*Progressive Education Society's*

**Modern College of Arts, Science and Commerce, Shivajinagar, Pune – 5**

**(Autonomous College)**

**First Year of B. Sc. (2019 Course)**

**SEMESTER – I Paper -II**

**Course Code :19ScEleU102 Course Name: Principles of Digital Electronics**

**Course Contents**

| | | |
|---|---|---|
| Chapter 1 | **Number Systems** | 16 lectures |
| | Introduction to decimal, binary and hexadecimal number systems and their inter conversions, Unsigned and Signed binary number representations, Rules of binary addition and subtraction, Binary addition and subtraction, Subtraction using 1's and 2's complements, BCD code, Excess-3 code, Gray code, Alphanumeric representation in ASCII codes, Code conversion –binary to gray, gray to binary. | |
| Chapter 2 | **Logic Gates** | 7 lectures |
| | Positive and Negative Logic, OR, AND, NOT gates, NAND, NOR, EX-OR, EX-NOR gates (Symbol and truth table). | |
| Chapter 3 | **Boolean Algebra** | 12 lectures |
| | Boolean algebra and Boolean laws: Commutative, Associative, Distributive, AND, OR and Inversion laws, De Morgen's theorem, NAND, NOR as universal gate, K-map Basics, Min terms, Max terms, Boolean expression in SOP and POS form, Simplifications of Logic expressions using Boolean algebra rules and Karnaugh map (up to 4 variables), Implementation of Boolean expressions using basic gates. | |
| Chapter 4 | **Experiential Learning** | 1 lecture |
| | Group Discussion / Field Work / Mini Project. | |

**Text/ Reference Books:**

1. Digital Electronics: Jain R.P., Tata McGraw Hill
2. Digital Principles and Applications: Malvino Leach, Tata McGraw-Hill
3. Digital Fundamentals: Floyd T.M., Pearson Education

## 1. Introduction to Number Systems

After hearing the word number we immediately think of the most familiar decimal number system with its 10 digits 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. However, it is only customary and not necessary to use these symbols. Basically a number system is 'an ordered set of symbols known as digits', and the total number of symbols used in a system defines the type of number system. Some of the commonly used number systems are binary, octal and hexadecimal number system. Each number system has predefined rules for performing arithmetic operations like addition, subtraction, multiplication etc. In this chapter, we will study different number systems, their inter conversion and the different codes used in the digital system.

## 2. Decimal Number System

In decimal number system, an ordered set of ten symbols 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9 are used to specify the quantities. The symbols used are known as digits. Since deci-stands for ten, decimal number system uses ten basic symbols. The number of different basic symbols used in a system is known as *base* or *radix* of a number system. Since decimal number system has 10 basic symbols, the base or radix of this number system is 10. The radix of binary system is 2, octal system is 8 and that of hexadecimal is 16. Table below shows the representation of different quantities in decimal number system, where black dots are used to represent the quantities. We can use decimal digit 0 through 9 to represent quantities up to 9. In order to represent 10 quantities, the decimal 10 is obtained by combining the second digit followed by the first.

**Table : Decimal digits**

| Quantity | Symbol |
|---|---|
| None | 0 |
| • | 1 |
| • | 2 |
| •• | 3 |
| ••• | 4 |
| •••• | 5 |
| ••••• | 6 |
| •••••• | 7 |
| ••••••• | 8 |
| •••••••• | 9 |
| ••••••••• | 10 |
| •••••••••• | 11 |

Since decimal number system uses 10 symbols, we can express decimal number (integer or a whole number) in units, tens, hundreds and so on. For example, the decimal number 298 decomposes into,

$$298 = 200 + 90 + 8$$

In powers of 10, this becomes

$$298 = 2 \times 10^2 + 9 \times 10^1 + 8 \times 10^0$$

Thus, the decimal number can be expressed as sum of powers of ten and in above example, the digit 2 has weight 200, the digit 9 has weight 90 and the digit 8 has weight 8.

As we go from right to the left, the weight also known as place value of decimal digit increases in terms of powers of 10. However, the decimal number may have an integer and a fractional part. The integer and the fractional part are separated by a point known as the *decimal point* e.g. decimal number 52.38, where 52 is an integer part and 38 is the fractional part of the given number.

The weights assigned in case of mixed number are

$$\text{etc.} \leftarrow 10^3\ 10^2\ 10^1\ 10^0 \cdot 10^{-1}\ 10^{-2}\ 10^{-3} \rightarrow \text{etc.}$$

$$\uparrow$$

Decimal point

Hence, the number 52.38 can be expressed as,

$$\begin{aligned} 52.38 &= 5 \times 10^1 + 2 \times 10^0 + 3 \times 10^{-1} + 8 \times 10^{-2} \\ &= 50 + 2 + \frac{3}{10} + \frac{8}{100} \\ &= 50 + 2 + 0.3 + 0.08 \\ &= 52.38 \end{aligned}$$

There are some other number systems that are also used to represent quantities. Some of the commonly used number systems are binary, octal and hexadecimal number system. These systems are widely used in computers, logic circuits, microprocessors, etc.

## 3. Binary Number System

Decimal number system is not suitable for numerical operation and computation using electronic machine like computers, microprocessors etc. It is very much essential to convert decimal numbers to binary numbers that will solve this problem. The main advantage of using binary number system is that, it minimizes the number of lines in a computer system.

The binary number system is a code that uses only two basic symbols i.e. 0 and 1. Generally, in digital electronics, '0' represents the low state and '1' the high state. These symbols are known as *bits*. These states 1 and 0 are also referred as ON and OFF, HIGH or LOW, magnetized and demagnetized, white or black etc.

We can form many other number systems by selecting a different group of basic symbols or digits. The base or radix of number system refers to the number of basic symbols used. For instance, ten is the base of the decimal number system because this system uses ten digits 0 through 9. The binary number system has a base of two because the only two digits 0 and 1 are used.

All binary numbers consist of a string of 0s and 1s. Examples are 10, 101, 1001 and which are read as one zero, one zero one, one zero zero one, to avoid confusion with decimal numbers.

An easy way to remember how to write binary sequence is as follows. If we begin counting decimal numbers starting from $0_{10}$ and only the count is taken into account having only 0 and 1 then we get,

0, 1, … 10, 11, … 100, 101 … 110, 111 … 1000, 1001, … 1010, 1011, … 1100, 1101, … 1110, 1111.

The above numbers are tabulated in systematic form in the following Table below.

**Table : Binary number system**

| Decimal number | Binary number |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 10 |
| 3 | 11 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |
| 8 | 1000 |
| 9 | 1001 |
| 10 | 1010 |
| 11 | 1011 |
| 12 | 1100 |
| 13 | 1101 |
| 14 | 1110 |
| 15 | 1111 |

As we see from the table, only two symbols are available. The group of four bits is known as *nibble* and group of eight bits is known as *byte*.

e.g. 1001, 1110 etc. are nibbles and 10101011, such group is called as byte.

