CONTENTS

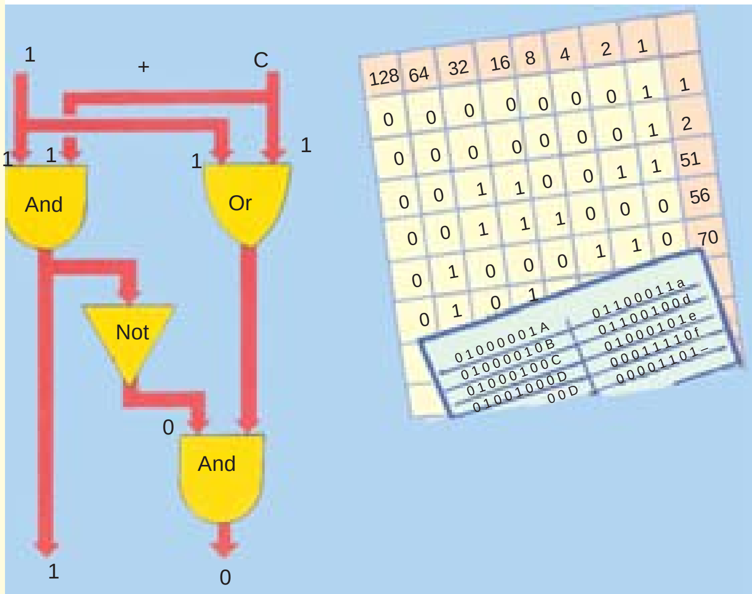
CHAPTER 69

NUMBER SYSTEMS AND CODES

Learning Objectives

- Number Systems
- The Decimal Number System
- Binary Number System
- Binary to Decimal Conversion
- Binary Fractions
- Double-Dadd Method
- Decimal to Binary Conversion
- Shifting the Place Point
- Binary Operations
- Binary Addition
- Binary Subtraction
- Complement of a Number
- 1's Complemental Subtraction
- 2's Complemental Subtraction
- Binary Multiplication
- Binary Division
- Shifting a Number to Left or Right
- Octal Number System
- Octal to Decimal Conversion
- Decimal to Octal Conversion
- Binary to Octal Conversion
- Octal to Binary Conversion
- Binary to Hexadecimal Conversion
- Decimal to Hexadecimal Conversion
- Hexadecimal to Decimal Conversion
- Digital Coding
- Binary Coded Decimal (BCD) Code
- Octal Coding
- Hexadecimal Coding
- Excess-3 Code
- Gray Code
- Excess-3 Gray Code
- ASCII Code

CONTENTS



Logic gates use switches that control the flow of an electrical current. '1' is 'true' and '0' is 'false'. The columns in the binary system wave the values 1, 2, 4, 8, 16, 32 and so no.

69.1. Number Systems

The number systems are used quite frequently in the field of digital electronics and computers. However the type of number system used in computers could be different at different stages of the usage. For example, when a user key-in some data into the computer, s(he), will do it using decimal number system *i.e.* the system we all have used for several years for doing arithmetic problems. But when the information goes inside the computer, it needs to be converted to a form suitable for processing data by the digital circuitry. Similarly when the data has to be displayed on the monitor for the user, it has to be again in the decimal number system. Hence the conversion from one number system to another one is an important topic to be understood.

There are four systems of arithmetic which are often used in digital circuits. These systems are:

1. **Decimal**—it has a base (or radix) of 10 *i.e.* it uses 10 different symbols to represent numbers.
2. **Binary**—it has a base of 2 *i.e.* it uses only two different symbols.
3. **Octal**—it has a base of 8 *i.e.* it uses eight different symbols.
4. **Hexadecimal**—it has a base of 16 *i.e.* it uses sixteen different symbols.

All these systems use the same type of **positional notation** except that

— decimal system uses powers of 10 — binary system uses power of 2

— octal system uses powers of 8 — hexadecimal system uses powers of 16.

Decimal numbers are used to represent quantities which are outside the digital system. Binary system is extensively used by digital systems like digital computers which operate on binary information. Octal system has certain advantages in digital work because it requires less circuitry to get information into and out of a digital system. Moreover, it is easier to read, record and print out octal numbers than binary numbers. Hexadecimal number system is particularly suited for microcomputers.

69.2. The Decimal Number System

We will briefly recount some important characteristics of this more-familiar system before taking up other systems. This system has a base of 10 and is a **position-value system** (meaning that value of a digit depends on its **position**). It has following characteristics :

(*i*) Base or Radix

It is defined as ***the number of different digits which can occur in each position in the number system.***

The decimal number system has a base of 10 meaning that it contains ten unique symbols (or digits). These are : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Any one of these may be used in each position of the number.

Incidentally, it may be noted that we call it a decimal (10's) system although ***it does not have a distinct symbol of 10.*** As is well-known, it expresses 10 and any number above 10 as a combination of its ten unique symbols.

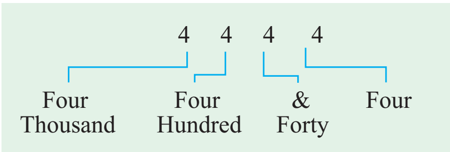


Fig. 69.1

(*ii*) Position Value

The absolute value of each digit is fixed but its **position value** (or place value or weight) is determined by its ***position*** in the overall number. For example, position value of 3 in 3000 is not the same as in 300. Also, position value of each 4 in the number 4444 is different as shown in Fig. 69.1.

Similarly, the number 2573 can be broken down as follows :

$$2573 = 2 \times 10^3 + 5 \times 10^2 + 7 \times 10^1 + 3 \times 10^0$$

It will be noted that in this number, 3 is the **least significant digit** (*LSD*) whereas 2 is the **most significant digit** (*MSD*).

Again, the number 2573.469 can be written as

$$2573.469 = 2 \times 10^3 + 5 \times 10^2 + 7 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 6 \times 10^{-2} + 9 \times 10^{-3}$$

It is seen that position values are found by raising the base of the number system (*i.e.* 10 in this case) to the power of the position. Also, powers ***are numbered to the left of the decimal point starting with 0 and to the right of the decimal point starting with –1.***

69.3. Binary Number System

Like decimal number (or denary) system, it has a radix and it also uses the same type of position value system.

(*i*) Radix

Its base or radix is **two** because it uses only two digits 0 and 1 (the word 'binary digit' is contracted to **bit**). All binary numbers consist of a string of 0s and 1s. Examples are 10, 101 and 1011 which are read as one-zero, one-zero-one and one-zero-one-one to avoid confusion with decimal numbers. Another way to avoid confusion is to add a subscript of 10 for decimal numbers and of 2 for binary numbers as illustrated below.

10_{10}, 101_{10}, 5742_{10} —decimal number and 10_2, 101_2, 110001_2 — binary numbers.

Binary numbers represent all values within computers

It is seen that the subscript itself is in decimal. It may be noted that binary numbers need ***more places for counting because their base is small***

(*ii*) Position Value

Like the decimal system, binary system is also positionally-weighted. However, in this case, the position value of each bit corresponds to some power of 2. In each binary number, the value increases in powers of 2 starting with 0 to the left of the binary point and decreases to the right of the binary point starting with power of –1. The position value (or weight) of each bit alongwith a 7-bit binary number 1101.011 is shown in Fig. 69.2.

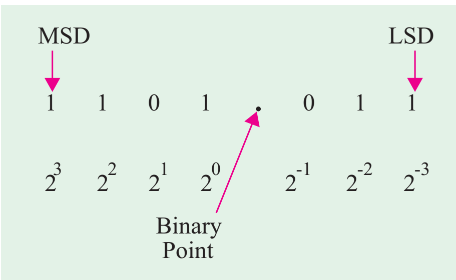


Fig. 69.2

As seen, the fourth bit to the left of binary point **carries the maximum weight** (*i.e.* it has the highest value) and is called most significant digit (*MSD*). Similarly, the third bit to the right of the binary point is called least significant digit (*LSD*). The decimal equivalent of the binary number may be found as under

$$1101.011_2 = (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) + (1 \times 2^{-3})$$

$$= 8 + 4 + 0 + 1 + 0 + \frac{1}{4} + \frac{1}{8} = 13.375_{10}$$

As stated earlier, position values of different bits are given by **ascending powers of 2** to the **left** of binary point and by **descending power** of 2 to the **right** of binary point. The different digit positions of a given binary number have the following decimal weight (Fig. 69.3)

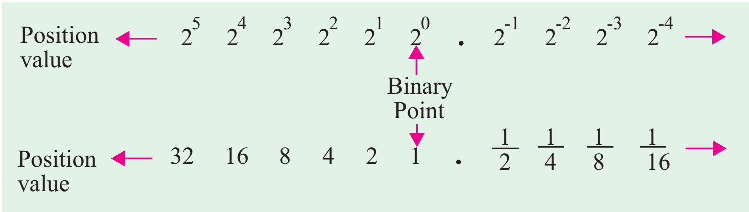


Fig. 69.3

Binary numbers are used extensively by all digital systems primarily due to the nature of electronics itself. The bit 1 may be represented by a saturated (fully-conducting) transistor, a light turned ON, a relay energised or a magnet magnetised in a particular direction. The bit 0, on the other hand, can be represented as a cut-off transistor, a light turned OFF, a relay de-energised or a magnet magnetised in the opposite direction. In such cases, there are only two values which a device can assume.

69.4. Binary to Decimal Conversion

Following procedure should be adopted for converting a given binary integer (whole number) into its equivalent decimal number :

Step 1. Write the binary number *i.e.* all its bits in a row.

Step 2. Directly under the bits, write 1, 2, 4, 8, 16, ...starting from *right to left*.

Step 3. Cross out the decimal weights which lie under 0 bits.

Step 4. Add the remaining weights to get the decimal equivalent.

Example 69.1. *Convert 11001_2 to its equivalent decimal number.*

Solution. The four steps involved in the conversion are as under

Step 1.	1	1	0	0	1
Step 2.	16	8	4	2	1
Step 3.	16	8	$\cancel{4}$	$\cancel{2}$	1
Step 4.	16 + 8 + 1 = 25			$\therefore$	$11001_2 = 25_{10}$

It is seen that the number contains 1 sixteen, one eight, 0 four's, 0 two's and 1 one. Certain decimal and binary equivalent numbers are tabulated below in Table No. 69.1

Table No. 69.1

Decimal	*Binary*	*Decimal*	*Binary*	*Decimal*	*Binary*
1	1	11	1011	21	10101
2	10	12	1100	22	10110
3	11	13	1101	23	10111
4	100	14	1110	24	11000
5	101	15	1111	25	11001
6	110	16	10000	26	11010
7	111	17	10001	27	11011
8	1000	18	10010	28	11100
9	1001	19	10011	29	11101
10	1010	20	10100	30	11110

69.5. Binary Fractions

Here, procedure is the same as for binary integers except that the following weights are used for different bit positions .

