***

$$(26.53125)_{10} = (1A.88)_{16}$$

6.4.40 To convert the ... steps can be adopted.

Step 1 *Separation* : Separate the given hexadecimal number into its integer and fractional part.

Step 2 *Conversion* : Multiply each digit of hexadecimal number by its positional weight.

Step 3 *Combination* : Add the product and combine the result obtained in integer and fractional conversion to provide the equivalent decimal number.

Example 6.14 *Convert* $(1101101.110110)_{16}$ *to its equivalent decimal number.*

Solution. Given $(1101101.110110)_{16}$

Step (i) : ***Separation***

Hexadecimal number = Integer Part + Fractional Part

$(1101101.110110)_{10} = (1101101)_{16} + (0.110110)_{10}$

Step (ii) : ***Conversion***

(a) Integer conversion

$$(1101101)_{16} = 16^6 + 1\times16^5 + 0\times16^4 + 1\times16^3 + 1\times16^2 + 0\times16^1 + 1\times16^0$$
$$= 16777216 + 1048576 + 0 + 4096 + 256 + 0 + 6$$
$$= (17830145)_{10}$$

(b) Fractional conversion :

$$(0.110110)_{16} = 1\times16^{-1} + 1\times16^{-2} + 0\times16^{-3} + 1\times16^{-4} + 1\times16^{-5} + 0\times16^{-6}$$
$$= 0.0625 + 0.0039 + 0 + .0000 + 0.0000 + 0$$
$$= (0.1015)_{10}$$

Step (iii) : ***Combination***

$$(1101101.110110)_{16} = (17830145.1015)_{15}$$

6.4.4E Octal to Hexadecimal Conversion

To convert octal number to its hexadecimal equivalent simplest method is : convert the octal number to binary number and then convert the binary number to its hexadecimal equivalent.

Example 6.15 *Convert the following octal numbers to its hexadecimal equivalents :*

(*a*) $(74.60)_8$ (*b*) $(76.524)_8$ (*c*) $(37.56)_8$ (*d*) $(177.546)_8$

Solution. (*a*) Given octal number $(74.60)_8$

Step (i) : ***Octal to binary conversion***

$$(74.60)_8 = (111\ 100.110\ 000)_2$$

Step (ii) : ***Binary to hexadecimal conversion***

$$(111100.110000)_2 = 111100.110000$$
$$= 0011\ \ 1100\ .\ 1100\ \ 0000$$
$$= (3C\ .\ C0)_{16}$$

(*b*) Given octal number $(76.524)_8$

Step (i) : ***Octal to binary conversion***

$$(76.524)_8 = (111\ 110.101\ 010\ 100)_2$$

Step (ii) : ***Binary to hexadecimal conversion***

$$(111\ 110\ .\ 101\ 010\ 100)_2 = 11\ \ 1110\ .\ 1010\ \ 1010\ \ 0$$
$$= 0011\ \ 1110\ .\ 1010\ \ 1010\ \ 0000$$
$$= (3E.AA0)_{16}$$

(*c*) Given Octal number $(37.56)_8$

Step (i) : ***Octal to binary conversion***

$$(37.56)_8 = (011\ \ 111\ .\ 101\ \ 110)_2$$

Step (ii) : ***Binary to hexadecimal conversion***

$$(011\ 111.101\ 110)_2 = 01\ \ 1111\ .\ 1011\ \ 10$$
$$= 0001\ \ 1111\ .\ 1011\ \ 1000$$
$$= (1F.B8)_{16}$$

(*d*) Given octal number $(177.546)_8$

Step (i) ***Octal to binary conversion***

$$(177.546)_8 = (001\ \ 111\ \ 111\ .\ 101\ \ 100\ \ 110)_2$$

Step (ii) ***Binary to hexadecimal conversion***

$$(001\ 111\ 111.101\ 100\ 110)_2 = 0\ \ 0111\ \ 1111\ .\ 1011\ \ 0011\ \ 0$$
$$= 0111\ \ 1111\ .\ 1011\ \ 0011$$
$$= (7F\ .\ B3)_{16}$$

6.4.4F Hexadecimal to Octal Conversion

To convert hexadecimal number to its octal equivalent simplest method is, first convert the hexadecimal number to its binary equivalent and then convert the binary number to its octal equivalent.

Example 6.16 *Convert the following hexadecimal numbers to its octal equivalents :*

(*a*) $(A23.E4)_{16}$ (*b*) $(F32.45)_{16}$ (*c*) $(AC.45B)_{16}$ (*d*) $(356.21)_{16}$

Solution. (*a*) Given hexadecimal number $(A23.E4)_{16}$

Step (i) : ***Hexadecimal to binary conversion***

$$(A23.E4)_{16} = (1010\ \ 0010\ \ 0011\ .\ 1110\ \ 0100)_2$$

Step (ii) : ***Binary to octal conversion***

$(1010\ 0010\ 0011.1110\ 0100)_2 = 101\ \ 000\ \ 100\ \ 011\ .\ 111\ \ 001\ \ 00$

