# * Computer Programming *



## Number systems and codes

**Introduction:**

— We all are familiar with the number system in which an ordered set of ten symbols — 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9, known as digits — are used to specify any number.

— This number system is popularly known as the decimal number system.

— The radix or base of this number system is '10' (number of distinct digits).

— Any number is a collection of these digits.

for example: 1980.0918 signifies a number with an integer part equal to 1980 and a fractional part equal to 0.0918, separated from the integer part with a radix point (.) also known as decimal point.

— There are some other systems also, used to represent numbers.

Some of the other commonly used number systems are:

binary, octal and hexadecimal number systems.

These number systems are widely used in digital systems like microprocessors, logic circuits, computers etc. and therefore, the knowledge of these number systems is very essential for understanding, analysing digital systems.

— computers and other digital circuits - use binary signals but are required to handle data which may be numeric, alphabets or special characters. Therefore, the information available in any other form is required to be converted into suitable binary form before it can be processed by digital circuits. This means that the information available in the form of numeral, alphabets and special characters or in any combination of these must be converted into binary format.

TO achieve this, a process of coding is employed where by each numeral, alphabet or special character is coded in a unique combination of 0's and 1's using a coding scheme, known as a code.

— The process of coding is known as encoding.

— There can be a variety of coding schemes (codes) to serve different purposes, such as arithmetic operations, data entry, error detection and correction etc. In digital systems, a large number of codes are in use. selection of a particular code depends on its suitability for the purpose. In one digital system, different codes may be used for different operations and it may be necessary to convert data from one code to another code.

## * Number system:

In general, in any number system there is an ordered set of symbols known as digits with rules defined for performing arithmetic operations like addition, multiplication, etc.

A collection of these digits makes a number which in general has two parts - integer and fractional, set apart by a radix point (•), that is

$$(N)_b = \underbrace{d_{n-1}\, d_{n-2}\, d_{n-3} \cdots d_1\, d_0}_{\text{integer portion}} \bullet \underbrace{d_{-1}\, d_{-2} \cdots d_{-m}}_{\text{fractional portion}}$$

(Radix point)

where

$N$ = a number

$b$ = radix or base of the number system

$n$ = number of digits in integer portion $(n-1, n-2, \ldots, 1, 0)$

$m$ = number of digits in fractional portion

$d_{n-1}$ = most significant digit (msd)

$d_{-m}$ = least significant digit (lsd)

and $0 \le (d_i \text{ or } d_{-f}) \le b-1$.

The digits in a number are placed side by side and each position in the number is assigned a weight or index of importance by some predesigned rule.

## * commonly used number systems:

The following table shows the

Table-1

| Number system | Base or radix(b) | Symbols used ($d_i$ or $d_{-f}$) | Weight assigned to position | | Example |
|---|---|---|---|---|---|
| | | | $i$ | $-f$ | |
| Binary | 2 | 0, 1 | $2^i$ | $2^{-f}$ | 1011.11 |
| Octal | 8 | 0, 1, 2, 3, 4, 5, 6, 7 | $8^i$ | $8^{-f}$ | 3567.25 |
| Decimal | 10 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 | $10^i$ | $10^{-f}$ | 3974.57 |
| Hexadecimal | 16 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 A, B, C, D, E, F | $16^i$ | $16^{-f}$ | 3FA9.56 |

Note: (1) The base is usually denoted as a suffix of the number. If no suffix is mentioned, the base is assumed to be 10.

(2) 10, 11, 12, 13, 14, 15 are denoted by A, B, C, D, E, F respectively.

(3) Value of the digit: We know that value of the number is based on each digit in it, for this, weights are attached to each digit position, on arriving at the value of the digit.

Basically, there are two types of values as follows:

(i) Face value and (ii) place value.

Face value: Face value of the number is simply the number.

place value: The place value of a number relates to weights attached to the position of the

## * Decimal Number System :

The number system that is normally used in decimal system. while assigning weights to each digit of a number, rightmost digit gets the weight of unity and the successive digits to its left usually have weights $10^1, 10^2, 10^3$ and so on. Each digit is multiplied by its weight and the sum of product provides value of the number.