●	2^{-1}	2^{-2}	2^{-2}	2^{-4}	$\rightarrow$
$\uparrow$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\rightarrow$

Binary Point

Example 69.2. *Convert the binary fraction 0.101 into its decimal equivalent.*

Solution. The following four steps will be used for this purpose.

Step 1.	0	●	1	0	1
Step 2.			$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$
Step 3.			$\frac{1}{2}$	$\cancel{\frac{1}{4}}$	$\frac{1}{8}$

Step 4. $\quad \frac{1}{2}+\frac{1}{8} = 0.625$

$\therefore \quad 0.101_2 = 0.625_{10}$

Example 69.3. *Find the decimal equivalent of the 6-bit binary number 101.101_2.*

Solution.

1	0	1	1	●	0	1	
4	2	1	$\frac{1}{2}$		$\frac{1}{4}$	$\frac{1}{8}$	
4	$\cancel{2}$	1	$\frac{1}{2}$		$\cancel{\frac{1}{4}}$	$\frac{1}{8}$	$= 5 + \frac{1}{2} + \frac{1}{8} = 5.625$

$\therefore \quad 101.101_2 = 5.625_{10}$

69.6. Double-Dadd Method

This method of converting binary integers into decimal equivalents is much simpler and quicker than the method given in Art. 69.4 especially in the case of large numbers. Following three steps are involved :

1. Double the first bit to the extreme left and add this doubled value to the next bit on the right.
2. Double the sum obtained and add the doubled value to the next bit.
3. Continue step 2 until the last bit has been added to the previously-doubled sum.

The conversion of 11001_2 is shown in Fig. 69.4. It is seen that $11001_2 = 25_{10}$

$$1 \quad 1 \quad 0 \quad 0 \quad 1$$

$$\boxed{2\times1} = 2,\ 2+\boxed{1} = 3,\ \boxed{2\times3} = 6,\ 6+\boxed{0} = 6,\ \boxed{2\times6} = 12,\ 12+\boxed{0} = 12,\ \boxed{2\times12} = 24,\ 24+\boxed{1} = 25$$

Fig. 69.4

Using double-dadd method, let us convert 111010_2 into its binary equivalent.

1. $2 \times 1 = 2,$ add next bit 1 so that $2 + 1 = 3$
2. $2 \times 3 = 6,$ add next bit 1 so that $6 + 1 = 7$
3. $2 \times 7 = 14,$ add next bit 0 so that $14 + 0 = 14$
4. $2 \times 14 = 28,$ add next bit 1 so that $28 + 1 = 29$

5. $2 \times 29 = 58$, add next bit 0 so that 58+0 = 58

$\therefore \quad 111010_2 = 58_{10}$

69.7. Decimal to Binary Conversion

(*a*) Integers

Such conversion can be achieved by using the so-called **double-dabble method.** It is also known as **divide-by-two** method. In this method, we progressively divide the given decimal number by 2 and write down the remainders after each division. These remainders taken in the *reverse* order (*i.e.* from bottom-to-top) form the required binary number. As an example, let us convert 25_{10} into its binary equivalent.

	$25 \div 2$	= 12 + remainder of 1	TOP
	$12 \div 2$	= 6 + remainder of 0	↑
	$6 \div 2$	= 3 + remainder of 0	
	$3 \div 2$	= 1 + remainder of 1	
	$1 \div 2$	= 0 + remainder of 1	
$\therefore$	25_{10}	$= 11001_2$	BOTTOM

The above process may be simplified as under :

Successive Divisions	***Remainders***
2) 2 5	
2) 1 2	1
2) 6	0
2) 3	0
2) 1	1
2) 0	1

Reading the remainders from bottom to top, we get $25_{10} = 11001_2$

It may also be put in the following form :

$$\begin{aligned} 25 \div 2 &= 12 + 1 \\ 12 \div 2 &= 6 + 0 \\ 6 \div 2 &= 3 + 0 \\ 3 \div 2 &= 1 + 1 \\ 1 \div 2 &= 0 + 1 \end{aligned}$$

$\therefore$ decimal 25 = 1 1 0 0 1 binary

(*b*) Fractions

In this case, **Multiply-by-two** rule is used *i.e.* we multiply each bit by 2 and record the carry in the integer position. These carries taken in the **forward** (top-to-bottom)direction gives the required binary fraction.

Let us convert 0.8125_{10} into its binary equivalent.

	0.8125×2	= 1.625	= 0.625	with a carry of 1
	0.625×2	= 1.25	= 0.25	with a carry of 1
	0.25×2	= 0.5	= 0.5	with a carry* of 0
	0.5×2	= 1.0	= 0.0	with a carry of 1
$\therefore$	0.8125_{10}	$= 0.1101_2$		

* It is so because there is 0 at the interger position *i.e.* to the left of the decimal point.

Please note that ***we have to add the binary point from our side***. Let us now convert 0.77_{10} into its binary equivalent.

$$\begin{array}{lclcl} 0.77 \times 2 & = & 1.54 & = & 0.54 \text{ with a carry of } 1 \\ 0.54 \times 2 & = & 1.08 & = & 0.08 \text{ with a carry of } 1 \\ 0.08 \times 2 & = & 0.16 & = & 0.16 \text{ with a carry of } 0 \\ 0.16 \times 2 & = & 0.32 & = & 0.32 \text{ with a carry of } 0 \\ 0.32 \times 2 & = & 0.64 & = & 0.64 \text{ with a carry of } 0 \\ 0.64 \times 2 & = & 1.28 & = & 0.28 \text{ with a carry of } 1 \end{array} \Big\downarrow$$

We may stop here but the answer would be approximate. $\therefore\ 0.77_{10} \cong .110001_2$

Example 69.4. *Convert 25.625_{10} into its binary equivalent.*

Solution. We will do the conversion in two steps ***(i)*** first for the integer and ***(ii)*** then for the fraction.

(*a*) Integer

$$\begin{array}{rcl} 25 \div 2 & = & 12 + 1 \\ 12 \div 2 & = & 6 + 0 \\ 6 \div 2 & = & 3 + 0 \\ 3 \div 2 & = & 1 + 1 \\ 1 \div 2 & = & 0 + 1 \\ \therefore\ 25_{10} & = & 11001_2 \end{array} \Big\uparrow$$

(*b*) fraction

$$\begin{array}{rcl} 0.625 \times 2 & = & 1.25 = 0.25 + 1 \\ 0.25 \times 2 & = & 0.5 = 0.5 + 0 \\ 0.5 \times 2 & = & 1.0 = 0.0 + 1 \\ \therefore\ 0.625_{10} & = & 0.101_2 \end{array} \Big\downarrow$$

Considering the complete number, we have $25.625_{10} = 11001.101_2$

Obviously, binary system needs more bits to express the same number than decimal system.

69.8. Shifting the Place Point

In a decimal number if the decimal point is moved one place to the right, **the number is multiplied by 10**. For example, when decimal point in 7.86 is shifted one place to the right, it becomes 78.6 *i.e.* it increases the value of the number 10 times. Moving the decimal point one place to the left reduces its value to one-tenth.

In binary numbers, shifting the binary point by one place multiplies or divides the number by 2. For example, 111.0_2 is equal to 7_{10} but 1110.0_2 is 14_{10}. As seen, 7 is doubled to 14 by moving the binary point one place to the right.

Similarly, 11.1_2 is $(2 + 1 + \frac{1}{2}) = 3.5_{10}$. Hence, 111.0_2 is halved to 3.5_{10} by moving its binary point one place to the left.

69.9. Binary Operations

We will now consider the following four binary operations :

1. addition **2.** subtraction **3.** multiplication **4.** division

Addition is the most important of these four operations. In fact, by using **'complements'**, subtraction can be reduced to addition. Most digital computers subtract by complements. It leads to reduction in hardware because only adding type of circuits are required. Similarly, multiplication is nothing but repeated addition and, finally, division is nothing but repeated subtraction.

69.10. Binary Addition

Addition is simply the manipulation of numbers for combining physical quantities. For example, in the decimal number system, 2 + 3 = 5 means the combination of • • with • • • to give a total of • • • • •. Addition of binary numbers is similar to the decimal addition.

Following points will help in understanding the rules of binary addition.

1. When 'nothing' is combined with 'nothing', we get nothing.
Binary representation of the above statement is : 0 + 0 = 0

2. When nothing is combined with •, we get •.
In binary language 0 + 1 = 1

3. Combining • with nothing, gives •.
The binary equivalent is 1 + 0 = 1

4. When we combine • with •, we get • •.
The binary representation of the above is 1 + 1 = 10

It should be noted that the above sum is not 'ten' but 'one-zero' *i.e.* it represents • • and not • • • • • • • • • •. In other words, it is 10_2 which represents decimal 2. ***It is not decimal ten.***

The last rule is often written as 1 + 1 = 0 with a carry of 1

The above rules for binary addition can be summarized as under :

0 + 0 = 0	0 + 1 = 1		
1 + 0 = 1	1 + 1 = 0	with a carry of 1	or $=10_2$

It is worth noting that 'carry-overs' are performed in the same manner as in decimal arithmetic. The rules of binary addition could also be expressed in the form of a table as shown below

	0	1
0	0	1
1	1	10_2

or

	0	1	
0	0	1	
1	1	0	with a carry 1

As an illustration, let us add 101 and 110.

$$\begin{array}{rr} & 101 \\ + & 110 \\ \hline & 1011 \end{array}$$

— first column 1 + 0 = 1
— second column 0 + 1 = 1
— third column 1 + 1 = 10

(*i.e.* 0 with carry 1)

Similarly,

$$\begin{array}{rr} & \text{1 carry} \\ & 111 \\ \hline + & 110 \\ & 1101 \end{array}$$

— 1st column 1 + 0 = 1
— 2nd column 1 + 1 = 0 with carry 1
— 3rd column 1 + 1 + carry of 1
$= 10 + 1 = 11_2$

Let us consider one more example :

$$\begin{array}{rr} & \text{1 carry} \\ & 1011 \\ \hline + & 1001 \\ & 0 \end{array} \qquad \begin{array}{rr} & \leftarrow \\ & \text{1 1 carry} \\ & 1011 \\ \hline + & 1001 \\ & 00 \end{array} \qquad \begin{array}{rr} & \leftarrow \\ & \text{1 1 carry} \\ & 1011 \\ \hline + & 1001 \\ & 100 \end{array} \qquad \begin{array}{rr} & \leftarrow \\ & \text{1 1 carry} \\ & 1011 \\ \hline + & 1001 \\ & 10100 \end{array}$$

Hence, we find from the above examples that the only two possible combinations with a carry are :

(a) 1 + 1 = sum of 0 with a carry of 1.
It is binary 10 *i.e.* 10_2 which equals *decimal* 2.