$\Rightarrow \quad (5043.710)_8$

(*b*) Given hexadecimal number $(F32.45)_{16}$

Step (i) : ***Hexadecimal to binary conversion***

$$(F32.45)_{16} = (1111\ \ 0011\ \ 0010\ .\ 0100\ \ 0101)_2$$

Step (ii) : ***Binary to octal conversion***

$(1111\ 0011\ 0010.0100\ 0101)_2 = 111\ \ 100\ \ 110\ \ 010\ .\ 010\ \ 001\ \ 010$

$$= (7462.212)_8$$

(*c*) Given hexadecimal number $(AC.45B)_{16}$

Step (i) : ***Hexadecimal to binary conversion***

$$(AC.45B)_{16} = (1010\ \ 1100\ .\ 0100\ \ 0101\ \ 1011)_2$$

Step (ii) : ***Binary to octal conversion***

$(1010\ 1100\ .\ 0100\ 0101\ 1011)_2 = 010\ 101\ 100\ .\ 010\ 001\ 011\ 011$

$$= (254.2133)_8$$

(*d*) Given hexadecimal number $(356.21)_{16}$

Step (i) : ***Hexadecimal to binary conversion***

$$(356.21)_{16} = (0011\ 0101\ 0110\ .\ 0010\ 0001)_2$$

Step (ii) : ***Binary to octal conversion***

$(0011\ 0101\ 0110\ .\ 0010\ 001)_2 = 001\ 101\ 010\ 110\ .\ 001\ 000\ 100$

$$= (1\ 5\ 2\ 6\ .\ 1\ 0\ 4)_8$$

6.4.5 Conversion of Arbitrary Radix to Decimal Number

In general any number with radix R, having m digits to the left (*i.e.*, integers) and n digits to the right (*i.e.*, fractions), of radix point can be represented in decimal form as

$$N = \alpha_m (R)^{m-1} + \alpha_{m-1}(R)^{m-2} + \ldots + \alpha_2 (R)^1 + \alpha_1 (R)^0 +$$

$$\beta_1 (R)^{-1} + \beta_2 (R)^{-2} + \ldots + \beta_{n-1}(R)^{-(n-1)} + \beta_n (R)^{-n}$$

where $(R)^r .(r = m-1, m-2, \ldots\ldots, 0, -1, -2, \ldots\ldots, -(n-1), -n)$ represents the positional weight. α_i and β_i are the digit values and N is the equivalent decimal number.

6.4.6 Conversion of Decimal Number to Arbitrary Radix

In order to convert the decimal number to any arbitrary radix follow the same method discussed previously. For integer part successive division method is used and for fractional part method of successive multiplication is used.

6.5 Signed and Unsigned Binary Numbers

6.5.1 Unsigned Binary Number

In all the number systems both positive and negative numbers are possible. Similar to other number systems, unsigned binary numbers are those numbers which can represent only positive numbers. For example, the smallest 16-bit binary number is 0000 0000 0000 0000 and the largest 16-bit binary number is 1111 1111 1111 1111 *i.e.*, the total range of 16-bit unsigned binary number is from $(0000)_H$ to $(FFFF)_H$. This represents the magnitude of 16-bit binary number is in the range of 0 to 65,536. In general, the n-bit unsigned binary number represents the maximum positive number $(2^n - 1)$. For example,

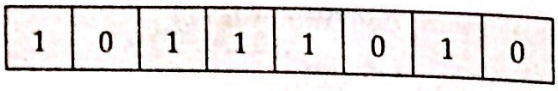


Figure 6.5 Unsigned 8-bit binary number.

6.5.2 Signed Binary Numbers

The unsigned binary numbers are always considered as positive numbers. How to represent negative numbers in binary number system ? As in the decimal number system, plus (+) and minus (–) signs are used to indicate the positive and negative numbers. In binary number system, MSB represents whether the binary number is positive or negative.

When the MSB is 1, the binary number is negative and when it is 0, the number is positive. The signed binary number system is represented by following *three* methods :

- Sign magnitude representation
- 1's complement representation
- 2's complement representation

6.5.2A Sign-Magnitude Representation

In sign magnitude representation of a binary number, most significant bit (MSB) is used to represent the sign of the that number as shown in Fig. 6.6. It is also termed as sign bit. Zero(0) is used to represent a positive number and 1 is for a negative number. The remaining bits are used to represent the magnitude of number. In an n-bit signed binary number the first bit *i.e.*, MSB represents the sign of the number and the remaining $(n-1)$ bits express the magnitude of the number in binary. Hence the n-bit binary sign-magnitude binary number system represents the maximum positive number $(2^{n-1} - 1)$ and the maximum negative number $-(2^{n-1} - 1)$. Hence the range of n-bit sign-magnitude binary number is $-(2^{n-1} - 1)$ to $(2^{n-1} - 1)$.

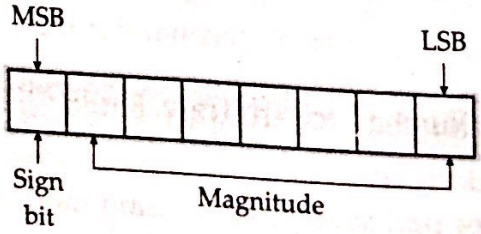


Figure 6.6 Sign-magnitude representation of a binary number.

6.5.2B 1's Complement Representation

The 1's complement of a binary number is found by simply changing all 1's to 0's and all 0's to 1's. In general, the n-bit binary number in 1's complement form represents the maximum positive number $(2^{n-1}-1)$ and the maximum negative number $-(2^{n-1}-1)$. If we take 1's complement of positive/negative number the resultant number will have same magnitude, but opposite sign *i.e.*, negative/positive *i.e.*, when a number is negative, the magnitude is written by taking 1's complement of the magnitude of the number.

Example 6.17 *Find the 1's complement of* $(+18)_{10}$.

Solution. Given decimal number $(+18)_{10}$

Binary representation of given decimal number

$$(18)_{10} = (0001\ 0010)_2$$

1's complement of $(18)_{10} = (1110\ 1101)_2$

6.5.2C 2's Complement Representation

The 2's complement of a binary number is found by simply adding 1 to its 1's complement number. In general the n-bit binary number in 2's complement form represents the maximum positive number $(2^{n-1}-1)$ and the maximum negative number $-(2^{n-1}-1)$. If we take 2's complement of any number the resultant number will have same magnitude but opposite sign. When the number is negative, the magnitude is written by taking 2's complement of that number and MSB is set to 1, representing the number is negative.