A binary number can be represented as,

$$N = d_n \times 2^n + d_{n-1} \times 2^{n-1} + \ldots d_1 \times 2^1 + d_0 \times 2^0 + d_{-1}\, 2^{-1} + \ldots + d_{-m}\, 2^{-m}$$

where $d_n$, $d_{n-1}$ … $d_{-(m-1)}$, $d_{-m}$ are binary symbols (0 or 1) and n, m are integers.

Like the decimal system binary system is also positional weighted. However, the position value of each bit corresponds to some power of 2. In each binary number, the value increases by power of 2 starting with 0 to left of binary point and decreases to the right of the binary point starting with power of –1. The left most bit is known as Most Significant Bit (MSB) and the right most bit is known as Least Significant Bit (LSB). MSB has highest weight while LSB has lowest weight.

The position value (or weight) of each bit along with 8 bit 1010.1001 is shown below :

| MSB | | | | | | | | LSB |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | · | 1 | 0 | 0 | 1 |
| $2^3$ | $2^2$ | $2^1$ | $2^0$ | ↓ | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ | $2^{-4}$ |
| | | | | Binary point | | | | |

As seen from the above example, fourth bit to the left of binary point carries the maximum weight (i.e. has the highest value) and is called Most Significant Bit (MSB) or most significant digit. Similarly, the fourth bit to the right of the binary point is called Least Significant Bit (LSB) or least significant digit.

The position value of different bits are given by ascending power of 2 to the left of binary point and descending power of 2 to the right of the binary point. The different digit position in general form may be stated as given below :

$\leftarrow \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \quad \bullet \quad 2^{-1} \quad 2^{-2} \quad 2^{-3} \quad 2^{-4} \quad \rightarrow$

$\downarrow$

Binary point

$\leftarrow \quad 16 \quad 8 \quad 4 \quad 2 \quad 1 \quad \quad 1/2 \quad 1/4 \quad 1/8 \quad 1/16 \quad \rightarrow$

Position value

If 2 bits are used, the highest count is up to 3. If 3 bits are used it is up to 7, and for 4 bit highest count is 15, means in general we can say,

Highest decimal count = $2^n - 1$, where n is number of bits.

If n = 5, i.e. with 5 bits we can count from 0 through 31.

$$2^5 - 1 = 32 - 1$$
$$= 31 \text{ is highest count}$$

If we prepare table for 5 bit binary number presentation, we will have the following Table below.

**Table**

| Decimal number | Binary number |
|---|---|
| 0 | 00000 |
| 1 | 00001 |
| 2 | 00010 |
| 3 | 00011 |
| 4 | 00100 |
| 5 | 00101 |
| 6 | 00110 |
| 7 | 00111 |
| 8 | 01000 |
| 9 | 01001 |
| … | … |
| … | … |
| … | … |
| … | … |
| … | … |
| … | … |
| 30 | 11110 |
| 31 | 11111 |

An easy way to remember how to write a binary sequence such as in Table above , for five bit example we have,

1. The LSB i.e. right most column binary number begins with 0 and alternates each bit.
2. The second LSB i.e. next column has two 0s and alternates every two bits.
3. The third LSB have four 0s and alternates every four bits.
4. The fourth LSB have eight 0s and alternates every eight bits.
5. The next column i.e. MSB have sixteen 0s and alternates every sixteen bits.

## 4. Binary to Decimal Conversion:

Any binary number can be converted into its equivalent decimal number using the weights assigned to each bit position. The rightmost bit (LSB) in a binary number has weight of $2^0 = 1$.The weights increase by power of two for each bit from right to left.

e.g. Binary number 111 becomes,

Binary number $\rightarrow$ 1 1 1

Binary weight $\rightarrow$ $2^2\ 2^1\ 2^0$

$= 4 + 2 + 1 = 7$

$111 = 7$

**Example 1 :** Convert the binary number 1111 to decimal.

**Solution :**

Binary number 1 1 1 1

Binary weight $2^3\ 2^2\ 2^1\ 2^0$

Weight value 8 4 2 1

$= 1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1$

$= 8 + 4 + 2 + 1$

$= (15)_D$ or $(15)_{10}$

It can be represented as

$(1111)_B = (15)_D$

or $(1111)_2 = (15)_{10}$

The subscript D or 10 identifies decimal number and subscript B or 2 identify binary number.

**Example 2 :** Convert the binary number 110101 to decimal number.

**Solution :**

Binary number 1 1 0 1 0 1

Binary weight $2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

Weight value 32 16 8 4 2 1

$= 1 \times 32 + 1 \times 16 + 0 + 1 \times 4 + 0 + 1 \times 1$

$= 32 + 16 + 4 + 1 = (53)_{10}$

We can also use streamlined method by the following procedure,

Step 1 1 1 0 1 0 1

Step 2 32 16 8 4 2 1

Step 3 32 16 ~~8~~ 4 ~~2~~ 1

In step 3 if zero appears in digit position, cross out the decimal weight for that position.

Step 4 $32 + 16 + 4 + 1 = (53)_{10}$

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

or $(110101)_B = (53)_D$

The binary numbers we have seen so far have been whole numbers. Fractional number can also be represented in a binary by placing bits to the right of binary point.

In general, binary number with fractions can be written as

$$2^n \ldots 2^4\ 2^3\ 2^2\ 2^1\ 2^0 \cdot 2^{-1}\ 2^{-2}\ 2^{-3} \ldots 2^{-n}$$

$$\downarrow$$

binary point

This indicates that all the bits to the left of binary point have weights that are positive power of two.

All bits to the right of binary point have weights that are negative power at two or fractional weight.

$\cdot 2^{-1},\ 2^{-2},\ 2^{-3}, 2^{-4}$ etc.

↓

binary point

$\cdot \frac{1}{2}\ ,\ \frac{1}{4}\ ,\ \frac{1}{8}\ ,\ \frac{1}{16}$ etc.

$\cdot 0.5,\ 0.25,\ 0.125,\ 0.0625$ etc.

↓

binary point

**Example 3 :** Determine the decimal number represented by binary fraction 0.101.

**Solution :** First determine the weight of each bit and then sum the weight times the bit.

| Binary weight | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ |
|---|---|---|---|
| Weight value | 0.5 | 0.25 | 0.125 |
| Binary number | 0.1 | 0 | 1 |

$= 1 \times 0.5 + 0 \times 0.25 + 1 \times 0.125$

$= 0.5 + 0 + 0.125$

$= 0.625$

**Example 4 :** Determine the decimal value of the binary number 0.1011.

**Solution :**

| Binary weight | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ | $2^{-4}$ |
|---|---|---|---|---|
| Weight value | 0.5 | 0.25 | 0.125 | 0.0625 |
| Binary value | 0.1 | 0 | 1 | 1 |

$= 1 \times 0.5 + 0 \times 0.25 + 1 \times 0.125 + 1 \times 0.0625$

$= 0.5 + 0 + 0.125 + 0.0625$

$= 0.5 + 0.125 + 0.0625$

$= (0.6875)_{10}$

Mixed numbers (numbers with integer and fractional part) convert each part according to the rules developed.