**Example (1) :**

Decimal number $7536_{10}$ may be expressed as

$$7 \times 10^3 = 7000$$
$$5 \times 10^2 = 500$$
$$3 \times 10^1 = 30$$
$$6 \times 10^0 = 6$$

Thus, $7 \times 10^3 + 5 \times 10^2 + 3 \times 10^1 + 6 \times 10^0 = 7536_{10}$

It should be noted that the digit 7 has a face value and a place value $10^3 = 1000$. Thus the digit represents the product $7 \times 10^3$.

**Example (2).** Decimal number $65355_{10}$ may be expressed as

$$6 \times 10^4 = 60000$$
$$5 \times 10^3 = 5000$$
$$3 \times 10^2 = 300$$
$$5 \times 10^1 = 50$$
$$5 \times 10^0 = 5$$

$6 \times 10^4 + 5 \times 10^3 + 3 \times 10^2 + 5 \times 10^1 + 5 \times 10^0 = 65355$

## * Binary systems:

The binary system may be defined as a number system, having two symbols 0 and 1. The base (or radix) of a binary number system is 2 and the symbols 0 and 1 are termed as binary digits or bits.

**Example(1):** $1101_2$ is a binary number, to find the decimal value of the binary number, powers of two (2) are used as weights in a binary system and is as follows:

$$1\times 2^3 = 8$$
$$1\times 2^2 = 4$$
$$0\times 2^1 = 0$$
$$1\times 2^0 = 1$$

Thus the decimal value of $1101_2$ is

$$1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0 = 13.$$

**Note:** (1) While assigning weights to each digit of a number, rightmost digit gets the weight of unity and the successive digits to its left usually have weights $2^1, 2^2, 2^3$ and so on. Each digit is multiplied by its weight and the sum of product provides the decimal value of the binary number

$1101_2$ — most significant bit; least significant bit.

The rightmost bit is called the least

— The leftmost bit is called most significant bit.

(2) Any number of 0's can be added to the left of the number without changing the value of the number.

(3) In the binary number system, a group of four bits is known as a nibble and a group of eight bits is known as a byte.

(4) Just as power of '10' are important in the decimal system of enumeration (countable), powers of '2' are important in binary systems.

We thus give in Table powers of 2 and their decimal equivalents.

The abbreviation K in table stands for 1024 which is approximately 1000, a kilo.

— Thus the notation 16K means $16 \times 1024 = 16384$.

— The abbreviation M (Mega) stands for $1024 \times 1024 = 1048576$.

— The abbreviation G (Giga) is used to represent $1024 \times 1024 \times 1024$ which is nearly a billion.

— The abbreviation T (Tera) is used to represent $1024 \times 1024 \times 1024 \times 1024$, which is nearly a trillion.

| power of 2 | Decimal equivalent | Abbreviation |
|---|---|---|
| $2^0$ | 1 | |
| $2^1$ | 2 | |
| $2^2$ | 4 | |
| $2^3$ | 8 | |
| $2^4$ | 16 | |
| $2^5$ | 32 | |
| $2^6$ | 64 | |
| $2^7$ | 128 | |
| $2^8$ | 256 | |
| $2^9$ | 512 | |
| $2^{10}$ | 1024 | 1 K |
| $2^{11}$ | 2048 | 2 K |
| $2^{12}$ | 4096 | 4 K |
| $2^{13}$ | 8192 | 8 K |
| $2^{14}$ | 16384 | 16 K |
| $2^{15}$ | 32768 | 32 K |
| $2^{16}$ | 65536 | 64 K |
| $2^{17}$ | 131072 | 128 K |
| $2^{18}$ | 262144 | 256 K |
| $2^{19}$ | 524288 | 512 K |
| $2^{20}$ | 1048576 | 1 M |
| $2^{30}$ | 1073741824 | 1 G |

## * Octal system:

The octal system is the number system consisting of eight symbols 0, 1, 2, 3, 4, 5, 6, and 7. The base of an octal number system is '8'. Thus the weight assigned to each digit in octal sytem is power of 8.