For example,

Number	2's complement representation
$(+7)_{10}$	0111
$(-7)_{10}$	1001

Example 6.18 *Find the 2's complement of* $(18)_{10}$.

Solution. Given decimal number $(18)_{10}$.

Binary representation of given decimal number $(18)_{10} = (0001\ 0010)_2$

1's complement of $(18)_{10} = (1110\ 1101)_2$

2's complement of $(18)_{10} = \quad +1$

$$= \overline{1110\ 1110}$$

$$\Rightarrow \quad (1110\ 1110)_2$$

6.6 Binary Codes

As we know that digital machines recognize only '0' and '1'. Hence any data as an input to the digital machine, which may contain numbers, alphabets, punctuations and special characters need to be coded to get processed by digital computers/machines. The coding must be done in such a manner that each number, alphabet, punctuation and special character etc. are represented by a unique combination of 0's and 1's. There are large number of coding schemes discussed in the following sections :

Classification of Binary Codes

Depending upon coding schemes binary codes are classified as :

- Weighted binary codes
- Non-weighted binary codes
- Cyclic or reflective codes
- Sequential codes
- Alphanumeric codes
- Error detecting and correcting codes

6.6.1 Weighted Binary Codes

The weighted binary codes obey the positional weighting principle. Each digit position represents a specific weight. In weighted binary coding scheme each digit is multiplied by the respective positional weight and the sum of these weighted bits produces the equivalent decimal number. The codes 8421, 2421, 5421, 3321, etc. all are weighted binary codes.

The few 4-bit weighted binary codes, with their decimal equivalents are presented in Table 6.3.

Table 6.3 *4-bit weighted binary codes.*

Decimal Number	*5421*	*2421*	*8421*
0	0000	0000	0000
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	0100	0100
5	1000	1011	0101
6	1001	1100	0110
7	1010	1101	0111
8	1011	1110	1000
9	1100	1111	1001

6.6.1A Straight Binary Codes

It is a method of representing a decimal number using natural (straight) binary conversion method to its equivalent binary form. It is a weighted code since a weight is assigned to every bit position.

6.6.1B BCD Code (8421-Code)

Binary coded decimal (BCD) uses the binary number system to specify the decimal numbers 0 to 9. It is a 4-bit weighted binary coding scheme. In this coding scheme each decimal digit of a decimal number is represented by its 4-bit equivalent BCD-code.

Table 6.4 shows the comparison of straight binary code and BCD code.

Table 6.4 *4-bit straight binary code and equivalent BCD code representation.*

Decimal Number	Straight Binary Code				BCD			
	B_3	B_2	B_1	B_0	8	4	2	1
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	0	1	0	1	0	1	0	1
6	0	1	1	0	0	1	1	0
7	0	1	1	1	0	1	1	1
8	1	0	0	0	1	0	0	0
9	1	0	0	1	1	0	0	1
10	1	0	1	0	0001	0	0	0	0
11	1	0	1	1	0001	0	0	0	1
12	1	1	0	0	0001	0	0	1	0
13	1	1	0	1	0001	0	0	1	1
14	1	1	1	0	0001	0	1	0	0
15	1	1	1	1	0001	0	1	0	1

6.6.2C BCD Conversions

Decimal to BCD. To convert the decimal number to its BCD equivalent each decimal digit is replaced by its equivalent 4-bit BCD number. The combination produces the BCD equivalent for the decimal numbers.

Example 6.19 Convert the decimal number $(148.68)_{10}$ *to BCD.*

Solution. Convert each decimal digit to BCD as follows :

1	4	8	.	6	8
↓	↓	↓		↓	↓
0001	0100	1000	.	0110	1000

$\Rightarrow$ (101001000.01101)

BCD to Decimal Conversion. To convert the given BCD number to its decimal equivalent, starting from BCD point break up the BCD number into groups of 4-bits and convert the decimal form.

Example 6.20 Convert the BCD number 10010011.01010101 *to decimal numbers.*

Solution. Convert the BCD number to decimal number equivalent as follows :

1001	0011	.	0101	0101
↓	↓		↓	↓
9	3	.	5	5

$\Rightarrow$ $(93.55)_{10}$

BCD to Binary Conversion. To convert the BCD number binary equivalent first convert the BCD number to its decimal equivalent, then follow the decimal to binary conversion method discussed previously.

Binary to BCD Conversion. This can be accomplished by first converting the binary number to decimal form, then follow the decimal to BCD conversion method discussed previously.

NOTE *BCD code is also termed as 8421 code, here 8, 4, 2 and 1 are weights of the four bits of the binary code from MSB to LSB.*

6.6.2 Non-Weighted Binary Codes

The non-weighted binary codes do not obey positional weighting principle. In these codes, the positional weights are not assigned to the binary digits. Excess-3 and gray codes are non-weighted codes.

6.6.2A Excess-3 Codes

The excess-3 code is derived from 8421(BCD) code. It is also a 4-bit non-weighted binary code. Under this coding scheme the equivalent excess-3 code for each decimal digit is obtained by adding decimal 3 to the natural BCD code of the digit. For example, excess-3 code for decimal number 7 is obtained as 0111 + 0011 = 1010. Excess-3 code is also termed as self-complementing code. In other words, 1's complement of an excess-3 number is the excess-3 code for the 9's complement of the corresponding decimal number. For example, Excess-3 code for the decimal number 3 is 0110, its 1's complement is 1001, which is excess-3 code for decimal number 6, which is 9's complement of 3.