$1100 \cdot 1011 = 1 \times 8 + 1 \times 4 + 0 + 0 + 1 \times 0.5 + 0 + 1 \times 0.125 + 1 \times 0.0625$

$= 8 + 4 + 0.5 + 0.125 + 0.0625 = 12.6875$

$(1100 \cdot 1011)_2 = (12.6875)_{10}$

**Example 5 :** Perform the following : $(11010.010)_2 = (?)_{10}$.

**Solution :** 11010.010 ← given number

**Step :**

(i) Write the weight 16 8 4 2 1 0.5 0.125 0.0625

(ii) Discard the weight whose coefficients are zero

16 8 ~~4~~ 2 1 ~~0.5~~ 0.125 ~~0.0625~~

(iii) Add the remaining weight $16 + 8 + 2 + 0.125 = 26.125$

$\therefore \quad (11010.010)_2 = (26.125)_{10}$

## 5. Decimal to Binary Conversion:

Any decimal number can be converted into its equivalent binary number. One way to convert a decimal number into binary equivalent is reverse process of binary to decimal conversion.

e.g. 9 can be written in the power of 2 as,

$$9 = 2^3 + 2^2 + 2^1 + 2^0$$
$$= 8 + 0 + 0 + 1$$
$$\rightarrow 1001$$

Or, number 23 can be written as,

$$23 = 2^4 + 2^3 + 2^2 + 2^1 + 2^0$$
$$= 16 + 0 + 4 + 2 + 1$$
$$\rightarrow 10111$$

A popular way to convert decimal to binary numbers is *double-dabble method* or divide by two method. In this method, progressively divide the decimal number by 2 and write down the remainders after each division. These remainders are taken in reverse order i.e. from bottom to top form the required binary number.

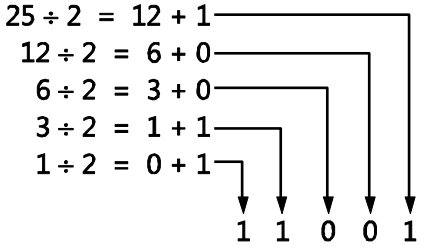
**Example 1 :** Convert the decimal number $25_{10}$ into equivalent binary number.

**Solution :**

| | Quotient | Remainder | |
|---|---|---|---|
| $\frac{25}{2}$ | 12 | 1 | LSB |
| $\frac{12}{2}$ | 6 | 0 | |
| $\frac{6}{2}$ | 3 | 0 | |
| $\frac{3}{2}$ | 1 | 1 | Read |
| $\frac{1}{2}$ | 0 | 1 | MSB |

$$(25)_{10} = (11001)_2$$

It may also be put in the following form :




$$(25)_{10} = (11001)_2$$

**Example 2 :** Convert the decimal number $45_{10}$ into equivalent binary number.

**Solution :**

| | Quotient | Remainder | |
|---|---|---|---|
| $\frac{45}{2}$ | 22 | 1 | (LSB) |
| $\frac{22}{2}$ | 11 | 0 | |
| $\frac{11}{2}$ | 5 | 1 | |
| $\frac{5}{2}$ | 2 | 1 | Read |
| $\frac{2}{2}$ | 1 | 0 | |
| $\frac{1}{2}$ | 0 | 1 | (MSB) |

$(45)_{10} = (101101)_2$

**6. Decimal Fractions to Binary conversion:**

In this case, multiply by two rule is used. We multiply bit by two and record the carry in the integer position, these are taken in the forward direction (top to bottom) i.e. top carry is MSB and bottom is LSB.

**Example 1 :** Convert a decimal fraction 0.8125 into binary equivalent.

**Solution :** Multiply the decimal fraction by 2 until the fractional product is zero.

$0.8125 \times 2 = 1.625 = 0.625$ with carry 1 MSB

$0.625 \times 2 = 1.25 = 0.25$ with carry 1

Read

$0.25 \times 2 = 0.50 = 0.5$ with carry 0

$0.5 \times 2 = 1.0 = 0$ with carry 1 LSB

Hence $(0.8125)_{10} = (0.1101)_2$

**Example 2 :** Convert decimal number 25.6 into binary number.

**Solution :** First split the decimal number into an integer 25 and fraction of 0.6 and apply double-dabble to each part.

| | Quotient | Remainder | |
|---|---|---|---|
| $\frac{25}{2}$ | 12 | 1 | LSB |
| $\frac{12}{2}$ | 6 | 0 | |
| $\frac{6}{2}$ | 3 | 0 | Read |
| $\frac{3}{2}$ | 1 | 1 | |
| $\frac{1}{2}$ | 0 | 1 | MSB |

$(25)_{10} = (11001)_2$

and

$0.6 \times 2 = 1.2 = 0.2$ with carry 1 — MSB

$0.2 \times 2 = 0.4 = 0.4$ with carry 0

$0.4 \times 2 = 0.8 = 0.8$ with carry 0 — Read

$0.8 \times 2 = 1.6 = 0.6$ with carry 1

$0.6 \times 2 = 1.2 = 0.2$ with carry 1 — LSB

$(0.6)_{10} = (10011)_2$

In the fraction conversion, we stop the conversion process after getting five binary digits because in the last step, we had got the repetition of initial number i.e. 0.2.

$$(25.6)_{10} = (11001 \cdot 10011)_2$$

**Example 3 :** Perform the following : $(28.47)_{10} = (?)_2$.

**Solution :** For mixed numbers we have to perform conversion separately for integer part and fraction part.

| | | |
|---|---|---|
| 2 | 28 | |
| | 14 | 0 |
| | 7 | 0 |
| | 3 | 1 |
| | 1 | 1 |
| | 0 | 1 |

$28_{10} = 11100_2$

$0.47 \times 2 = 0.94 = 0.94$ with carry 0

$0.94 \times 2 = 1.88 = 0.88$ with carry 1

$0.88 \times 2 = 1.76 = 0.76$ with carry 1

$0.76 \times 2 = 1.52 = 0.52$ with carry 1

$\therefore \quad (0.47)_{10} = (0.0111)_2$

$\therefore \quad (28.47)_{10} = (11100.0111)_2$

## 7. Hexadecimal number system:

Hexadecimal number system has base of sixteen. It is extensively used in microprocessor work. They are much shorter than binary numbers. This makes them easy to write and remember. Hexadecimal number represents group of four bit binary numbers.

It uses 16 distinct symbols 0 to 9 and A to F.