(b) 1 + 1 + carry of 1 = a sum of 1 with a carry of 1.
It equals binary 11 *i.e.* 11_2 or decimal 3.

Example 69.5. *Add* 110011_2 *to* 101101_2 **(Digital Electronics, Bombay Univ. April)**

Solution.

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

1. **first column :** 1 + 1 = 0 with a carry of 1. Hence, we put down zero there and carry 1 to the second column.
2. **second column :** 1 + 0 = 1. But combined with carry 1 from first column, it gives 1 + 1 = 0 and a carry of 1. Hence, we put down 0 there and carry 1 further to the third column.
3. **third column :** 0 + 1 = 1. Again, when it is combined with carry of 1 from the second column, we get 1 + 1 = 0 with a carry of 1. Hence, again, we put down 0 and carry 1 to the fourth column.
4. **fourth column :** Here, 0 + 1 = 1. When combined with carry 1 from third column, we get 1 + 1 = 0 with a carry of 1. Hence, we put down 0 there and carry 1 to the fifth column.
5. **fifth column :** Here, 1 + 0 = 1 When combined with the carry 1 from fourth column, we get 1 + 1 = 0 with a carry of 1. Hence, we put down 0 there and carry 1 to the sixth column.
6. **sixth column :** Here, it is a case of 1 + 1 + carry of 1 = 11_2 as stated earlier in (*b*) above.

69.11. Binary Subtraction

It is also performed in a manner similar to that used in decimal subtraction. Because binary system has only two digits, binary subtraction requires more borrowing operations than decimal subtraction. The four rules for binary subtraction are as under:

1. 0 – 0 = 0, **2.** 1 – 0 = 1, **3.** 1 – 1 = 0,

4. 0 – 1 = 1 with a borrow of 1 from the next column of the minuend

or 10 – 1 = 1

The last result represents • • – • – • which makes sense.

While using Rule 4, it should be borne in mind that borrow reduces the remaining minuend by 1. It means that a borrow will cause a 1 in the next column to the left in the minuend to become 0. If the next column also happens to contain 0, it is changed to a 1 and the succeeding 0s in the minuend are changed to 1s until a 1 is found which is then changed to a 0.

Example 1

Let us subtract 0101_2 from 1110_2. The various steps are explained below :

$$\begin{array}{r} \cancel{0}\ 1 \text{ borrow} \\ 1\,1\,1\,0 \\ -\,0\,1\,0\,1 \\ \hline 1 \end{array} \qquad \begin{array}{r} \cancel{0}\ 1 \\ 1\,1\,1\,0 \\ -\,0\,1\,0\,1 \\ \hline 0\,1 \end{array} \qquad \begin{array}{r} \cancel{0}\ 1 \\ 1\,1\,1\,0 \\ -\,0\,1\,0\,1 \\ \hline 0\,0\,1 \end{array} \qquad \begin{array}{r} \cancel{0}\ 1 \\ 1\,1\,1\,0 \\ -\,0\,1\,0\,1 \\ \hline 1\,0\,0\,1 \end{array}$$

Explanation

1. In the first column, since we cannot subtract 1 from 0, we borrow 1 from the next column to the **left**. Hence, we put down 1 in the answer and change the 1 of the next left column to a 0.
2. We apply Rule 1 to next column *i.e.* 0 – 0 = 0
3. We apply Rule 3 to the 3rd column *i.e.* 1 – 1 = 0
4. Finally, we apply Rule 2 to the last *i.e.* fourth column it. 1 – 0 = 1

As a check, it may be noted that talking in terms of decimal numbers, we have subtracted 5 from 14. Obviously, the answer has to be 9 (1001_2).

Example 2.

Let us now try subtracting 0001_2 from 1000_2.

Step 1	*Step 2*	*Step 3*	*Step 4*
1	1 1	01 1	01 1
1 0 $\cancel{0}$ 0	1 $\cancel{0}$ $\cancel{0}$ 1	$\cancel{1}$ $\cancel{0}$ $\cancel{0}$ 1	$\cancel{1}$ $\cancel{0}$ $\cancel{0}$ 1
– 0 0 0 1	– 0 0 0 1	– 0 0 0 1	– 0 0 0 1
1	1	1	0 1 1 1

Since there happened to be a 0 in the second column, it was changed to 1. Again, there was a 0 in the third column, so it was also changed to a 1. Finally, we met a 1 in the forth column which was changed to a 0 and the final answer was written down as shown above.

Example 69.6. *Subtract 0111_2 from 1001_2* **(Digital Computations, Punjab Univ. 1991)**

Solution.

1 0 0 1	1st column	:	1 – 1 = 0
– 0 1 1 1	2nd column	:	0 – 1 = 1 with a borrow of 1
0 0 1 0	3rd column	:	1 (after borrow) – 1 = 0
	4th column	:	0 (after borrow) – 0 = 0

Example 69.7. *Subtract 01011_2 from 10110_2* **(Computer Technology, Pune Univ.)**

Solution. Step-wise the solution is as under :

Step 1	*Step 2*	*Step 3*	*Step 4*	*Step 5*
0	0 0	0 0	0 0 0	0 0 0
1 0 1 $\cancel{1}$ 0	1 0 $\cancel{1}$ $\cancel{1}$ 0	1 0 $\cancel{1}$ $\cancel{1}$ 0	$\cancel{1}$ 0 $\cancel{1}$ $\cancel{1}$ 0	$\cancel{1}$ 0 $\cancel{1}$ $\cancel{1}$ 0
– 0 1 0 1 1	– 0 1 0 1 1	– 0 1 0 1 1	– 0 1 0 1 1	– 0 1 0 1 1
1	1 1	0 1 1	1 0 1 1	0 1 0 1 1

69.12. Complement of a Number

In digital work, two types of complements of a binary number are used for ***complemental subtraction*** **:**

(*a*) 1's complement

The 1's complement of a binary number is obtained by changing its each 0 into a 1 and each 1 into a 0. It is also called ***radix-minus-one complement.*** For example, 1's complement of 100_2 is 011_2 and of 1110_2 is 0001_2.

(*b*) 2's complement

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

2's complement = 1's complement + 1

It is also known **as true** complement. Suppose we are asked to find 2's complement of 1011_2. Its 1's complement is 0100_2. Next, add 1 to get 0101_2. Hence, 2's complement of 1011_2 is 0101_2.

The complement method of subtraction reduces **subtraction to an addition process.**

This method is popular in digital computers because

1. only adder ciruits are needed thus simplifying the circuitry,

2. it is easy with digital circuits to get the complements.

69.13. 1's Complemental Subtraction

In this method, instead of subtracting a number, we add its 1's complement to the minuend. The last carry (whether 0 or 1) is then added to get the final answer. The rules for subtraction by 1's complement are as under :

1. compute the 1's complement of the subtrahend by changing all its 1s to 0s and all its 0s to 1s.
2. add this complement to the minuend
3. perform the end-around carry of the last 1 or 0
4. if there is no end-around carry (*i.e.* 0 carry), then the answer must be recomplemented and a negative sign attached to it.
5. if the end-around carry is 1, no recomplementing is necessary.

Suppose we want to subtract 101_2 from 111_2. The procedure is as under :

```
   1 1 1
 + 0 1 0      ← 1's complement of subtrahend 101
 -------
 1 0 0 1
       1      ← end-round carry
 -------
   0 1 0
```

As seen, we have removed from the addition sum the 1 carry in the last position and added it onto the remainder. It is called ***end-around carry.***

Let us now subtract 1101_2 from 1010_2.

```
   1 0 1 0
 + 0 0 1 0      ← 1's complement of 1101
 ---------
   1 1 0 0
     NO CARRY
```

As seen, there is no end-around carry in this case. Hence, as per Rule 4 given above, answer must be recomplemented to get 0011 and a negative sign attached to it. Therefore, the final answer becomes—0011.

Finally, consider the complemental subtraction of 1110_2 from 0110_2.

```
     0 1 1 0
 +   0 0 0 1      ← 1's complement of 1110₂
 -----------
     0 1 1 1
     NO CARRY
```

As seen, there is no carry. However, we may add an extra 0 from our side to make it a 0 carry as shown below.

```
     0 1 1 0
 +   0 0 0 1      ← 1's complement of subtrahend
 -----------
   0 0 1 1 1
           0      ← end-around carry
 -----------
   0 0 1 1 1
```

After recomplementing, it becomes 1000. When negative sign is attached, the final answer becomes -1000_2.

Example 69.8. *Using 1's complemental method, subtract 01101_2 from 11011_2.*

Solution.

```
     1 1 0 1 1
 +   1 0 0 1 0      ← 1's complement of subtrahend
 -------------
   1 0 1 1 0 1
             1      ← end-around carry
 -------------
     0 1 1 1 0
```

Since end-around carry is 1, we take the final answer as it is (Rule 5).

Example 69.9. *Use 1's complement to subtract 11011_2 from 01101_2.*

(Computer Science, Allahabad Univ.)

Solution.

$$\begin{array}{rl} 0\,1\,1\,0\,1 & \\ +\,0\,0\,1\,0\,0 & \leftarrow \text{1's complement of } 11011_2 \\ \hline 1\,0\,0\,0\,1 & \rightarrow -01110 \end{array}$$

NO CARRY

Since there is no final carry, we recomplement the answer and attach a minus sign to get the final answer—01110_2.

69.14. 2's Complemental Subtraction

In this case, the procedure is as under :

1. find the 2's complement of the subtrahend, **2.** add this complement to the minuend,

3. drop the final carry,

4. if the carry is 1, the answer is positive and needs no recomplementing,

5. if there is no carry, recomplement the answer and attach minus sign.

Example 69.10. *Using 2's complement, subtract 1010_2 from 1101_2.*

Solution. The 1's complement of 1010 is 0101. The 2's complement is 0101 + 1 = 0110. We will add it to 1101.

$$\begin{array}{rl} 1\,1\,0\,1 & \\ +\;0\,1\,1\,0 & \leftarrow \text{2's complement of } 1010_2 \\ \hline 1\,0\,0\,1\,1 & \end{array}$$

DROP

The final answer is 0011_2.

Example 69.11. *Use 2's complement to subtract 1101_2 from 1010_2.*

(Digital Computations, Punjab Univ. 1992)

Solution. The 1's complement of 1101 is 0010. The 2's complement is 0011.

$$\begin{array}{rl} 1\,0\,1\,0 & \\ +\;0\,0\,1\,1 & \leftarrow \text{2's complement of } 1101_2. \\ \hline 1\,1\,0\,1 & \end{array}$$

NO CARRY

In this case, there is no carry. Hence, we have to recomplement the answer. For this purpose, we first subtract 1 from it to get 1100. Next, we recomplement it to get 0011. After attaching the minus sign, the final answer becomes -0011_2.