6.6.2B Gray Code

Gray codes are non-weighted code as no weight can be assigned to the bit position. The advantage of gray code over binary codes is that only one bit changes when going from one number to next one. This code is extensively used for shaft encoders due to this property. Also because of this property the possibility of errors decreases. The disadvantage of the gray code is that it cannot be used in arithmetic operations. Before using the Gray code in arithmetic operations the gray code must be converted to binary form.

Gray code to binary conversion. The gray code to binary code conversion involves following steps :

Step 1 The most significant bit (MSB) of the gray code is the MSB of the binary code. Hence write it down.

Step 2 Perform an exclusive OR operation between the binary bit just written down and the next gray code bit.

Step 3 Repeat the step 2, till the least significant bit (LSB) of gray code is Ex-OR with the previous binary bit.

The final pattern achieved will be the binary code equivalent of the gray code. The gray code to binary code conversion method is shown in Fig. 6.7.

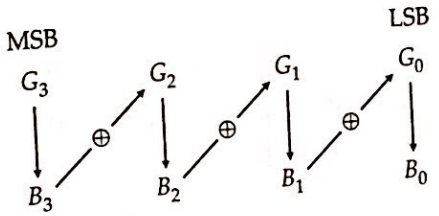


Figure 6.7 Gray code to binary code conversion.

Binary code to gray code conversion. The binary code to gray code conversion involves following steps :

Step 1 The most significant bit of the binary code is the MSB of the gray code. Hence write it down.

Step 2 Perform the Exclusive-OR operation between the MSB of the binary code to the bit immediately on its right and write the result obtained.

Step 3 Repeat step 2 till the least significant bit (LSB) of binary code is Ex-ORed with its immediately left primary code bit.

The final sequence achieved will be the gray code equivalent of the binary code. The binary code to gray code conversion method is shown in Fig. 6.8.

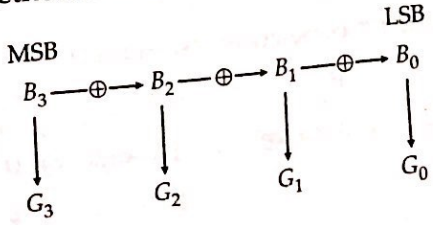


Figure 6.8 Binary code to gray code conversion.

Decimal Number	Primary Code				Gray Code				Excess 3			
	B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	0	1	1	0	1	0	1
3	0	0	1	1	0	0	1	0	0	1	1	0
4	0	1	0	0	0	1	1	0	0	1	1	1
5	0	1	0	1	0	1	1	1	1	0	0	0
6	0	1	1	0	0	1	0	1	1	0	0	1
7	0	1	1	1	0	1	0	0	1	0	1	0
8	1	0	0	0	1	1	0	0	1	0	1	1
9	1	0	0	1	1	1	0	1	1	1	0	0
10	1	0	1	0	1	1	1	1				
11	1	0	1	1	1	1	1	0				
12	1	1	0	0	1	0	1	0				
13	1	1	0	1	1	0	1	1				
14	1	1	1	0	1	0	0	1				
15	1	1	1	1	1	0	0	0				

Example 6.22 Convert the following gray code to its binary code equivalent : (*i*) 1110 (*ii*) 1010.

Solution. (*i*) Given gray code 1110. Now using gray code to binary code conversion method.

G_3		G_2		G_1		G_0		
1		1		1		0		
↓	⊕	↓	⊕	↓	⊕	↓	⇒	(1011)
1		0		1		1		

(*ii*) Given gray code 1010. Now using gray code to binary code conversion method

G_3		G_2		G_1		G_0		
1		0		1		0		
↓	⊕	↓	⊕	↓	⊕	↓	⇒	(1100)
1		1		0		0		

6.6.3 Cyclic (Reflective code)

A binary code is said to be reflective, if the code for 5 is complement for 0, code for 8 is complement for 1, 7 for 2, 6 for 3, 5 for 4. Reflectivity is desirable in a code when the 9's complement is required to be found, for example, as in 9's complement subtraction. Binary codes such as 2421, 5211, Excess-3 are the reflective codes, whereas 8421-code is not.

6.6.4 Sequential Code

The binary code is said to be sequential, when each succeeding code is one binary number greater than its preceding code. The 8421-code, excess-3 code are examples of sequential codes.

6.6.5 Alphanumeric Codes

When the digital system requires to handle the data which may contain numerals, letters and special symbols, then it is necessary to have a binary coding scheme for alphabets and special symbols also. The alphanumeric codes are designed to represent numbers as well as alphabetic characters. The alphabetic characters including the lower case letters, upper case letters and special control characters needs to have unique binary code representation. The *two* most widely used alphanumeric codes are :

1. Extended BCD Interchange Code (EBCDIC)
2. American Standard for Information Interchange (ASCII)

6.6.6 Error Detecting and Correcting Codes

When the digital information is transmitted in the binary form from one digital system to another, it may be possible that the some information is lost during transmission. It may happen because of interference of external noises or some other disturbances with the original signal containing information, and produces an error in the digital signal. Different techniques are used to detect and correct these errors. These techniques are classified as :

1. Error detecting codes :
   (*a*) Even parity method (*b*) Odd parity method (*c*) Block parity method
2. Error correcting codes
   (*a*) Hamming codes (*b*) Block code (*c*) Convolution code (*d*) Cyclic code

6.7 Binary Arithmetics

The digital computer performs various arithmetic operations. It performs addition subtraction, multiplication and division over the binary data. For better understanding of digital circuit operations of a digital system it is necessary to study binary arithmetic.

6.7.1 Binary Addition

In general, for n-bit binary addition maximum 2^n cases may occur. Binary addition is similar to decimal number addition. The rules for 2-bit binary addition is given in Table 6.6.