Thus the symbols in hexadecimal number systems are,

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

A subscript 16 or H is used which indicates a hexadecimal number.

e.g. 42H or $(42)_{16}$

9CH or $(9C)_{16}$

**Table 1.4 : Hexadecimal number system**

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

The hexadecimal number after F can be read as,

10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F,

20, 21, …… 29, 2A, 2B, 2C, 2D, 2E, 2F,

30, 31 …… so on …

With two hexadecimal digits we can count up to $FF_{16}$ which is $255_{10}$. To count beyond this, three hexadecimal digits are needed. For instance $100_{16}$ is decimal $256_{10}$, $101_{16}$ is $257_{10}$ and so on. The maximum four digit hexadecimal number is $FFFF_{16}$ which is $65535_{10}$.

## 8. Hexadecimal to Binary number conversion:

Hexadecimal number can be converted into binary number by converting each hexadecimal digit to 4 bit binary equivalent.

**Example 1 :** Convert $(9CA)_H$ to binary.

**Solution :**

$$\begin{array}{cccc} & 9 & C & A \\ & \downarrow & \downarrow & \downarrow \\ & 1001 & 1100 & 1010 \end{array}$$

$$(9CA)_H = (1001\ 1100\ 1010)_2$$

**Example 2 :** Convert $(D2E \cdot 8)_{16}$ to binary.

**Solution :**

$$\begin{array}{cccccc} & D & 2 & E & \cdot & 8 \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ & 1101 & 0010 & 1110 & \cdot & 1000 \end{array}$$

$$(D2E \cdot 8)_{16} = (1101\ 0010\ 1110 \cdot 1000)_2$$

**9. Binary to Hexadecimal number conversion:**

Binary number can be easily converted to hexadecimal by making a group of four bit starting from binary point. Then group is converted to equivalent hexadecimal.

**Example 1 :** Convert the 1100101001010111 binary number to hexadecimal.

**Solution :**

$$\begin{array}{cccc} 1100 & 1010 & 0101 & 0111 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ C & A & 5 & 7 \end{array}$$

$$(1100101001010111)_2 = (CA57)_{16}$$

**Example 2 :** Convert $(10110110110011)_2$ to hexadecimal.

**Solution :**

$$\begin{array}{cccc} 001 & 1101 & 1011 & 001 \\ 0 & & & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 2 & D & B & 3 \end{array}$$

$$(101101101100011)_2 = (2DB3)_{16}$$

**Example 3 :** Convert $11110101 \cdot 111$ to hexadecimal number.

**Solution :**

$$\begin{array}{cccccc} 1111 & 0101 & \cdot & 1110 \\ \downarrow & \downarrow & & \downarrow \\ F & 5 & \cdot & E \end{array}$$

$$(1111\ 0101 \cdot 111)_2 = (F5 \cdot E)_{16}$$

**10. Hexadecimal to Decimal number conversion:**

In the hexadecimal number system each digit position corresponds to power of 16. The weights of the digit position in a hexadecimal number are as follows :

$$16^3\ 16^2\ 16^1\ 16^0 \cdot 16^{-1}\ 16^{-2}\ 16^{-3}$$

$$\uparrow$$

Hexadecimal point

Therefore, to convert from hexadecimal to decimal, multiply each hexadecimal digit by its weight and add the resulting products.

**Example 1 :** Convert $(E5)_{16}$ to decimal number.

**Solution :**

$$\begin{aligned} (E5)_{16} &= E \times 16^1 + 5 \times 16^0 \\ &= E \times 16 + 5 \times 1 \\ &= 14 \times 16 + 5 \times 1 \\ &= 224 + 5 \\ &= 229 \\ (E5)_{16} &= (229)_{10} \end{aligned}$$

**Example 2 :** Convert $(B2F \cdot 39)_{16}$ to decimal number.

**Solution :**

$$\begin{aligned} (B2F \cdot 39)_{16} &= B \times 16^2 + 2 \times 16^1 + F \times 16^0 + 3 \times 16^{-1} + 9 \times 16^{-2} \\ &= 11 \times 256 + 2 \times 16 + 15 \times 1 + 0.1875 + 0.0352 \\ &= (2863 \cdot 2227)_{10} \end{aligned}$$

**Example 3 :** $(A2.2)_{16} = (?)_{10}$.

**Solution :** $(A2.2)_{16} = A \times 16^1 + 2 \times 16^0 + 2 \times 16^{-1}$

$= 10 \times 16 + 2 \times 1 + 2 \times \frac{1}{16}$

$= 160 + 2 + 0.125$

$= 162.125$

$\therefore \quad (A2.2)_{16} = (162.125)_{10}$

**11. Decimal to Hexadecimal Number conversion:**

The decimal number can be converted to hexadecimal number by successively dividing by 16. Then the remainder is converted to the equivalent hexadecimal. The remainder taken in reverse order, written from the bottom to the top, gives hexadecimal number.

**Example 1 :** Convert $(650)_{10}$ to hexadecimal by repeated division by 16.

**Solution :**

| Successive division | Quotient | Remainder | Hexadecimal | |
|---|---|---|---|---|
| $\frac{650}{16}$ | 40 | 10 | A | ↑ |
| $\frac{40}{16}$ | 2 | 8 | 8 | Read |
| $\frac{2}{16}$ | 0 | 2 | 2 | |

$\therefore \quad (650)_{10} = (28A)_{16}$

**Example 2 :** Convert $(423)_{10}$ to hexadecimal.

**Solution :**

| Successive division | Quotient | Remainder | Hexadecimal | |
|---|---|---|---|---|
| $\frac{423}{16}$ | 26 | 7 | 7 | ↑ |
| $\frac{26}{16}$ | 1 | 10 | A | Read |
| $\frac{1}{16}$ | 0 | 1 | 1 | |

$\therefore \quad (423)_{10} = (1A7)_{16}$

**Example 3 :** $(43.4)_{10} = (?)_{16}$.

**Solution :**

| | | | |
|---|---|---|---|
| 16 \| 43 | | | |
| 2 | 11 | → B | ↑ |
| 0 | 2 | → 2 | |

$(43)_{10} = (2B)_{16}$

$0.4 \times 16 = 6.4 = 0.4$ with carry 6 ↓

$0.4 \times 16 = 6.4 = 0.4$ with carry 6

$0.4 \times 16 = 6.4 = 0.4$ with carry 6

$\therefore \quad (43.4)_{10} = (2B.666)_{16}$

**12. Hexadecimal Fraction:**

For converting the decimal fraction to hexadecimal, multiply the decimal fraction by 16 and collect all the numbers to the left of the decimal point. Convert them into hexadecimal equivalent. Reading the number downwards gives the hexadecimal fraction.

**Example 1 :** Convert 0.357 to hexadecimal equivalent.

**Solution :** 0.357 is original number.

| | Carry | Hexadecimal | |
|---|---|---|---|
| $0.357 \times 16 = 5.712$ | 5 | 5 | |
| $0.712 \times 16 = 11.392$ | 11 | B | Read |
| $0.392 \times 16 = 6.272$ | 6 | 6 | |

Reading the hexadecimal equivalent from top to the bottom.