Talking in terms of decimal numbers, we have subtracted 13 from 10. Obviously the answer is –3.

69.15. Binary Multiplication

The procedure for this multiplication is the same as for decimal multiplication though it is comparatively much easier. The four simple rules are as under:

1. $0 \times 0 = 0$, **2.** $0 \times 1 = 0$, **3.** $1 \times 0 = 0$, **4.** $1 \times 1 = 1$.

The rules of binary multiplication could be summarized in the form of a table as shown.

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

As in the decimal system, the procedure is

1. copy the multiplicand when multiplier digit is 1 but not when it is 0
2. shift as in decimal multiplication
3. add the resulting binary numbers according to the rules of binary addition.

Example 69.12. *Multiply 111_2 by 101_2 using binary multiplication method.*
(Electronics-1, Indore Univ. 1991)

Solution.

$$\begin{array}{rl}
111 & \\
\times\,101 & \\
\hline
111 & \\
000 & \text{— shift left, no add} \\
111 & \text{— shift left and add} \\
\hline
100111 &
\end{array}$$

Example 69.13. *Multiply 1101_2 by 1100_2.* **(Digital Electronics, Bombay Univ. 1990)**

Solution.

$$\begin{array}{r}
1101 \\
\times\,1100 \\
\hline
0000 \\
0000 \\
1101 \\
1101 \\
\hline
10011100
\end{array}$$

Example 69.14. *Multiply 1111 by 0111_2.*

Solution. This example has been included for the specific purpose of explaining how to handle the addition if multiplication results in ***columns with more than two 1s.***

1. Result of the first column is 1.
2. In the second column, addition of 1 + 1 = 10_2. Hence, we put down 0 there and carry 1 to the third column.
3. In the third column, 1 + 1 + 1 + 1 = 100_2 (decimal 4). We keep one 0 there, put the second 0 in fourth column and pass on 1 to the fifth column.
4. In the fourth column, 1 + 1 + 1 + 0 = 11, (decimal 3). Hence, one 1 is kept there and the other 1 is passed on to the fifth column.
5. In the fifth column, 1 + 1 + 1 + 1 = 100_2 (decimal 4). Again, one 0 is retained there, second 0 is passed on to the sixth column and 1 to the seventh column.
6. In the sixth column, 1 + 0 = 1.
7. The seventh column already has 1 given by the addition of the fifth column.

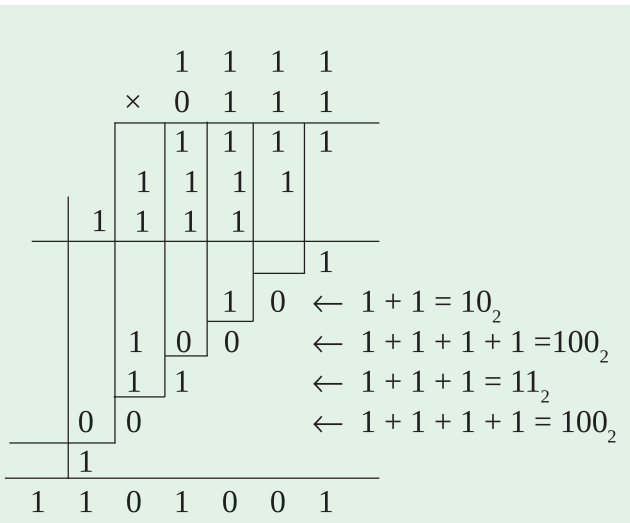


Fig. 69.5

69.16. Binary Division

It is similar to the division in the decimal system. As in that system, here also division by 0 is meaningless. Rules are :

1. $0 \div 1 = 0$ or $\frac{0}{1} = 0$, **2.** $1 \div 1 = 1$ or $\frac{1}{1} = 1$.

Example 69.15. *Carry out the binary division 11001 ÷ 101*

Solution.

```
           1 0 1
        ---------
 1 0 1 ) 1 1 0 0 1
         1 0 1
         -----
           1 0
           ------
             1 0 1
             1 0 1
             -----
             0 0 0
```

After we bring down the next 0 bit, the number 10 so formed is not divisible by 101. Hence, we put a 0 in the quotient. Therefore, the answer is 101_2 (decimal 5).

Incidentally, it may be noted that the dividend 25_{10} and divisor is 5_{10} so that the result of division, as expected, is 5_{10}.

Example 69.16. *Divide 11011_2 by 100_2* **(Digital Computations, Punjab Univ. 1990)**

Solution.

```
          1 1 0 . 1 1                        6.75
        -----------                         ------
 1 0 0 ) 1 1 0 1 1                      4 )  27
         1 0 0                                24
         -----                                --
           1 0 1                               30
           1 0 0                               28
         -------                               --
             1 1                                20
         ---------                              20
             1 1 0                              --
             1 0 0                              00
             -----
               1 0 0
               1 0 0
               -----
               0 0 0
```

69.17. Shifting a Number to Left or Right

Shifting binary numbers one step to the left or right corresponds respectively to multiplication or division by decimal 2.

When binary number 101100_2 (44_{10}) is shifted one step **to the left**, it becomes 1011000_2 which is 88_{10} *i.e.* it is doubled. If the given number is shifted one step **to the right**, it becomes 10110_2 which is 22_{10}. Obviously, the number is halved.

69.18. Representation of Binary Numbers as Electrical Signals

As seen from above, any binary number can be represented as a string of 0s and 1s. However, it is fine for paper and pencil calculations only. Practical problem is how to apply the desired binary information to logic circuits in digital computers. For that purpose, two types of electrical signals are selected to represent 1 and 0. Since speed and accuracy are of primary importance in digital circuits, the two electrical signals chosen to represent 1 and 0 must meet very rigid requirements.

1. they must be suitable for use in high-speed circuitry,
2. the signals should be very easy to tell apart,
3. they must be hard to confuse with each other.

The second and third statements may look alike but they, in fact, are not so. It is found that all transistor circuits distort, to some extent, the electrical signals that pass through them. Sometime, these distorted signals can look confusingly alike. Hence, this effect of distortion or degradation has to be kept in mind while selecting the two signals.

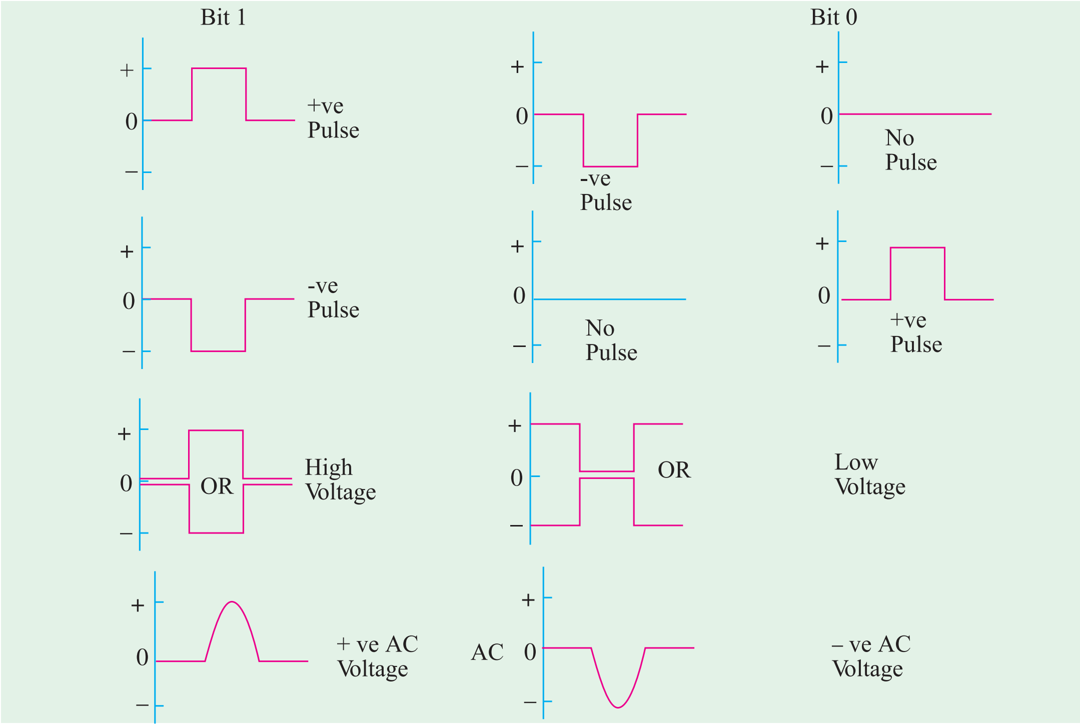


Fig. 69.6

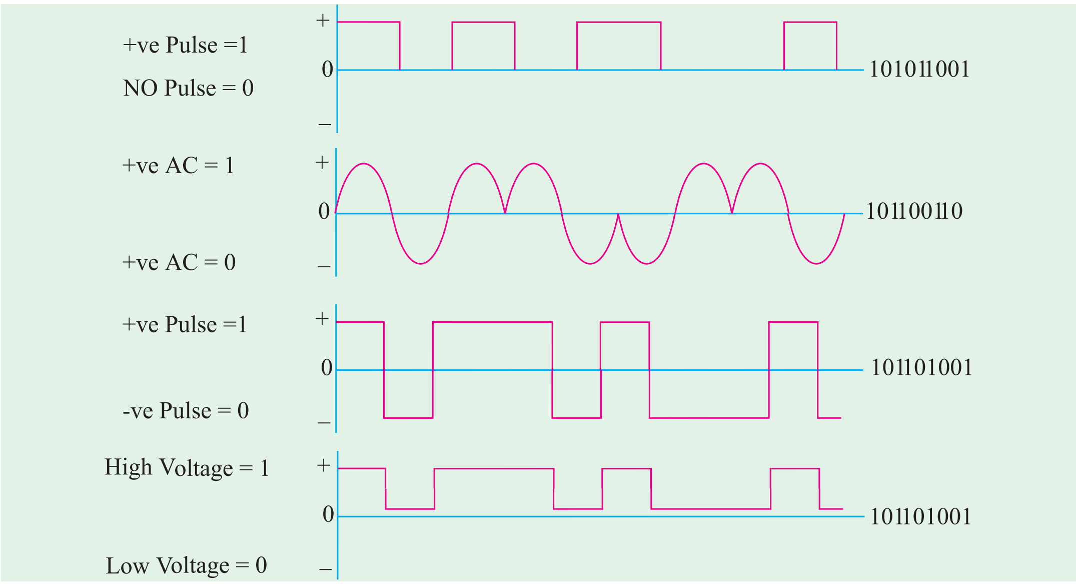


Fig. 69.7

In Fig. 69.6 are shown several signal pairs that meet the above requirements. It will be noted that it is impossible to distort a positive pulse (representing 1) to look like the no pulse or negative pulse (representing 0).