Table 6.6 *Rules for addition of binary bits.*

Augend	Addend	Sum	Carry
A	B	(S)	(C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

In addition of two binary numbers, the addition takes place bit by bit starting from LSB. The carry generated in previous sum is forwarded to next bit addition.

Example 6.23 *Add binary numbers* $(11101101)_2$ *and* $(11010010)_2$.

Solution.

Carry generated (C_i)	**1**	**1**						
1st binary number (A_i)		1	1	1	0	1	1	0	1
2nd binary number (B_i)		1	1	0	1	0	0	1	0
	1	0	0	1	1	1	1	1	1

Example 6.24 *Add* $(53)_{10}$ *and* $(37)_{10}$ *in binary.*

Solution. Binary equivalents of $(53)_{10}$ and $(37)_{10}$ are computed using decimal to binary conversion method *i.e.,*

$$(53)_{10} = (110101)_2$$

$$(57)_{10} = (100101)_2$$

Now perform the binary addition

Carry	**1**		**1**		**1**		
1st binary number $(53)_{10}$		1	1	0	1	0	1
2nd binary number $(37)_{10}$		1	0	0	1	0	1
$(90)_{10}$	1	0	1	1	0	1	0

Table 6.7

Minued	Subtrahend	Difference	Borrow
A	B	D	Br
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

In subtraction of two binary numbers, the subtraction takes place bit by bit strating from LSB.

Example 6.25 Subtract $(0111)_2$ from $(1100)_2$.

Solution.

Column	B_3	B_2	B_1	B_0
Borrow		1	1	
Minued	1	1	0	1
Subtrahend	0	1	1	1
Difference	0	1	1	0

NOTE *(i) At column B_1, due to borrow the subtraction takes place as under*

$$(10)_2 - (1)_2 = (1)_2$$

(ii) At column B_2, due to borrow the subtraction takes place as under

$$(10)_2 - (1)_2 = (1)_2$$

Example 6.26 *Subtract decimal number $(37)_{10}$ from $(53)_{10}$ in binary.*

Solution. Binary equivalents of $(53)_{10}$ and $(57)_{10}$ are computed using decimal to binary conversion method *i.e.,*

$$(53)_{10} = (110101)_2$$

$$(37)_{10} = (100101)_2$$

Now perform the binary subtraction

Borrow						
1st number $(53)_{10}$	=	1	1	0	1	0	1
2nd number $(37)_{10}$	=	1	0	0	1	0	1
$(16)_{10}$	=	0	1	0	0	0	0

$\Rightarrow$ $(16)_{10} = (01000)_2$

6.7.3 Binary Subtraction using 1's Complement and 2's Complement

As the number of bits in a binary number increases, the binary subtraction becomes tedious and time-consuming. Hence the subtraction of two binary numbers $A_i - B_i$ $(i = 0,1,\ldots\ldots,n)$, can be achieved using binary addition as $A_i + (-B_i)$. The subtrahend B_i is represented in 1's complement or 2's complement form and the resultant is added to minued (A_i), to obtain $A_i - B_i$.

1's Complement Subtraction

Subtraction of two binary numbers $(A_i - B_i)$, using 1's complement involves following steps :

Step 1 Find the 1's complement of the subtrahend B_i.

Step 2 Add the minued A_i to the resultant of step 1.

Step 3 Check the carry, if the carry is generated, answer is positive ; add the carry in LSB position to the resultant of step 2. If the carry is not generated, answer is negative then take 1's complement to the resultant of step 2 and put a negative sign in front.

Example 6.27 *Use 1's complement to perform $A_i - B_i$ with the given binary numbers :*

(*i*) $A_i = 1010$, $B_i = 1001$ (*ii*) $A_i = 1001$, $B_i = 1010$

Solution. (*i*) Take the 1's complement of subtrahend *i.e.*, $B_i = 1001$, which is 0110 and add it to minued A_i *i.e.*, 1010.

	A_3	A_2	A_1	A_0
Minued	1	0	1	0
1's complement of subtrahend	0	1	1	0
	1	0	0	0

Carry ↑ ; ↓ resultant

Add the carry with resultant

	0	0	0	0
				1
	0	0	0	1

The result is $(0001)_2$.

(*ii*) Take the 1's complement of the subtrahend *i.e.*, $B_i = 1010$, which is 0101 and add it to minued A_i *i.e.*, 1001,

	A_3	A_2	A_1	A_0
Minued	1	0	0	1
1's complement of subtrahend	0	1	0	1
Resultant	1	1	1	0

Since no carry is generated, take the 1's complement of resultant and put negative sign *i.e.*,

$-(0\ 0\ 0\ 0\ 1)_2$

The result is $-(0\ 0\ 0\ 0\ 1)_2$.

2's Complement Subtractions

Subtraction of two binary numbers $(A_i - B_i)$ using 2's complement method involves following steps :

Step 1 Find the 2's complement of the subtrahend B_i.

Step 2 Add the minued A_i to the resultant of step 1.

Step 3 Check the carry. If the carry is generated discard it. The result is a positive. If the carry is not generated, answer is negative, then take the 2's complement of the result and place a negative sign in front.

Example 6.28 *Use 2's complement method to perform $A_i - B_i$ with the given binary numbers :*

(*i*) $A_i = 0100$, $B_i = 1001$ (*ii*) $A_i = 1001$, $B_i = 0101$

Solution. (*i*) Take the 2's complement of the subtrahend β_i, which is 0111 and add it to the minued A_i *i.e.*, 0100.