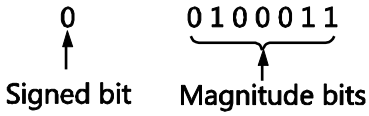
$$(0.357)_{10} = (0.5B6)_{16}$$

**13. Signed and unsigned binary number presentation:**

We know that in the decimal number system (+) sign is used to denote a positive number and a minus (–) sign for denoting a negative number. Thus, a positive integer 5 can be represented by +5 and negative integer 5 by –5. It is also customary to drop plus sign, and the absence of any sign means that the number has positive value. This representation of numbers is known as *signed numbers*. Similar to signed decimal numbers we have signed binary numbers. A signed binary number consists of both sign and magnitude information. The sign indicate whether a number is positive or negative and the magnitude is the value of the number. There are three forms in which signed integer numbers can be represented in binary, they are sign magnitude, 1's complement and 2's complement form.

**Sign Magnitude Representation :** Since, digital circuit can understand only two symbols 0 and 1, in the form of voltage levels assigned to it, therefore the same symbol is used to indicate the sign of the number also. Normally an additional bit is used as the sign bit and is placed as the most significant bit. A 0 is used to represent a positive number and a 1 to represent a negative number. Thus, when a binary number is represented in sign magnitude, the leftmost bit is the sign bit and the remaining bits are the magnitude bits.

The decimal number + 35 is expressed as an 8-bit signed binary number as




The decimal number –35 is expressed as 10100011. Thus, in sign-magnitude representation, the left most 0 (MSB) indicates that the number is positive and 1 in the left most position (MSB) indicates that the number is negative and the other seven bits give its magnitude. In sign binary representation as soon as we observe 1 in the left most position the number is viewed as negative number otherwise positive number.

**14. Rules of binary addition and subtraction:**

We are familiar with the arithmetic operations such as addition, subtraction, multiplication and division of decimal numbers. Similar operations can be performed on binary numbers in which only two digits 0 and 1 are involved.

A digital computer can perform such arithmetic operations. For this, it uses many arithmetic circuits built in it.

## 14.1 Rules for Binary Addition

There are four basic rules for adding binary digits :

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 10_2$

**Table 3.1 : Rules of binary addition**

| Augend | Addend | Sum | Carry | Result |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 10 |

In the first three rows in Table 3.1, there is no carry i.e. carry = 0, whereas in the fourth row a carry is produced i.e. carry = 1 and it is added to the next higher binary position as similar to decimal addition.

When there is a carry, we have a situation where three bits are being added (a bit in each of two numbers and a carry bit). The rules for this are as follows :

Carry bits
$1 + 0 + 0 = 01_2$ 1 with a carry of 0
$1 + 1 + 0 = 10_2$ 0 with a carry of 1
$1 + 0 + 1 = 10_2$ 0 with a carry of 1
$1 + 1 + 1 = 11_2$ 1 with a carry of 1

The subscript 2 (like $10_2$) represents a binary number (8 for octal, 10 for decimal, 16 for hexadecimal).

**Example 1 :** (a)

$$\begin{array}{r} 01_2 \\ +\,01_2 \\ \hline 10_2 \end{array} \qquad \begin{array}{r} 01_{10} \\ +\,01_{10} \\ \hline 02_{10} \end{array}$$

(b)

$$\begin{array}{r} 11_2 \\ +\,11_2 \\ \hline 110_2 \end{array} \qquad \begin{array}{r} 3_{10} \\ +\,3_{10} \\ \hline 6_{10} \end{array}$$

(c)

$$\begin{array}{r} 111_2 \\ +\;11_2 \\ \hline 1010_2 \end{array} \qquad \begin{array}{r} 7_{10} \\ +\;3_{10} \\ \hline 10_{10} \end{array}$$

(d)

$$\begin{array}{r} 1011 \\ +\;1100 \\ \hline 10111 \end{array}$$

↑
carry

(e) Add two numbers 0101 and 1111.

```
      (1)   (1)   (1)          ← carry
       0     1     0     1
   + 1       1     1     1
  ------------------------
   1  0     1     0        0
   ↑
  carry
```

For 8-bit arithmetic addition, least significant bit (LSB) is on the right and most significant bit (MSB) is on the left.

(MSB) (LSB)

$A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$

$B_7\ B_6\ B_5\ B_4\ B_3\ B_2\ B_1\ B_0$

**Example 2 :** Add two 8-bit numbers 01010111 and 00110101.

**Sol. :**

```
      (1) (1) (1) (1) (1) (1)      ← carry
       0  1  0  1  0  1  1  1
   +   0  0  1  1  0  1  0  1
  ---------------------------
       1  0  0  0  1  1  0  0
```

## 14.2 Rules for binary subtraction:

There are four basic rules for subtracting binary digits.

$0 - 0 = 0$

$1 - 1 = 0$

$1 - 0 = 1$

$10_2 - 1 = 1$

**Table 3.2 : Rules of binary subtraction**

| Minuend | Subtrahend | Difference | Borrow |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

While subtracting numbers, we sometime have to borrow from the next higher column. In Table 3.2 except in the second row, borrow = 0. When borrow = 1, as in the second row, this is to be subtracted from the next higher binary bit as it is done in decimal subtraction.

**Example 1 :** (a)

$$\begin{array}{r} 11_2 \\ -\ 10_2 \\ \hline 01_2 \end{array} \qquad \begin{array}{r} 3_{10} \\ -\ 2_{10} \\ \hline 1_{10} \end{array}$$

(b) Subtract 011 from 101.

When 1 is borrowed, 0 is left

```
      0  ──────►          Borrow 1 from next column making
    1  10  1             10 in this column (10 – 1 = 1)
  –  0   1   1
  ------------
     0   1   0           ← (1 – 1 = 0)
```

(c) Subtract the following :

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

In columns 1 and 2, borrow = 0 and in column 3, it is 1. Therefore, in column 4, first subtract 0 from 1 and from this result obtained subtract the borrow bit.

**15. Subtraction using 1's and 2's complements:**

Let us first we will understand concept of 1's and 2's complements, then subtraction methods using it.

**15.1 One's Complement Representation :** In a binary number, if each 1 is replaced by 0 and each 0 by 1, the resulting number is known as one's complement of the first number. In fact both the numbers are complement of each other. If one of these number is positive, then the other number will be negative with the same magnitude and vice-versa. For example, the 1's complement of a binary number 0101 is 1010. Thus $(0101)_2$ represents $(+5)_{10}$ whereas $(1010)_2$ represents $(-5)_{10}$ in this representation, using 8 bits, the decimal number –25 is expressed as the 1's complement of +25 (00011001) as 11100110. Note that in 1's complement representation also MSB is 0 for positive numbers and 1 for negative numbers. In one's complement representation $(+7)_{10} = (0111)_2$ and $(-7)_{10} = (1000)_2$.