Fig. 69.7 shows how signal pairs can be used to represent different binary numbers.

69.19. Octal Number System

(*i*) Radix or Base

It has a base of 8 which means that it has eight distinct counting digits :

0, 1, 2, 3, 4, 5, 6, and 7

These digits 0 through 7, have exactly the same physical meaning as in decimal system.

For counting beyond 7, **2-digit combinations are formed taking the second digit followed by the first, then the second followed by the second and so on.** Hence, after 7, the next octal number is 10 (second digit followed by first), then 11 (second digit followed by second) and so on. Hence, different octal numbers are :

0,	1,	2,	3,	4,	5,	6,	7,
10,	11,	12,	13,	14,	15,	16,	17,
20,	21,	22,	23,	24,	25,	26,	27,
30,	31,	32,	...	...	...	...	...

(*ii*) Position Value

The position value (or weight) for each digit is given by different powers of 8 as shown below:

$$\leftarrow \quad 8^3 \quad 8^2 \quad 8^1 \quad 8^0 \quad \underset{\substack{\uparrow \\ \text{octal point}}}{\bullet} \quad 8^{-1} \quad 8^{-2} \quad 8^{-3} \quad \rightarrow$$

For example, decimal equivalent of octal 352 is

$$\begin{array}{cccccl} 3 & 5 & 2 & \bullet & 0 & \\ 8^2 & 8^1 & 8^0 & & & \\ 64 & 8 & 1 & & & = 3 \times 64 + 5 \times 8 + 2 \times 1 = 234_{10} \end{array}$$

or $$352_8 = 3 \times 8^2 + 5 \times 8^1 + 2 \times 8^0 = 192 + 40 + 2 = 234_{10}$$

Similarly, decimal equivalent of octal 127.24 is

$$\begin{aligned} 127.24_8 &= 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 2 \times 8^{-1} + 4 \times 8^{-2} \\ &= 64 + 16 + 7 + \frac{2}{8} + \frac{4}{64} = 87.3125_{10} \end{aligned}$$

69.20. Octal to Decimal Conversion

Procedure is exactly the same as given in Art. 68.4 except that we will use digit of 8 rather than 2.

Suppose, we want to convert octal 206.104_8 into its decimal equivalent number. The procedure is as under :

$$\begin{array}{ccccccc} 2 & 0 & 6 & \bullet & 1 & 0 & 4 \\ 8^2 & 8^1 & 8^0 & & 8^{-1} & 8^{-2} & 8^{-3} \end{array}$$

$$\therefore \quad 206.104_8 = 2 \times 8^2 + 6 \times 8^0 + \frac{1}{8} + \frac{4}{8^3} = 128 + 6 + \frac{1}{8} + \frac{1}{128} = \left(134 \frac{17}{128}\right)_{10}$$

69.21. Decimal to Octal Conversion

The **double-dabble** method (Art 69.7) is used with 8 acting as the **multiplying** factor for **integers** and the **divid**ing factor for **fractions.**

Let us see how we can convert 175_{10} into its octal equivalent.

$175 \div 8 = 21$ with 7 remainder
$21 \div 8 = 2$ with 5 remainder
$2 \div 8 = 0$ with 2 remainder

Taking the remainders in the **reverse order**, we get 257_8. $\therefore$ $175_{10} = 257_8$

Let us now take decimal fraction 0.15. Its octal equivalent can be found as under:

$0.15 \times 8 = 1.20 = 0.20$ with a carry of 1
$0.20 \times 8 = 1.60 = 0.60$ with a carry of 1
$0.60 \times 8 = 4.80 = 0.80$ with a carry of 4

$\therefore$ $0.15_{10} \cong 114_8$

As seen, here carries have been taken in the **forward direction** *i.e.* from top to bottom.

Using positional notation, the first few octal numbers and their decimal equivalents are shown in Table 69.2.

Table No. 69.2

Octal	*Decimal*	*Octal*	*Decimal*	*Octal*	*Decimal*
0	0	12	10	24	20
1	1	13	11	25	21
2	2	14	12	26	22
3	3	15	13	27	23
4	4	16	14	30	24
5	5	17	15	31	25
6	6	20	16	32	26
7	7	21	17	33	27
10	8	22	18	34	28
11	9	23	19	35	29

69.22. Binary to Octal Conversion

The simplest procedure is to use **binary-triplet** method. In this method, the given binary number is arranged into groups of 3 bits starting from the octal point and then each group is converted to its equivalent octal number. Of course, where necessary, extra 0s can be added **in front** (*i.e.* left end) of the binary number to complete groups of three.

Suppose, we want to convert 101011_2 into its octal equivalent. Converting the bits into groups of three, we have $\therefore$

101 011

Now, 101_2 is 5 octal and 011 is 3 octal.

$$\begin{array}{cc} 101 & 011 \\ \downarrow & \downarrow \\ 5 & 3 \end{array} \qquad \therefore \quad 101\ 011_2 = 53_8$$

Now, take 111110111_2. We will first split it into groups of **three bits** (space is left between the groups for easy reading). Then, each group is given its octal number as shown below.

$$\begin{array}{ccc} 111 & 110 & 111 \\ \downarrow & \downarrow & \downarrow \\ 7 & 6 & 7 \end{array} \qquad \therefore \quad 111\ 110\ 111_2 = 767_8$$

Finally, take the example of a mixed binary number 10101.11_2. Here, we will have to add one 0 in front of the integral part as well as to the fractional part

$$\begin{array}{ccc} 010 & 101 & 110 \\ \downarrow & \downarrow & \downarrow \\ 2 & 5 & 6 \end{array} \qquad \therefore \quad 10101.11_2 = 25.6_8$$

The equivalence between binary triplets and octal numbers is given in Table 69.3.

Table No. 69.3

Binary	*Octal*	*Binary*	*Octal*
000	0	1010	12
001	1	1011	13
010	2	1100	14
011	3	1101	15
100	4	1110	16
101	5	1111	17
110	6	10000	20
111	7	10001	21
1000	10	10010	22
1001	11	10011	23

69.23. Octal to Binary Conversion

The procedure for this conversion is just the opposite of that given in Art 69.22. Here, each digit of the given octal number is converted into its equivalent binary triplet. For example, to change 75_8 into its binary equivalent, proceed as under:

$$\begin{array}{cc} 7 & 5 \\ \downarrow & \downarrow \\ 111 & 101 \end{array} \qquad \therefore \quad 75_8 = 111\ 101_2$$

Similarly, 74.562_8 can be converted into binary equivalent as under:

$$\begin{array}{cccccc} 7 & 4 & \cdot & 5 & 6 & 2 \\ \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow \\ 111 & 100 & & 101 & 110 & 010 \end{array} \qquad \therefore \quad 74.562_8 = 111\ 100.101\ 110\ 010_2$$

Incidentally, it may be noted that number of digits in octal numbers is **one-third of that in equivalent binary numbers**. In the present case, it is five versus fifteen.

69.24. Usefulness of Octal Number System

We have already discussed the octal number system and conversion from the binary and decimal numbers to octal and vice versa. The ease with which conversions can be made between octal and binary makes the octal system attractive as a "shorthand" means of expressing large binary numbers, In computer work, binary number with up to 64 bits are not uncommon. These binary numbers, as we shall see, do not always represent a numerical quantity but are often some type of code that conveys non numerical information. In computers, binary numbers might represent :

1. actual numerical data
2. numbers corresponding to a location called (address) in memory,
3. an instruction code

4. a code representing alphabetic and other non numerical characters,

5. group of bits representing the status of devices internal or external to the computer

When dealing with a large quantity of binary numbers of many bits, it is convenient and more efficient for us to write the numbers in octal rather than binary. However keep in mind that the digital circuits and systems work strictly in binary. We use octal numbers only as a convenience for the operators of the system.

69.25. Hexadecimal Number System

The characteristics of this system are as under :

1. it has a base of 16. Hence, it uses sixteen distinct counting digits 0 through 9 and A through F as detailed below :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

2. place value (or weight) for each digit is in **ascending powers** of 16 for integers and **descending powers** of 16 for fractions.

The chief use of this system is in connection with **byte-organised machines.** It is used for specifying addresses of different binary numbers stored in computer memory.

69.26. How to Count Beyond F in Hex Number System ?

As usual, we resort to **2-digit combinations.** After reaching **F**, we take the second digit followed by the first digit, then second followed by second, then second followed by third and so on. The first few 'hex' numbers and their decimal equivalents are given in Table 69.4.

Table No. 69.4

Hexadecimal	*Decimal*	*Hexadecimal*	*Decimal*	*Hexadecimal*	*Decimal*
0	0	B	11	16	22
1	1	C	12	17	23
2	2	D	13	18	24
3	3	E	14	19	25
4	4	F	15	1A	26
5	5	10	16	1B	27
6	6	11	17	1C	28
7	7	12	18	1D	29
8	8	13	19	1E	30
9	9	14	20	1F	31
A	10	15	21	20	32
				21	33

69.27. Binary to Hexadecimal Conversion

The simple method is to split the given binary number into **4-bit groups** (supplying 0s from our own side if necessary) and then give each group its 'hex' value as found from Table 69.5.

Table No. 69.5

Binary	*Hex.*	*Binary*	*Hex.*	*Binary*	*Hex.*
0000	0	0110	6	1100	C
0001	1	0111	7	1101	D
0010	2	1000	8	1110	E
0011	3	1001	9	1111	F
0100	4	1010	A	10000	10
0101	5	1011	B	10001	11

Let us see how we would convert $10001\,100_2$ into its hexadecimal equivalent number. We will first split the given binary number into 4-bit groups and then give each group its proper value from Table 69.5.

$$\begin{array}{cc} 1000 & 1100 \\ \downarrow & \downarrow \\ 8 & C \end{array} \qquad \therefore \qquad 1000\,1100_2 = 8C_{16}$$

Let us now consider 1011010111_2. Following the above procedure, we have

$$\begin{array}{ccc} 0010 & 1101 & 0111 \\ \downarrow & \downarrow & \downarrow \\ 2 & D & 7 \end{array} \qquad \therefore \qquad 1011010111_2 = 2D7_{16}$$

It is seen that two 0s have been added to complete the 4-bit groups.

69.28. Hexadecimal to Binary Conversion

Here, the procedure is just the reverse of that given in Art. 69.27. Each hexadecimal digit is converted into its equivalent 4-bit binary.