Minued		0	1	0	0
2's complement of subtrahend +		0	1	1	1
		1	0	1	1 ← Resultant

Since no carry is generated, the answer is negative and is in the 2's complement form. Now take the 2's complement of the resultant obtained *i.e.*,

1	0	1	1	← Resultant
0	1	0	0	← 1's complement
+			1	
0	1	0	1	

The result is $-(0101)_2$.

(*ii*) Take the 2's complement of the subtrahend B_i which is 1011, and add it to the minued A_i, *i.e.*, 1001

Minued		1	0	0	1
2's complement of subtrahend		1	0	1	1
Carry →	1	0	1	0	0 ← Resultant

As the carry bit is generated, the answer is positive and discard it.

The result is $(0100)_2$.

6.7.4 Binary Multiplication

Binary multiplication is also similar to decimal multiplication. The rules for 2-bit binary multiplication is given in Table 6.8.

Table 6.8 *Rules for multiplication of binary bits.*

A	B	Y = (A × B)
0	0	0
0	1	0
1	0	0
1	1	1

Example 6.29 *Multiply* 1001 *by* 1010.

Solution.

$$\begin{array}{r} 1\ 0\ 0\ 1 \\ \times\ 1\ 0\ 1\ 0 \\ \hline 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 1\ \times \\ 0\ 0\ 0\ 0\ \times\ \times \\ 1\ 0\ 0\ 1\ \times\ \times\ \times \\ \hline 1\ 0\ 1\ 1\ 0\ 1\ 0 \end{array}$$

The result is $(1011010)_2$.

6.7.5 Binary Division

Binary division is also similar to decimal division. The rules for 2-bit binary division are given in Table 6.9.

Table 6.9 *Rules for division of binary bits.*

A	B	Y = (A ÷ B)
0	0	0
0	1	0
1	0	undetermined
1	1	1

Example 6.30 *Divide* 11001 *by* 101.

Solution.

$$\begin{array}{r} 1\ 0\ 1 \quad \leftarrow \text{Quotient} \\ 1\ 0\ 1 \overline{)\ 1\ 1\ 0\ 0\ 1} \\ 1\ 0\ 1 \\ \hline 0\ 0\ 1\ 0\ 1 \\ \quad 1\ 0\ 1 \\ \hline 0\ 0\ 0 \quad \leftarrow \text{Remainder} \end{array}$$

The result is $(101)_2$.

Formulae at a Glance

6.1 *Decimal to Radix conversion.* For integer part, divide the decimal number by Radix R, producing quotient and remainder. The quotient obtained in previous step is further divided by radix R, until the quotient becomes zero. Now arrange the remainder from bottom to top to give equivalent representation. For fractional part, repeated multiplication method is used.

6.2 *Radix R to decimal conversion.* Any number with radix R, having m digits to the left and n digits to the right of the radix point can be expressed to its decimal equivalent as :

$$\alpha_m(R)^{m-1} + \alpha_{m-1}(R)^{m-2} + \ldots\ldots + \alpha_2(R)^1 + \alpha_1(R)^0 + \beta(R)^{-1} + \beta_2(R)^{-2} + \ldots\ldots + \beta_n(R)^{-1}$$

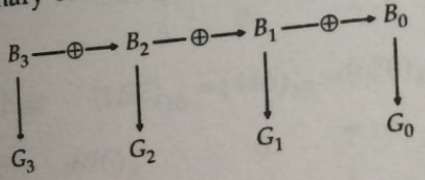
where

$(R)^r.(r = m-1, m-2, \ldots\ldots, 0, -1, -2, \ldots\ldots, -n)$ represents the positional weight. α_i and β_i are the digit values and N is the equivalent decimal number.

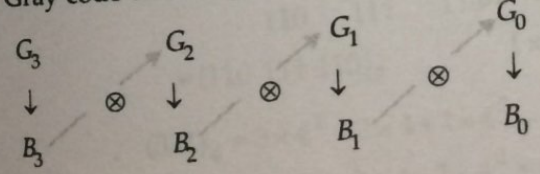
6.3 1's complement of any binary sequence can be obtained by changing all 1 to 0 and all 0 to 1.

6.4 2's complement of any binary sequence can be obtained by adding 1 to its 1's complement.

6.5 Binary code to gray code conversion



6.6 Gray code to binary code conversion



6.7 Binary addition

Augend	Addend	Sum	Carry
A	B	(S)	(C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

6.8 Binary subtraction

Minued	Sub-trahend	Diff-erence	Borrow
A	B	D	Br
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

6.9 Binary multiplication

A	B	$Y = (A \times B)$
0	0	0
0	1	0
1	0	0
1	1	1

6.10 Binary division

A	B	$Y = (A \div B)$
0	0	0
0	1	0
1	0	undetermined
1	1	1

Miscellaneous Solved Numerical Problems

Problem 6.1 *Obtain the binary equivalent of the following decimal numbers : (a)* $(35)_{10}$ *(b)* $(105)_{10}$

Solution. (a)

Division		Remainder	
2	35		
2	17	1	(LSB)
2	8	1	
2	4	0	
2	2	0	
2	1	0	
	0	1	(MSB)

The binary equivalent of $(35)_{10} = (100011)_2$.

(b)

Division		Remainder	
2	105		
2	52	1	(LSB)
2	26	0	
2	13	0	
2	6	1	
2	3	0	
2	1	0	
	0	1	(MSB)

The binary equivalent of $(105)_{10} = (1001001)_2$.