Thus in 4-bit, maximum positive number is + 7 and maximum negative number is – 7. In general, it is found that for an n-bit number, the maximum positive number which can be represented in 1's complement representation is $(2^{n-1} - 1)$ and the maximum negative number is $-(2^{n-1} - 1)$.

**15.2 Two's Complement Representation :** The two's complement of binary number is obtained by adding 1 to 1's complement. For example, 2's complement of 0110 is 1010. Since 0110 represents $(+6)_{10}$. Therefore, 1010 represents $(-6)_{10}$ in 2's complement representation. In this representation also, if the MSB is 0, the number is positive, whereas if the MSB is 1 the number is negative. More details regarding 2's complement of a number and its use in binary addition are explained in chapter 4. For an n-bit number, the maximum positive number which can be represented in 2's complement form is $(2^{n-1} - 1)$ and the maximum negative number is $-2^{n-1}$. Table below gives sign magnitude, 1's and 2's complement numbers represented by 4-bit binary numbers.

**Table 1 : Signed binary numbers using 4-bits**

| Decimal Numbers | Binary Number | | |
|---|---|---|---|
| | Sign Magnitude | One's Complement | Two's Complement |
| 0 | 0000 | 0000 | 0000 |
| 1 | 0001 | 0001 | 0001 |
| 2 | 0010 | 0010 | 0010 |
| 3 | 0011 | 0011 | 0011 |
| 4 | 0100 | 0100 | 0100 |
| 5 | 0101 | 0101 | 0101 |
| 6 | 0110 | 0110 | 0110 |
| 7 | 0111 | 0111 | 0111 |
| – 7 | 1111 | 1000 | 1001 |
| – 6 | 1110 | 1001 | 1010 |
| – 5 | 1101 | 1010 | 1011 |
| – 4 | 1100 | 1011 | 1100 |
| – 3 | 1011 | 1100 | 1101 |
| – 2 | 1010 | 1101 | 1110 |
| – 1 | 1001 | 1110 | 1110 |
| 0 | 1000 | 1111 | 1111 |

## 15.3 Subtraction by 1's Complement method :

To subtract a smaller number from a larger number, the 1's complement method is as follows :

1. Determine the 1's complement of the smaller number.
2. Add the 1's complement to the larger number.
3. Remove the carry and add it to the result. This is called end-around carry.

**Example 1 :** Subtract 1001 from 1100 using the 1's complement method.

**Solution :**

**Direct subtraction**

$$\begin{array}{r} 1100 \\ -\ 1001 \\ \hline 0011 \end{array}$$

**1's complement method**

$$\begin{array}{rl} & 1100 \\ + & 0110 \rightarrow \text{1's complement of } 1001 \\ \hline \text{Add End around carry} \rightarrow & \boxed{1}\ 0010 \\ & \hookrightarrow\ +\ 1 \\ \hline & 0011 \quad \text{Final answer} \end{array}$$

**Example 2 :** Subtract 11001 from 11111 using 1's complement method.

**Solution :**

**Direct subtraction**

$$\begin{array}{r} 11111 \\ -\ 11001 \\ \hline 00110 \end{array}$$

11001

**1's complement method**

$$\begin{array}{r} 11111 \\ +\ 00110 \rightarrow \text{1's complement of} \\ \hline \boxed{1}\ 00101 \\ \hookrightarrow +\ 1 \\ \hline 00110 \end{array}$$

**Example 3 :** Subtract 1000 from 0111 using 1's complement method.

**Solution :**

**Direct subtraction**

$$\begin{array}{r} 7 \\ -\ 8 \\ \hline -\ 1 \end{array} \qquad \begin{array}{r} 0111 \\ -\ 1000 \\ \hline -\ 0001 \end{array}$$

For direct subtraction subtract smaller number from larger number and assign the sign of larger number.

**1's complement method**

$$\begin{array}{r} 0111 \\ 0111 \rightarrow \text{1's complement} \\ \hline 1110 \leftarrow \text{There is no carry,} \end{array}$$

$\therefore$ result is –ve and is in 1's complement form.

$\therefore$ result is – 0001.

## 15.4 Subtraction using 2's Complement method

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

**Example 1 :** Binary number is 1010. 1's complement of 1010 is 0101.
Add 1 to 0101

$$\text{i.e.}\quad \begin{array}{r} 0101 \\ +\ 1 \\ \hline 0110 \end{array}$$

**Example 2 :** Binary number 10101 has 2's complement as,

$$\begin{array}{rl} 01010 & \text{1's complement of 10101} \\ +\ 1 & \text{Add 1} \\ \hline 01011 & \text{2's complement of 10101} \end{array}$$

To subtract a smaller number from larger one, 2's complement method is applied as follows :

1. Determine the 2's complement of the smaller number.
2. Add the 2's complement to the larger number.
3. Discard the carry (there is always a carry in this case).

**Example 1 :** Subtract 1101 from 1111 using 2's complement method.

**Solution : Direct subtraction**

$$\begin{array}{r} 1111 \\ -\ 1101 \\ \hline 0010 \end{array}$$

**2's complement method**

$$\begin{array}{r} 1111 \\ +\,0011 \\ \hline \boxed{1}\;\;0010 \end{array} \rightarrow \text{2's complement of 1101}$$

↑

discard the carry

Final result is 0010.

**Example 2 :** Perform the following subtraction.

**Solution : Direct method**

$$\begin{array}{r} 1110 \\ -\,1001 \\ \hline 0101 \end{array} \qquad \begin{array}{r} 1110 \\ +\,0111 \\ \hline \boxed{1}\;\;0101 \end{array} \rightarrow \text{2's complement of 1001}$$

↑

discard the carry Result : 0101

**Example 3 :** Perform subtraction 11000 from 11111.

**Solution : Direct method**

$$\begin{array}{r} 11111 \\ -\,11000 \\ \hline 00111 \end{array} \qquad \begin{array}{r} 11111 \\ +\;\;01000 \\ \hline \boxed{1}\;\;00111 \end{array} \rightarrow \text{2's complement of 11000}$$

↑

discard the carry Result : 00111

$$\begin{array}{r} 111112 \\ -\,110002 \\ \hline 001112 \end{array} \qquad \begin{array}{r} 3110 \\ -\,2410 \\ \hline -\,0710 \end{array}$$

Binary subtraction Decimal subtraction

**16. BCD Code:**

The BCD code expresses each digit in a decimal number by its 4 bit binary equivalent i.e. nibble. The 8421 code is a type of binary coded decimal (BCD) code and is composed of four bit having binary weight ($2^3$ $2^2$ $2^1$ $2^0$).

With four bit combination 16 numbers can be represented. But in 8421 code only ten of these are used as given in Table below.