Suppose, we want to convert $23A_{16}$ into its binary equivalent. It can be done as given below:

$$\begin{array}{ccc} 2 & 3 & A \\ \downarrow & \downarrow & \downarrow \\ 0010 & 0011 & 1010 \end{array} \qquad \therefore \qquad 23A_{16} = 0010\,0011\,1010_2$$

Incidentally, hex numbers contain **one-fourth** the number of bits contained in the equivalent binary number. Optionally, we could drop off the two 0s in front of the binary equivalent.

69.29. Decimal to Hexadecimal Conversion

Two methods are available for such a conversion. One is to go from decimal to binary and then to hexadecimal. The other method called **hex dabble** method is similar to the double-dabble (or divide-by-two) method of Art 69.7 except that we use 16 (instead of 2) for successive divisions. As an example let us convert decimal 1983 into hexadecimal by consulting Table No. 69.4 for remainders.

$$\text{Hence, } 1983_{10} = 7BF_{16} \qquad \begin{array}{rcll} 1983 \div 16 & = & 123 + 15 & \rightarrow \; F \\ 123 \div 16 & = & 7 + 11 & \rightarrow \; B \\ 7 \div 16 & = & 0 + 7 & \rightarrow \; 7 \end{array} \uparrow$$

69.30. Hexadecimal to Decimal Conversion

Two methods are available for such a conversion. One is to convert from hexadecimal to binary and then to decimal. The other direct method is as follows :

Instead of using powers of 2, use power of 16 for the weights. Then, sum up the products of hexadecimal digits and their weights to get the decimal equivalent. As an example, let us convert F6D9 to decimal.

$$\begin{aligned} F6D9 &= F(16^3) + 6(16^2) + D(16^1) + 9(16^0) = 15 \times 16^3 + 6 \times 16^2 + 13 \times 16^1 + 9 \times 16^0 \\ &= 61{,}440 + 1536 + 208 + 9 = 63{,}193_{10} \end{aligned}$$

Example 69.17. *Find the binary, octal and hexadecimal equivalents of the following decimal numbers (i) 32 (ii) 256 (iii) 51.* **(Digital Computations, Punjab Univ. May 1990)**

Solution. (*i*) Decimal number 32

As seen from Art : 10–7

$$\begin{array}{cccccc} 32 & 16 & 8 & 4 & 2 & 1 \\ 1 & Ø & Ø & Ø & Ø & Ø \end{array} \qquad \therefore \quad 32_{10} = 100000_2$$

For octal conversion :

$$\begin{array}{cc} 100 & 000 \\ \downarrow & \downarrow \\ 4 & 0 \end{array} \qquad \therefore \quad 32_{10} = 40_8$$

For hexadecimal conversion

$$\begin{array}{cc} 0010 & 0000 \\ \downarrow & \downarrow \\ 2 & 0 \end{array} \qquad \therefore \quad 32_{10} = 20_{16}$$

As will be seen, the binary number has been divided into two 4-bit groups for which purpose two 0s have been added to the left.

(ii) In the same way, it can be found that $256_{10} = 100000000_8 = 100_{16}$

(iii) Also $51_{10} = 110011_2 = 63_8 = 33_{16}$

Example 69.18. *Convert the following numbers to decimal*

(i) $(11010)_2$ ***(ii)*** $(AB60)_{16}$ ***(iii)*** $(777)_8$

(Digital Computations, Punjab Univ. 1992)

Solution. ***(i)*** For binary to decimal conversion, we will follow the procedure given in Art. 69.4.

$$\therefore \quad 11010_2 = 16 + 8 + 2 = 26_{10} \qquad \begin{array}{ccccc} 1 & 1 & 0 & 1 & 0 \\ 16 & 8 & \cancel{4} & 2 & \cancel{1} \\ 16 & 8 & \cancel{4} & 2 & \cancel{1} \end{array}$$

(ii) Following the procedure given in Art. 69.30, we have,

$$AB60_{16} = A\,(16^3) + B\,(16^2) + 6\,(16^1) + 0\,(16^0) = 10 \times 16^3 + 11 \times 16^2 + 6 \times 16^1 + 0 \times 16^0$$
$$= 43{,}872_{10}$$

(iii) As per the procedure given in Art. 69.20, $777_8 = 7(8^2) + 7(8^1) + 7(8^0) = 511_{10}$.

Example 69.19. *A computer is transmitting the following groups of bytes (each consisting of 8-bits) to some output device. Give the equivalent octal and hexadecimal listings.*

1000	*1100*	*0011*	*1010*
0010	*1110*	*1001*	*0101*
0101	*1111*	*1011*	*0110*
0111	*1011*	*0101*	*1011*

Solution.

Binary		***Octal***			***Hexadecimal***	
1000	1100	10	001	100	1000	1100
		2	1	4	8	C
0010	1110	00	101	110	0010	1110
		0	5	6	2	E
0101	1111	01	011	111	0101	1111
		1	3	7	5	F
0111	1011	01	111	011	0111	1011
		1	7	3	7	B
0011	1010	00	111	010	0011	1010
		0	7	2	3	A
1001	0101	10	010	101	1001	0101
		2	2	5	9	5
1011	0110	10	110	110	1011	0110
		2	6	6	B	6
0101	1011	01	011	011	0101	1011
		1	3	3	5	B

Hence, the groups of given memory bytes when expressed in different number systems become as under :

Binary	*Octal*	*Hexadecimal*
10001100	214	8C
00101110	056	2E
0101111	137	5F
01111011	173	7B
00111010	072	3A
10010101	225	95
10110110	266	B6
01011011	133	5B

It is clear from above that so far as the computer operator (or programmer) is concerned, it is much easier to handle this data when expressed in octal or hexadecimal system than in the binary system. For example, it is much easier and less error-prone to write the hexadecimal 8C than the binary 10001100 or 6AF than 011010101111. Of course, when the need arises, the operator can easily convert from octal or hexadecimal to binary.

69.31. Digital Coding

In digital logic circuits, each number or piece of information is defined by an equivalent combination of binary digits. A complete group of these combinations which represents numbers, letters or symbols is called a ***digital code.***

Codes have been used for security reasons so that others may not be able to read the message even if it is intercepted. In modern digital equipment, codes are used to represent and process numerical information. The choice of a code depends on the function or purpose it has to serve. Some codes are suitable where arithmetic operations are performed whereas others have high efficiency *i.e.* they give more information using fewer bits.

In certain applications, use of one code or the other simplifies and reduces the circuitry required to process the information. By limiting the switching circuitry, reliability of the digital system is increased. Of continuing importance are other codes which allow for error detection or correction. These codes enable the computers to determine whether the information that was coded and transmitted is received correctly and, if there is an error, to correct it. Since coding itself is a detailed subject, only few of the more familiar codes will be discussed.

69.32. Binary Coded Decimal (BCD) Code

It is a binary code in which *each* decimal digit is represented by a group of four bits. Since the right-to-left weighting of the 4-bit positions is 8-4-2-1, it is also called an 8421 code. It is a weighted numerical code. As said above, here each decimal digit from 0 through 9 requires a 4-bit binary-coded number. For example, the decimal number 35 in *BCD* code is 0101. The coding of ten decimal digits is given in Table No. 69.6. Lest you think that *BCD* code is the same thing as binary numbers, consider the following.

In the binary system, ten is represented by 1010 but in *BCD* code, it is 0001 0000. Seventeen in binary is 10001 but in *BCD* code, it is 0001 0111. See the difference ! Actually, the confu-

Table No. 69.6

Decimal	*BCD*
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

sion is due to the fact that the first nine numbers in *BCD* and binary are exactly similar (Table No. 69.6). After that, they become quite different (Table No. 69.7).

It should be realized that with four-bits, sixteen num bers (2^4) can be represented although in the *BCD* code only ten of these are used. The following six combinations are *invalid* in the *BCD* code : 1010, 1011, 1100, 1101, 1110 and 1111.

Table No. 69.7

Decimal	*BCD*			
26			0010	0110
59			0101	1001
673		0110	0111	0011
2498	0010	0100	1001	1000

The main advantage of *BCD* code is that it can be read and recognised easily although special adders are needed for arithmetic operations.

Any decimal number can be expressed in *BCD* code by replacing each decimal digit by the appropriate 4-bit combination. Conversely, a *BCD* number can be easily converted into a decimal number by dividing the coded number into groups of four bits (starting with *LSB*) and then writing down the decimal digit represented by each four-bit group.

Example 69.20. *Write the decimal number 369 in BCD code.*

Solution. For writing this 3-digit number in *BCD*, the value of each digit must be replaced by its 4-bit equivalent from the *BCD* code. From Table No. 69.6, we get

3 = 0011, 6 = 0110, 9 = 1001

$\therefore \quad 369_{10} = 001101101001_{BCD}$

Example 69.21. *Typically digital thermometers use BCD to drive their digital displays. How many BCD bits are required to drive a 3-digit thermometer display? What 12 bits are sent to display for a temperature of 157 degrees.*

Solution. There are 12 *BCD* bits required to drive a 3-digit thermometer display because each *BCD* digit is represented by a group of four bits.

In order to display a temperature of 157 degrees, we know that we have to send 12-bits. These bits can be determined by replacing each decimal digit by its equivalent four bit binary. Thus,

$$\begin{array}{ccc} 1 & 5 & 7 \\ \downarrow & \downarrow & \downarrow \\ 0001 & 0101 & 0111 \end{array}$$

$\therefore \quad 157_{10} = 000101010111_{BCD}$.