Problem 6.2 *Convert hexadecimal number* $(B5A)_{16}$ *and* $(32E)_{16}$ *into decimal number.*

Solution. $(B5A)_{16} = B\times16^2 + 5\times16^1 + A\times16^0$

$= 11\times256 + 5\times16 + 10\times1$

$= 2816 + 80 + 10 = (2906)_{10}$

Thus $(B5A)_{16} = (2906)_{10}$

$(32E)_{16} = 3\times16^2 + 2\times16^1 + E\times16^0$

$= 3\times256 + 2\times16 + 14\times1$

$= 768 + 32 + 14 = 814$

Thus $(32E)_{16} = (814)_{10}$

Problem 6.3 *...* $_{10}$ *and* $(65)_{10}$ *into octal number.*

Solution.

Division		Remainder	
8	15		
8	1	7	(LSB)
	0	1	(MSB)

Thus $(15)_{10} = (17)_8$.

Division		Remainder	
8	65		
8	8	1	(LSB)
8	1	0	
	0	1	(MSB)

Thus $(65)_{10} = (101)_8$.

Problem 6.4 *Convert the following numbers as indicated :*

(a) $(IBE)_{16} = (\ )_8$ *(b)* $(676)_8 = (\ )_2$ *(c)* $(321)_4 = (\ )_{10}$

Solution. (a) In order to convert the hexadecimal number into octal number, first we convert the hexadecimal number into decimal number then the resultant is converted into binary i.e.,

$(1BE)_{16} = 1\times16^2 + B + 16^1 + E\times16^0$

$= 1\times256 + 11\times16 + 14\times1$

$= 256 + 176 + 14 = (446)_{10}$

Now

Division		Remainder	
8	446		
8	55	6	(LSB)
8	6	7	
	0	6	(MSB)

Thus $(1BE)_{16} = (446)_{10} = (676)_8$

(b) $(676)_8$ 6 7 6

↓ ↓ ↓

110 111 110

$= (110\ 111\ 110)_2$

(c) $(321)_4 = 3\times4^2 + 2\times4 + 1\times4^0$

$= 3\times16 + 2\times4 + 1\times4^0 = 48 + 8 + 1 = (57)_{10}$

Problem 6.5 *Convert* $(2861)_{10}$ *into BCD code.*

Solution. Convert each decimal digit to BCD as follows :

2	8	6	1
↓	↓	↓	↓
0010	1000	0110	0001

$\Rightarrow$ $(0010\ 1000\ 0110\ 0001)_2$

Problem 6.6 *Convert the decimal number* $(250.5)_{10}$ *to base-3, base-4, base-7, base-8, base-16.* **[GGSIPU, Dec. 2013 (3 marks)]**

Solution. (*a*) **Decimal to Base 3.**

Step (*i*) : $(250.5)_{10} = (250)_{10} + (0.5)_{10}$

Step (*ii*) : ***Integer conversion***

Division		Remainder	
3	250		
3	83	1	(LSB)
3	27	2	
3	9	0	
3	3	0	
3	1	1	(MSB)
	0		

$\Rightarrow$ $(250)_{10} = (10021)_3$

Step (*iii*) : ***Fraction conversion***

Multiplication	Integer		Fraction
$0.5 \times 3 = 1.5$	1	(MSB)	0.5
$0.5 \times 3 = 1.5$	1		0.5
$0.5 \times 3 = 1.5$	1	(LSB)	0.5

$\Rightarrow$ $(0.5)_{10} = (111)_3$

Step (*iv*) : ***Combination***

$(250.5)_{10} = (10021.111)_3$

(*b*) **Decimal to Base-4**

Step (*i*) : ***Separation***

Decimal Number = Integer Part + Fractional Part

$(250.5)_{10} = (250)_{10} + (0.5)_{10}$

Step (*ii*) : ***Integer conversion***

Division		Remainder	
4	250		
4	62	2	(LSB)
4	15	2	
4	3	3	
	0	3	(MSB)

$\Rightarrow$ $(250)_{10} = (3322)_4$

Step (*iii*) : ***Fraction conversion***

Multiplication	Integer		Fraction
$0.5 \times 4 = 2.00$	2	(MSB)	0.0
$0.0 \times 4 = 0.00$	1	(LSB)	0.0

$\Rightarrow$ $(0.5)_{10} = (0.20)_4$

Step (*iv*) : ***Combination***

$\Rightarrow$ $(250.5)_{10} = (3322.20)_4$

(*c*) **Decimal to Base-7**

Step (*i*) : ***Separation***

$(250.5)_{10} = (250)_{10} + (0.5)_{10}$

Step (*ii*) : ***Integer conversion***

Division		Remainder	
7	250		
7	35	5	(LSB)
7	5	0	
	0	5	(MSB)

$\Rightarrow$ $(250.5)_{10} = (505)_7$

Step (*iii*) : ***Fraction conversion***

Multiplication	Integer		Fraction
$0.5 \times 7 = 3.5$	3	(MSB)	0.5
$0.5 \times 7 = 3.5$	3		0.5
$0.5 \times 7 = 3.5$	3	(LSB)	0.5

$\Rightarrow$ $(0.5)_{10} = (0.333)_7$

Step (*iv*) : ***Combination***

$(250.5)_{10} = (505.333)_7$

(d) Decimal to Base-8

Step (i) : ***Separation***

$$(250.5)_{10} = (250)_{10} + (0.5)_{10}$$

Step (ii) : ***Integer conversion***

	Division		Remainder	
8	250			
8	31	⟶	2	(LSB) ↑
8	3	⟶	7	
	0	⟶	3	(MSB)

$$\Rightarrow \quad (250)_{10} = (372)_8$$

Step (iii) : ***Fraction conversion***

Multiplication	Integer		Fraction
$0.5 \times 8 = 4.0$	4	(MSB) ↓	0.0
$0.0 \times 8 = 0.0$	0	(LSB)	0.0