**Table**

| 8421 (BCD) | Decimal |
|---|---|

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

The six code combinations 1010, 1011, 1100, 1101, 1110 and 1111 are invalid in the 8421 BCD code.

**Example 1 :** Convert each of the following decimal numbers into BCD.

**Solution :**

(i) 429

| 4 | 2 | 9 |
|---|---|---|
| ↓ | ↓ | ↓ |
| 0100 | 0010 | 1001 |

(ii) 9745

| 9 | 7 | 4 | 5 |
|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ |
| 1001 | 0111 | 0100 | 0101 |

(iii) 39.2

| 3 | 9 | · | 2 |
|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ |
| 0011 | 1001 | · | 0010 |

**Example 2 :** Find the decimal number represented by the following BCD code.

**Solution :** (i) 10000110

Make group of four bit

| 1000 | 0110 |
|---|---|
| ↓ | ↓ |
| 8 | 6 |

(ii) 100101110101

| 1001 | 0111 | 0101 |
|---|---|---|
| ↓ | ↓ | ↓ |
| 9 | 7 | 5 |

**Advantages and Disadvantages of BCD :**

**Advantages :**

Decimal numbers can be easily converted into BCD numbers and vice versa. Only we have to remember the binary equivalent of decimal 0 through 9. Since we convert only one digit at a time.

**Disadvantages :**

1.BCD numbers are not used in modern computers because of its limited value. A modern computer must be able to process alphanumeric. Therefore, computer uses binary numbers rather than BCD numbers.

2.The rules for binary addition are not applicable to entire 8421 BCD number, but only to the individual 4 bit group. For example, consider the addition of 12 and 8.

| Decimal | Binary | BCD | |
|---|---|---|---|
| 12 | 1100 | 0001 | 0010 |
| + 08 | + 1000 | + 0000 | 1000 |
| 20 | 10100← 20 | 0001 | 1010 |

0001 1010 is not the BCD equivalent of 20. As 1010 does not exist in the BCD code.

**17. Excess- 3 code:**

The ease of conversion is the most important advantage of BCD, i.e. 8421 code. However the rules of binary addition are not valid for 8421 code and hence excess-3 code is developed. The excess 3 is an important 4 bit code sometimes used with BCD number. It is derived by adding 3 to each decimal digit and then converting the result to the 4-bit binary. Since no definite weight can be assigned to the four digit position, Excess-3 is an unweighted code.

**Example 1 :** Excess-3 code for the decimal 2 is,

$$\begin{array}{r} 2 \\ +3 \\ \hline 5 \end{array} \rightarrow 0101$$

**Example 2 :** The excess-3 code for 9 is,

$$\begin{array}{r} 9 \\ +3 \\ \hline 12 \end{array} \rightarrow 1100$$

For two digit decimal number, excess-3 code can be written as given in the following examples.

**Example 3 :** To convert 15 to an excess-3 number, add 3 to each decimal number.

| 1 | 5 |
|---|---|
| + 3 | + 3 |
| 4 | 8 |
| ↓ | ↓ |
| 0100 | 1000 |

So 0100 · 1000 in the excess-3 code stands for decimal 15.

**Example 4 :** Excess-3 code of 39 is,

| 3 | 9 | |
|---|---|---|
| + 3 | + 3 | |
| 6 | 12 | |
| ↓ | ↓ | |
| 0110 | 1100 | ← Excess-3 for 39. |

**Example 5 :** Excess-3 code of 430 is,

| 4 | 3 | 0 | |
|---|---|---|---|
| + 3 | + 3 | + 3 | |
| 7 | 6 | 3 | |
| ↓ | ↓ | ↓ | |
| 0111 | 0110 | 0011 | ← Excess-3 for $430_{10}$ |

Table 1.9 shows the Excess-3 code for decimal and BCD.

**Table 1.9 : Excess-3 code**

| Decimal | BCD | Excess-3 |
|---|---|---|
| 0 | 0000 | 0011 |
| 1 | 0001 | 0100 |
| 2 | 0010 | 0101 |
| 3 | 0011 | 0110 |
| 4 | 0100 | 0111 |
| 5 | 0101 | 1000 |
| 6 | 0110 | 1001 |
| 7 | 0111 | 1010 |
| 8 | 1000 | 1011 |
| 9 | 1001 | 1100 |

To convert excess-3 to decimal, subtract 3 from excess-3 code and then convert the difference into decimal number.

**18. Gray Code:**

The gray code is an unweighted code, which means that there are no specific weights assigned to the bit position. Each gray-code number differs from the preceding number by a single bit.

**Table : Gray code**

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

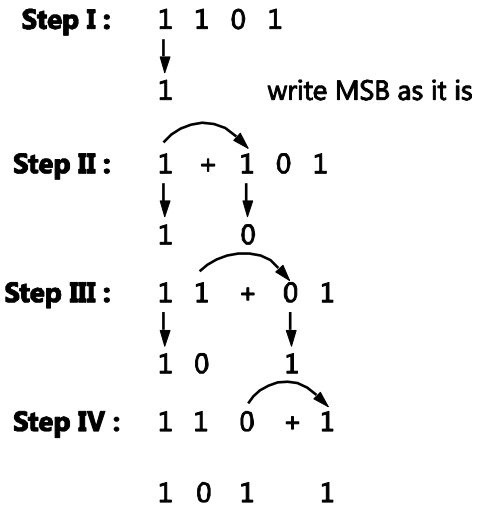
The Table above shows listing of gray code number for decimal and binary from 0 to 15. Gray code can have any number of bit. Considering gray code from decimal 3 to 4, the gray code changes from 0010 to 0110 while the binary code for the same i.e. 3 to 4 changes from 0011 to 0100.

**Binary to Gray CONVERSION:**

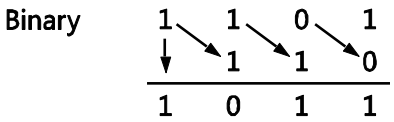
The MSB of binary is copied as it is; this represents the MSB of gray code. Then going from left to right add each adjacent pair of binary digits to get the next gray code digit. In addition, carry (if any) is ignored.

**Example 1:** Find gray code of binary number 1101.

**Solution:**




Or we may write the same example as 1101

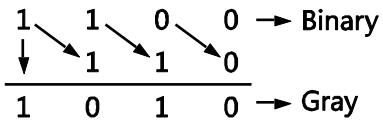



The gray code for binary 1101 is 1011.

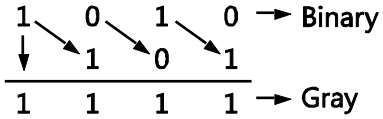
**Example 2:** Convert the following binary numbers to the gray code:
(i) $1100_2$, (ii) $1010_2$, (iii) $11010_2$, (iv) $1011110_2$.