Example 69.22. *Find the equivalent decimal value for the BCD code number 0001010001110101.*
(Applied Electronics, A.M.I.E.E., London)

Solution. Starting from the *LSB*, the given number can be divided into groups of four bits as 0001 0100 0111 0101. As seen from Table No. 69.6,

0001 = 1, 0100 = 4,
0111 = 7 and 0101 = 5

Hence, $0001010001110101_{BCD} = 1475_{10}$

Example 69.23. **(*a*)** *Convert the hexadecimal number F8E6 to the corresponding decimal number.*

(*b*) *Convert the decimal number 2479 to the corresponding hexadecimal number.*

(*c*) *Encode the following decimal numbers into 8421 BCD numbers* ***(i)*** *59* ***(ii)*** *39 and* ***(iii)*** *584.*

(*d*) *Decode the following 8421 BCD numbers* ***(i)*** *0101* ***(ii)*** *0111.*

(Digital Computations, Punjab Univ. 1990)

Solution. (*a*)

$$\text{F8E6} = F(16^3) + 8(16^2) + E(16^1) + 6(16^0)$$
$$= 15 \times 16^3 + 8 \times 16^2 + 14 \times 16^1 + 6 \times 16^0 = \mathbf{63{,}71810}$$

(*b*) We would use the hex-dabble method explained in Art 69.29

$$\begin{array}{lllllll} 2479 \div 16 & = & 154 & + & 15 & \rightarrow & F \\ 154 \div 16 & = & 9 & + & 10 & \rightarrow & A \\ 9 \div 16 & = & 0 & + & 9 & \rightarrow & 9 \end{array}$$

$\therefore$ $2479_{10} = 9AF_{16}$

(*c*) As seen from Table No. 69.6

$$\begin{array}{ccccccc} 5 & 9 & 3 & 9 & 5 & 8 & 4 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 0101 & 1001 & 0011 & 1001 & 0101 & 1000 & 0100 \end{array}$$

(*d*) Again, consulting Table No. 69.6, we have

$$\begin{array}{cc} 0101 & 0111 \\ \downarrow & \downarrow \\ 5 & 7 \end{array}$$

69.33. Octal Coding

It involves grouping the bits in *three's*. For example, $(1756)_8 = (001\ \ 111\ \ 101\ \ 110)_2 = (001111101110)_2$. Similarly, the 24-bit number stored in the computer memory such as 101 010 011 100 010 111 000 110 can be read in the octal as

$$\begin{array}{cccccccc} 101 & 010 & 011 & 100 & 010 & 111 & 000 & 110 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 5 & 2 & 3 & 4 & 2 & 7 & 0 & 6 \end{array}$$

Apart from ease of recognition and conversion to binary, one important feature of the octal code is that its numbers are straight binary numbers which can be manipulated mathematically. For example, octal 25 expressed in octal code is 010 101 which can be read as binary 010101

$$(25)_8 = 2 \times 8^1 + 5 \times 8^0\ (21)_{10}$$
$$(010101)_2 = 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (21)_{10}$$

You might recall that in *BCD* code, the resulting number is always a 4-bit group and a special adder is needed to convert it into decimal. In octal coding, 3-bit grouping is used but the resulting binary number can be considered a single number in natural binary form.

69.34. Hexadecimal Coding

The advantage of this coding is that four bits are expressed by a single character. However, the disadvantage is that new symbols have to be used to represent the values from 1010 to 1111 binary. As seen from Table No. 69.5, the binary number 1010 0101 is hex number A5. Similarly, hexadecimal number *C*7 is $(11000111)_2$. To prove that the resulting binary number is the same as the hexadecimal value, consider the following example:

$$(3D)_{16} = 3 \times 16^1 + D \times 16^0 = 3 \times 16 + 13 \times 1 = (61)_{10}$$
$$(3D)_{16} = (00111101)_2 = 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (61)_{10}$$

Table No. 69.8

Decimal	*XS–3*
3	0110
26	0101 1001
629	1001 0101 1100
3274	0110 0101 1010 0111

69.35. Excess-3 Code

It is an unweighted code and is a modified form of *BCD*. It is widely used to represent numerical data in digital equipment. It is abbreviated as XS-3. As its name implies, each coded number in XS-3 is ***three larger*** than in *BCD* code. For example, six is written as 1001. As compared to *BCD*, the XS-3 has poorer recognition but it is more desirable for arithmetic operations. A few numbers using Excess-3 code are given in Table 69.8.

69.36. Gray Code

It is an unweighted code for numbers 0 through 9 and is largely used in mechanical switching systems. As seen from Table No. 69.9, only a single bit changes between each successive word. Because of this, the amount of switching is minimized and the reliability of the switching system is improved.

69.37. Excess-3 Gray Code

It is shown in Table No. 69.9 and is the original gray code shifted by three binary combinations. It exhibits the same properties as the Gray Code.

Table No. 69.9

Decimal	*Gray*	*XS-3 Gray*
0	0000	0010
1	0001	0110
2	0011	0111
3	0010	0100
4	0110	0100
5	0111	1100
6	0101	1101
7	0100	1111
8	1100	1110
9	1101	1010

Example 69.24. *Express the number 43_{10} in XS-3 code.*

Solution. Let us first represent each decimal digit by its 4-bit XS-3 code.

$$4 = 0111, \qquad 3 = 0110 \qquad \therefore \quad 43_{10} = 01110110_{\,XS-3}$$

Example 69.25. *The number 0110 1001 is expressed in XS-3 code. What is its decimal value ?*

Solution. Starting from least significant bit (*LSB*), the given number is first separated into groups of four and then each group is replaced by its equivalent value *i.e.* actual value decreased by 3.

$$0110 = 6 - 3 = 3;\ 1001 = 9 - 3 = 6$$

$$\therefore \quad 01101001_{XS-3} = (36)_{10}$$

69.38. Other Codes

Some of the other codes which are presently popular are given below:

(*a*) 4-bit codes

The different 4-bit weighted *BCD* codes for decimal numbers 0 through 9 in use are : 5421, 2*421, 7421, 7421, etc. and are tabulated below:

Table No. 69.10

Decimal	*5421*	*2*421*	*7421*	*$74\overline{21}$*
0	0000	0000	0000	0000
1	0001	0001	0001	0111
2	0010	0010	0010	0110
3	0011	0011	0011	0101
4	0100	0100	0100	0100
5	1000	1011	0101	1010
6	1001	1100	0110	1001
7	1010	1101	1000	1000
8	1011	1110	1001	1111
9	1100	1111	1010	1110

(*b*) 5-bit Codes

(*i*) **2-out-of-5 codes** is an unweighted *BCD* code and allows easy error detection. It has been used in communications and telephone operation.

(*ii*) **51111 Code** is a weighted *BCD* code and is much easier to operate with electronic circuitry.

(*iii*) **Shift-counter (Johnson) Code** is an unweighted *BCD* code and because of its pattern is easily operated on with electronic circuitry.

(*c*) **7-bit Biquinary Code**— it uses a group of seven bits to represent decimal numbers and has code features which provide easy error detection and ease of operation.

(*d*) **Ring-counter Code**— it is also called 10-bit code because it uses a group of 10 bits to represent a decimal number. Though it requires as many as 10 positions, the ease of error detection with the code and of operating electronic circuits to implement the code make it quite attractive.

(*e*) **Alphanumeric Code**— In addition to numerical data, a computer must be able to handle non-numerical information used in input/output (I/O) processing. In other words, a computer should recognize codes that represent letters of the alphabet, punctuation marks, and other special characters as well as numbers. These codes are called alphanumeric codes. A complete alphanumeric code would include the 26 lowercase letters, 26 uppercase letters, 10 numeric digits, 7 punctuation marks, and anywhere from 20 to 40 other characters, such as +, –, /, #, $, “, and so on. We can say that an alphanumeric code represents all of the various characters and functions that are found on a computer keyboard. The most widely used alphanumeric code is the American standard code for Information Interchange (ASCII). Another similarly I/O-oriented code is EBCDIC (Extended Binary Coded Decimal Interchange Code).

69.39. ASCII Code

The ASCII code (Pronounced “askee) is a seven-bit code, and so it has 2^7 (=128) possible code groups. This is more than enough to represent all of the standard keyboard characters as well as control functions such as the (RETURN) and LINEFEED) functions. Table No. 69.11 shows a partial listing of the ASCII code. In addition to the binary code group for each character, the table gives the octal and hexadecimal equivalents. The complete list of the ASCII code is given in the Appendix.

Table No. 69.11

Character	*7-Bit ASCII*	*Octal*	*Hex*	*Character*	*7-Bit ASCII*	*Octal*	*Hex*
A	100 0001	101	41	Y	101 1001	131	59
B	100 0010	102	42	Z	101 1010	132	5A
C	100 0011	103	43	0	011 0000	060	30
D	100 0100	104	44	1	011 0001	061	31
E	100 0101	105	45	2	011 0010	062	32
F	100 0110	106	46	3	011 0011	063	33
G	100 0111	107	47	4	011 0100	064	34
H	100 1000	110	48	5	011 0101	065	35
I	100 1001	111	49	6	011 0110	066	36
J	100 1010	112	4A	7	011 0111	067	37
K	100 1011	113	4B	8	011 1000	070	38
L	100 1100	114	4C	9	011 1001	071	39
M	100 1101	115	4D	blank	010 0000	040	20
N	100 1110	116	4E	.	010 1110	056	2E
O	100 1111	117	4F	(	010 1000	050	28
P	101 0000	120	50	+	010 1011	053	2B
Q	101 0001	121	51	$	010 0100	044	24
R	101 0010	122	52	*	010 1010	052	2A

S	101 0011	123	53	)	010 1001	051	29
T	101 0100	124	54	—	010 1101	055	2D
U	101 0101	125	55	/	010 1111	057	2F
V	101 0110	126	56	,	010 1100	054	2C
W	101 0111	127	57	=	011 1101	075	3D
X	101 1000	130	58	<RETURN	000 1101	015	0D
				<LINEFEED>	000 1010	012	0A

Example. 69.26. *The following is a message encoded in ASCII code. What is the message ?*

1001000 1000101 1001100 1001100 1001111

Solution. Convert each seven-bit code to its equivalent hexadecimal number. The resulting values are:

48 45 4C 4C 4F

Now locate these hexadecimal values in Table No. 19.11 and determine the character represented by each. The results are:

"H E L L O".