$$\Rightarrow \quad (0.5)_{10} = (0.40)_8$$

Step (iv) : ***Combination***

$$(250.5)_{10} = (372.40)_8$$

(e) Decimal to Base-16

Step (i) : ***Separation***

$$(250.5)_{10} = (250)_{10} + (0.5)_{10}$$

Step (ii) : ***Integer conversion***

	Division		Remainder	
16	250			
16	15	⟶	A	(LSB) ↑
	0	⟶	F	(MSB)

$$\Rightarrow \quad (250)_{10} = (FA)_{16}$$

Step (iii) : ***Fraction conversion***

Multiplication	Integer		Fraction
$0.5 \times 16 = 8.0$	8	(MSB) ↓	0.0
$0.0 \times 16 = 0.0$	0	(LSB)	0.0

$$\Rightarrow \quad (0.5)_{10} = (0.80)_{16}$$

Step (iv) : ***Combination***

$$(250.5)_{10} = (FA.80)_{16}$$

Conceptual Questions

6.1 *What is meant by LSB and MSB ?*

Ans. LSB stands for least significant bit. It is rightmost bit of a binary number. MSB stands for most significant bit and it is the leftmost bit of a binary number.

6.2 *Explain number system.*

Ans. A number is a collection of symbols. Each symbol's value is a function of the type of system and its position. All the number systems are positional.

6.3 *What is meant by BCD code ?*

Ans. BCD stands for binary coded decimal. In this coding scheme each digit of the decimal number is represented by a four bit binary number.

6.4 *Explain the advantages of gray code.*

Ans. Gray code number differs from the preceding and succeeding number by a single bit, because of this property it is used in shaft encoders.

6.5 *Explain the use of Excess-3 code.*

Ans. Excess-3 code is a self complementary code. It means that 1's complement of the coded number yields 9's complement of the number itself. The self complementary property of this code helps in performing subtraction operation in digital systems.

6.6 *What is 1's complement number ?*

Ans. The 1's complement of a binary number can be obtained by changing 1s to 0s and 0s to 1s.

6.7 *What is 2's complement number ?*

Ans. The 2's complement of a binary number can be obtained by adding 1 to its 1's complement.

6.8 *What are the basic rules for binary addition ?*

Ans. The basic rules for binary addition are

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10_2$$

6.9 *What are the basic rules for binary subtraction ?*

Ans. The basic rules for binary subtraction are

$$0 - 0 = 0$$
$$1 - 1 = 0$$
$$1 - 0 = 1$$
$$10_2 - 1 = 1$$

6.10 *Show that positive logic AND gate is negative logic OR gate and vice versa.* **[GGSIPU, Dec. 2013 (2-5 marks)]**

Ans. In the case of positive logic, the lower voltage represents low or 0 state and the higher voltage represents high or 1 state.

Whereas in the case of negative logic, the lower voltage represents high or 1 state and the higher voltage represents high or 0 state. The following table shows the two assignments that defines positive and negative logic systems.

Postive logic	*Negative logic*
$H = 1$	$H = 0$
$L = 0$	$L = 1$

The physical gate has two different interpretations depending on if positive logic or negative logic is used.

PHYSICAL

A	B	F
L	L	L
L	H	L
H	L	L
H	H	H

Postive logic

A	B	F (AND)
0	0	0
0	1	0
1	0	0
1	1	1

Negative logic

A	B	F (OR)
1	1	1
1	0	1
0	1	1
0	0	0

Hence from above it is clear that positive logic AND gate is negative logic OR gate and vice versa.

6.11 *Define base (or radix) of a number system.*

Ans. The number of distinct symbols (digits) used in a number system is called radix or base of a number system. For example, in decimal number system 10 symbols (0,1,2,3,4,5,6,7,8,9) are used, so the radix for decimal number system is 10.

6.12 *Define sign bit.*

Ans. The MSB of a signed binary number is termed as sign bit. If it is 0 the number is considered as positive number, and if it is 1 the number is considered as negative number.

6.13 *What is weighted code ?*

Ans. Weighted code is a binary code in which weight assigned to each digit position in the number.

6.14 *What is non-weighted code ?*

Ans. A non-weighted code is a binary code in which no weight assigned to each digit position in the number.

6.15 *What is advantage of gray code ?*

Ans. Gray Code is a binary code in which only one bit changes between successive numbers. Because of this feature amount of switching is minimized and the system reliability increases.

Exercises

Theoretical Questions

6.1 Write the characteristics of a digital system.

6.2 What do you mean by gray code ? Explain the method for the conversion of Gray code to its binary code equivalent.

6.3 Explain briefly (*a*) *r*'s complement (*b*) (*r*−1)'s complement.

6.4 Briefly define each of the following :

(*a*) radix (*b*) 2's complement (*c*) sign magnitude representation (*d*) 8421-code

6.5 What do you understand by weighted code ? Discuss its area of applications.

6.6 Why we need to study hexadecimal system ? When the digital machine understand only and 1 ?

6.7 Explain need fo error defecting and correcting codes.

6.8 Why octal number system is used in digital systems ?

6.9 What is meant by continuous and discrete time signals and systems ?

6.10 Explain why digital systems are preferred over analog systems ?

6.11 Explain binary, octal and hexadecimal number systems. **[GGSIPU, June 2012, 2.5**

6.12 Explain with examples 1's and 2's complement of a number and its uses. **[GGSIPU, June 2012, 2.**

6.13 Explain the difference between binary and BCD codes.

6.14 Explain the rules for subtraction using the 1's complement and 2's complement.

6.15 Write a short note on weighted and non-weighted codes.

6.16 What is gray code ? Why it is important ?

6.17 What is BCD code ? What are its advantages and disadvantages ?

Numerical Problems

6.1 Convert the $(24.4)_{10}$ to its binary and octal number equivalent. **[Ans.** (01000.(