**Solution:**

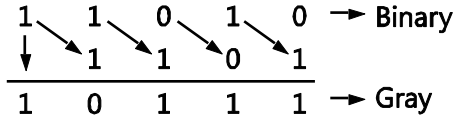
(i)



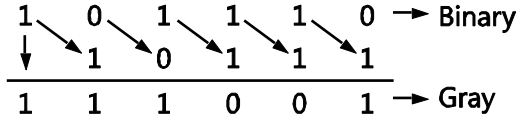

(ii) $1010_2$




(iii) $11010_2$




(iv) $1011110_2$

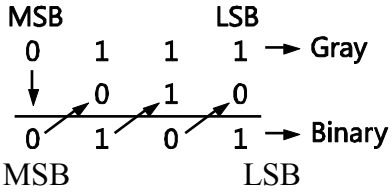


**Gray Code to Binary CONVERSION:**

To convert gray code to binary, first write the MSB (left most bit) same as corresponding digit in gray code and then add each MSB of binary to the next bit of gray code. If carry is generated, it should be ignored. Process is carried out till LSB is reached.

**Example 1:** Convert gray code number 0111 to binary.

**Solution:**




i.e. Gray code 0111 has binary 0101.

**19. Alphanumeric representation in ASCII codes:**

Non-weighted binary codes do not follow the positional weighting principle, i.e., digit position within the number has not assigned fixed value. Examples of non-weighted code are gray code, excess-3 code and alphanumeric code etc.

**Character CODE:**

In communication, along with ordinary number, we need some character, punctuation symbols and also control signals. The codes, we have studied so far, is not enough for data communication. For overcoming this problem, binary symbols are used for representing numbers, alphabetic character and special symbols.

The binary codes which are used to represent number, alphabetic character and special symbols are called *alphanumeric codes*.

**ASCII (American Standard Code for Information Interchange)**

The ASCII code is extensively used for data communication and in computers. It is a 7-bit code, so it can represent $2^7$ or 128 different characters. It can represent decimal number 0 to 9, letters of alphabet both upper and lower case, special symbols and control instructions. Each symbol has 7-bit code which is made up of a 3-bit group followed by a 4-bit group. The numbers are represented by 8421 BCD code preceded by 011. For instance, decimal number 3 is represented by 0110011. The Table 1.7 shows code combination of alphabet and other symbols. For example, upper case 'B' is represented by 1000010 whereas lower case 'b' is represented by 1100010, '+' is represented by 0101011.

**Table 1.7: American Standard Code for Information Interchange**

| | **Column** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| **Row** | MSB → / ↓ LSB | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| 0 | 0000 | NUL | DLE | SP | 0 | @ | P | | p |
| 1 | 0001 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 2 | 0010 | STX | DC2 | " | 2 | B | R | b | r |
| 3 | 0011 | ETX | DC3 | # | 3 | C | S | c | s |
| 4 | 0100 | EOT | DC4 | $ | 4 | D | T | d | t |
| 5 | 0101 | ENQ | NAK | % | 5 | E | U | e | u |
| 6 | 0110 | ACK | SYN | & | 6 | F | V | f | v |
| 7 | 0111 | BEL | ETB | ' | 7 | G | W | g | w |
| 8 | 1000 | BS | CAN | ( | 8 | H | X | h | x |
| 9 | 1001 | HT | EM | ) | 9 | I | Y | i | y |
| 10 | 1010 | LF | SUB | * | : | J | Z | j | z |
| 11 | 1011 | VT | ESC | + | ; | K | [ | k | { |
| 12 | 1100 | FF | FS | , | < | L | \ | *l* | \| |
| 13 | 1101 | CR | GS | – | = | M | ] | m | } |
| 14 | 1110 | SO | RS | . | > | N | ↑ | n | ~ |
| 15 | 1111 | SI | US | / | ? | O | ← | o | DEL |

Control functions in ASCII code are listed below.

**Control Functions:**

NUL – Null
SOH – Start of heading
STX – Start text
ETX – End text
EOT – End of transmission
ENQ – Enquiry
ACK – Acknowledge
BEL – Bell

| | | |
|---|---|---|
| BS | – | Backspace |
| HT | – | Horizontal tabulation |
| LF | – | Line feed |
| VT | – | Vertical tabulation |
| FF | – | Form feed |
| CR | – | Carriage return |
| SO | – | Shift out |
| SI | – | Shift in |
| DLE | – | Data link escape |
| $DC_1$, $DC_2$, $DC_3$, $DC_4$ | – | Device control 1 – 4 |
| NAK | – | Negative acknowledge |
| SYN | – | Synchronous idle |
| ETB | – | End of transmission block |
| CAN | – | Cancel |
| EM | – | End of medium |
| SUB | – | Substitute |
| ESC | – | Escape |
| FS | – | File separator |
| GS | – | Group separator |
| RS | – | Record separator |
| US | – | Unit separator |
| DEL | – | Delete |

# *Questions*

1. What is the radix used in the case of decimal, binary, octal and hexadecimal?
2. Write a note on BCD code.
3. Distinguish between binary code from BCD code.
4. What is excess-3 code?
5. What is a gray code? What are its main characteristics?
6. What is alphanumeric code?
7. Write a short note on ASCII code.
8. What is difference between binary and BCD number system?
9. Explain gray code system with suitable example.
10. Convert the following binary numbers into decimal, octal and hexadecimal.
    (a) (i) 1101, (ii) 110010, (iii) 10101010, (iv) 11111111, (v) 11011011, (vi) 111000
    (b) (i) 1101.11, (ii) 1010.001, (iii) 1001.111.
11. Convert the following decimal numbers into binary, octal and hexadecimal.
    (a) (i) 95, (ii) 123, (iii) 221, (iv) 252, (v) 529, (vi) 445.
    (b) (i) 33.87, (ii) 89.85, (iii) 107.23.

12. Convert the following octal numbers into decimal, binary and hexadecimal.
    (a) (i) 11, (ii) 82, (iii) 105, (iv) 187, (v) 256, (vi) 519.
    (b) (i) 18.19, (ii) 53.35, (iii) 128.85.
13. Convert the following hexadecimal numbers into decimal, binary and octal.
    (a) (i) 1E, (ii) ABC, (iii) F2, (iv) A99, (v) 23C, (vi) C92.
    (b) (i) AE.35, (ii) 98.B2, (iii) C1.77.
14. Convert the following numbers in BCD code.
    (i) 72, (ii) 365, (iii) 593
15. Express the following numbers in XS-3 code.
    (i) 92, (ii) 952, (iii) 1234
16. Convert the following numbers to gray code.
    (i) 01001, (ii) 101101, (iii) 10110110.
17. Solve the following
    (i) $(77)_{10} = (?)_8$, (ii) $(99)_{10} = (?)_{16}$, (iii) $(1011101) = (?)_{16}$
    (iv) $(FA)_{16} = (?)_{10}$

***---oOo---***