Tutorial Problems No. 69.1

1. Find the decimal equivalents of the following binary numbers:
(*a*) 101 (*b*) 1001 (*c*) 10.011 **[(*a*) 510 (*b*) 910 (*c*) 3.037510]**

2. What are the decimal equivalents of the following binary numbers ?
(*a*) 1111 (*b*) 10100 (*c*) 11011001 (*d*) 10011001
[(*a*) 1510 (*b*) 2010 (*c*) 10910 (*d*) 15310]

3. Express the following binary numbers into their equivalent decimal numbers :
(*a*) 11.01 (*b*) 101.11 (*c*) 110.01 **[(*a*) 3.2510 (*b*) 5.7510 (*c*) 6.2510]**

4. Convert the following decimal numbers into their binary equivalents:
(*a*) 25 (*b*) 125 (*c*) 0.85
[(*a*) 110012 (*b*) 11111012 (*c*) 0.1101102]

5. What are the binary equivalents of the following decimal numbers ?
(*a*) 27 (*b*) 92 (*c*) 64
[(*a*) 110112 (*b*) 10111002 (*c*) 10000002]

6. Perform the following binary additions:
(*a*) 1011 + 1001 (*b*) 1011 + 110 (*c*) 101101 + 1101101 (*d*) 1011.01 + 1001.11
(*e*) 0.0011 + 0.1110 (*f*) 1111 + 111 + 1111
[(*a*) 101002 (*b*) 100012 (*c*) 11000112 (*d*) 101012 (*e*) 1.00012 (*f*) 11010012]

7. Add the following binary numbers 1011 and 1001
[10100](*Digital Computations, Punjab Univ. Dec*.)

8. Perform the following binary subtractions :
(*a*) 1101 – 1011 (*b*) 111 – 101 (*c*) 1000 – 11 (*d*) 101011 – 10010
[(*a*) 00102 (*b*) 0102 (*c*) 010012 (*d*) 0110012]

9. Carry out the following subtractions using binary number system:
(*a*) $64_{10} - 32_{10}$ (*b*) $128_{10} - 64_{10}$ (*c*) $93.5_{10} - 42.75_{10}$ (*d*) $7_{10} - 11_{10}$
[(*a*) 1000002 (*b*) 10000002 (*c*) 110010.112 (*d*) Ä1002]

10. Subtract the following binary numbers: 100011 – 111010
[–10111](*Digital Computations, Punjab Univ., Dec. 1984*)

11. Find the 1's complements of the following binary numbers:
(*a*) 01101 (*b*) 1101 (*c*) 1001 (*d*) 1010
[(*a*) 100102 (*b*) 00102 (*c*) 01102 (*d*) 01012]

12. What are 2's complements of the following binary numbers ?
(*a*) 1011 (*b*) 11011 (*c*) 11011.01 (*d*) 10011.11
[(*a*) 01012 (*b*) 001012 (*c*) 00100.112 (*d*) 01100.012]

13. Use the 1's and 2's complements of the following binary subtractions :
(*i*) 1111 – 1011 (*ii*) 110011 – 100101 (*ii*) 100011 – 111010
[(*i*) 0100 (*ii*) 1110 (*iii*) –10111] (*Digital Computations, Punjab Univ. Dec.*)

14. Multiply the following binary numbers:
(*a*) 1100 × 101 (*b*) 10101 × 101 (*c*) 10111 × 101 (*d*) 1110 × 111
[(*a*) 1111002 (*b*) 11010012 (*c*) 11100112 (*d*) 11000102]

15. Perform the following binary divisions :
(*a*) 11011 ÷ 100 (*b*) 1110011 ÷ 101 (*c*) 1100010 ÷ 111
[(*a*) 110.112 (*b*) 101112 (*c*) 11102]

16. Convert the following binary numbers into their octal equivalents :
(*a*) 1001 (*b*) 11011 (*c*) 10 101 111 (*d*) 1101.0110111
(*e*) 11 111 011 110 101
[(*a*) 118 (*b*) 338 (*c*) 258 (*d*) 15.3348 (*e*) 373658]

17. Convert the undergiven octal numbers into their binary equivalents:
(*a*) 13 (*b*) 11 (*c*) 713 (*d*) 3674
[(*a*) 0112 (*b*) 10012 (*c*) 111 001 0112 (*d*) 11 110 111 1002]

18. Convert the following numbers :
(*a*) 357_8 to decimal (*b*) 6421_8 to decimal (*c*) 1359_{10} to octal (*d*) 7777_{10} to octal
[(*a*) 23910 (*b*) 334510 (*c*) 25178 (*d*) 171418]

19. Convert the following real numbers to the binary numbers:
(*i*) 12.0 (*ii*) 25.0 (*iii*) 0.125
[(*i*) 1100 (*ii*) 11001 (*iii*) 0.001] (*Digital Computations, Punjab Univ.*)

20. Convert the following binary numbers into their equivalent hexadecimal numbers:
(*a*) 1101 0111 (*b*) 1010 0110 (*c*) 1001 1110 (*d*) 1100 1111
[(*a*) D716 (*b*) A616 (*c*) 9E16 (*d*) CF16]

21. Convert the following decimal numbers to binary numbers by converting them to octal, then to binary.
(*i*) 850 (*ii*) 7563
[(*a*) 1101010010 (*ii*) 1110110001011] (*Digital Computations, Punjab Univ. Dec.*)

22. Convert the following numbers to decimal
(*i*) $(11010)_2$ (*ii*) $(777)_8$
[(*i*) 26 (*ii*) 511] (*Digital Computations, Punjab Univ. June*)

23. What are the binary equivalents of the following hexadecimal numbers ?
(*a*) 9F (*b*) A2 (*c*) ED (*d*) 27
[(*a*) 1001 11112 (*b*) 1010 00102 (*c*) 1110 11012 (*d*) 0011 01112]

24. Convert the decimal number 4397 and the octal number 2735 to hexadecimal numbers.
[(*i*) 5DD (*ii*) 12D] (*Power & Digital Electronics, Punjab Univ. Dec. 1984*)

25. Convert the following decimal numbers into *BCD* code
(*a*) 18 (*ii*) 92 (*iii*) 321 and (*iv*) 4721
[(*i*) 0001 1000 (*ii*) 1001 0010 (*iii*) 0011 0010 0001 (*iv*) 0100011100100001]

26. Find the decimal values for the BCD-coded numbers
(*i*) 0110 1000 (*ii*) 0111 0100 1001 and (*iii*) 1000 0100 0111 0110]
[(*i*) 68 (*ii*) 749 (*iii*) 8476]

27. Convert the following hexadecimal numbers into binary numbers:
(*i*) E 5 (*ii*) B4D (*iii*) 7AF4
(*Digital Computations, Punjab Univ. May*)
[(*i*) 11100101 (*ii*) 101101001101 (*iii*) 0111 1010 1111 0100]

28. Encode the following decimal numbers into 8421 numbers.
(*i*) 45 (*ii*) 732 (*iii*) 94,685
(*Digital Computations, Punjab Univ. May*)
[(*i*) 0100 0101 (*ii*) 111 0011 0010 (*iii*) 001 0100 0110 1000 0101]

29. Decode the following 8421 *BCD* numbers :
(*i*) 0011 1000 0111 (*ii*) 1001 0110 0111 1000 0111 0011
[(*i*) 387 (ii) 967,873] (*Digital Computations, Punjab Univ. Dec.*)

30. Decode the following 8421 *BCD* numbers :
(*i*) 0011 (*ii*) 0111 (*iii*) 0101 (*iv*) 0110
(*v*) 1000 **[(*i*) 3 (*ii*) 7 (*iii*) 5 (*iv*) 6 (*v*) 8] (*Digital Computations, Punjab Univ. June*)**

31. Convert the following binary numbers into octal code :
(*i*) 100110111 and (*ii*) 101010001 **[(*i*) 4678 (*ii*) 5218]**

32. Convert the following decimal numbers into XS-3 code:
(*i*) 35 (*ii*) 159 (*iii*) 340
[(*i*) 01101000 (*ii*) 010010001100 (*iii*) 011001110011]

33. What is the decimal value of the following numbers expressed in Excess-3 code ?
(*i*) 10010110 and (*ii*) 100101011100 **[(*i*) 63 (*ii*) 629]**

OBJECTIVE TESTS – 69

1. The digital systems usually operate on ....system.
(*a*) binary (*b*) decimal
(*c*) octal (*d*) hexadecimal.

2. The binary system uses powers of ....for positional values.
(a) 2 (*b*) 10
(*c*) 8 (*d*) 16

3. After counting 0, 1, 10, 11, the next binary number is
(*a*) 12 (*b*) 100
(*c*) 101 (*d*) 110.

4. The number 1000_2 is equivalent to decimal number
(*a*) one thousand
(*b*) eight (*c*) four
(*d*) sixteen.

5. In binary numbers, shifting the binary point one place to the right.
(*a*) multiplies by 2 (*b*) divides by 2
(*c*) decreases by 10 (*d*) increases by 10.

6. The binary addition 1 + 1 + 1 gives
(a) 111 (*b*) 10
(*c*) 110 (*d*) 11

7. The cumulative addition of the four binary bits (1 + 1 + 1 + 1) gives
(*a*) 1111 (*b*) 111
(*c*) 100 (*d*) 1001

8. The result of binary subtraction (100 – 011) is
(*a*) –111 (*b*) 111
(*c*) 011 (*d*) 001.

9. The 2's complement of 1000_2 is
(*a*) 0111 (*b*) 0101
(*c*) 1000 (*d*) 0001

10. The chief reason why digital computers use complemental subtraction is that it
(*a*) simplifies their circuitry
(*b*) is a very simple process
(*c*) can handle negative numbers easily
(*d*) avoids direct subtraction.

11. The result of binary multiplication $111_2 \times 10_2$ is
(*a*) 1101 (*b*) 0110
(*c*) 1001 (*d*) 1110

12. The binary division $11000_2 \div 100_2$ gives
(a) 110 (*b*) 1100
(*c*) 11 (*d*) 101

13. The number 12_8 is equivalent to decimal
(*a*) 12 (*b*) 20
(*c*) 10 (*d*) 4

14. The number 100101_2 is equivalent to octal
(*a*) 54 (*b*) 45
(*c*) 37 (*d*) 25.

15. The number 17_8 is equivalent to binary
(*a*) 111 (*b*) 1110
(*c*) 10000 (*d*) 1111.

16. Which of the following is NOT an octal number ?
(*a*) 19 (*b*) 77
(*c*) 15 (*d*) 101

17. Hexadecimal number system is used as a shorthand language for representing ....numbers.
(*a*) decimal (*b*) binary
(*c*) octal (*d*) large

18. The binary equivalent of A_{16} is
(*a*) 1010 (*b*) 1011
(*c*) 1000 (*d*) 1110.

19. *BCD* code is
(*a*) non-weighted
(*b*) the same thing as binary numbers
(c) a binary code
(*d*) an alphanumeric code.

20. Which of the following 4-bit combinations is/are invalid in the *BCD* code ?
(*a*) 1010 (*b*) 0010
(*c*) 0101 (*d*) 1000.

21. Octal coding involves grouping the bits in
(*a*) 5's (*b*) 7's
(*c*) 4's (*d*) 3's.

22. In Excess-3 code each coded number is ....than in BCD code.
(*a*) four larger (*b*) three smaller
(*c*) three larger (*d*) much larger.

23. Which numbering system uses numbers and letters as symbols ?
(*a*) decimal (*b*) binary
(*c*) octal (*d*) hexadecimal

24. To convert a whole decimal number into a hexadecimal equivalent, one should divide the decimal value by.......
(*a*) 2 (*b*) 8
(*c*) 10 (*d*) 16.

ANSWERS

1. (*a*) **2.** (*a*) **3.** (*b*) **4.** (*b*) **5.** (*a*) **6.** (*d*) **7.** (*b*) **8.** (*d*) **9.** (*c*) **10.** (*a*)
11. (*d*) **12.** (*a*) **13.** (*c*) **14.** (*b*) **15.** (*d*) **16.** (*a*) **17.** (*a*) **18.** (*a*) **19.** (*c*) **20.** (*a*)
21. (*d*) **22.** (*c*) **23.** (*d*) **24.** (*a*)

GO To FIRST