**Example (1):** The decimal value of the octal number 176 can be obtained in the following manner.

$$1 \times 8^2 = 64$$
$$7 \times 8^1 = 56$$
$$6 \times 8^0 = 6$$

Thus the derived decimal value of 176 is $1 \times 8^2 + 7 \times 8^1 + 6 \times 8^0 = 126$.

**Example (2):** The decimal value of the octal number 1116 can be obtained in the following manner:

$1\times 8^3 = 512$

$1\times 8^2 = 64$

$1\times 8^1 = 8$

$6\times 8^0 = 6$

Thus the derived decimal value of 1116 is $1\times 8^3+1\times 8^2+1\times 8^1+6\times 8^0 = 590$.

* **Hexadecimal system:**

The number system having symbols such as 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F is called hexadecimal system. It has 16 as the basis.

**Example(1):** The decimal value of hexadecimal number 84A8 is obtained as follows:

$8\times 16^3 = 32768$

$4\times 16^2 = 1024$

$A\times 16^1 = 160$

$8\times 16^0 = 8$

Thus the decimal value of 84A8 is $8\times 16^3+4\times 16^2+A\times 16^1+8\times 16^0 = 33960$.

**Example(2):** The decimal value of hexadecimal number 21E3 is obtained as follows:

$2\times 16^3 = 8192$

$1\times 16^2 = 256$

$E\times 16^1 = 224$

$3\times 16^0 = 3$

Thus the decimal value of 21E3 is

Example (3):

The decimal value of hexadecimal number 12AF is obtained as follows:

$1 \times 16^3 = 4096$

$2 \times 16^2 = 512$

$A \times 16^1 = 160$

$F \times 16^0 = 15$

sum 4783

Thus the decimal value of $(12AF)_{16}$ is $(4783)_{10}$.

i.e $(12AF)_{16} = 1 \times 16^3 + 2 \times 16^2 + A \times 16^1 + F \times 16^0$

$= (4783)_{10}$

## * Binary to Decimal conversion:

Method 1:

To convert a binary system number to a decimal system, the digits are obtained first. The sum of the products of the value of the digit in the decimal system and

**Example(1):**

convert $10110_2$ to decimal.

**Sol**

| Binary number | value of the digit (in decimal system) |
|---|---|
| 1 | $2^5 = 32$ |
| 0 | $2^4 = 16$ |
| 1 | $2^3 = 8$ |
| 1 | $2^2 = 4$ |
| 0 | $2^1 = 2$ |
| 1 | $2^0 = 1$ |

Thus the decimal value of the binary number $10110_2$ is

$1\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 = 45.$

HW → convert the binary number $111111_2$ into decimal number.

**Method(2)**

The procedure to convert a binary number to decimal is called dibble-dabble method

We start with the left hand bit.
Multiply this value by 2, and add the next bit
Again multiply by 2 and add the next bit.
i.e.
Stop when the bit on extreme right hand side is reached.

Example(1) Convert $100101_2$ to decimal.

Sol Left hand bit 1

Multiply by 2 and add next bit $2\times1+0 = 2$

Multiply by 2 and add next bit $2\times2+0=4$

Multiply by 2 and add next bit $2\times4+1 = 9$

Multiply by 2 and add next bit $2\times9+0 = 18$

Multiply by 2 and add next bit $2\times18+1 = 37$

$\therefore 100101_2 = 37_{10}$.

**Method (3)**

To convert binary number to decimal number is as under:

(1) Write the binary number.

(2) Write the weights $2^0, 2^1, 2^2, 2^3$ etc., under the binary digits starting with the bit on right hand side.

(3) Cross out weights under zeros.

(4) Add the remaining weights.

**Example (1):** Convert $1101_2$ into equivalent decimal number.

Sol

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 0 | 1 | Binary number |
| 8 | 4 | 2 | 1 | write weights |
| 8 | 4 | ~~2~~ | 1 | cross out weights under zeros. |

$8+4+0+1 = 13$ add weights

$\therefore 1101_2 = 13$

Hw → Convert $11001101001_2$ into equivalent decimal number.

Ans: $3289_{10}$.

* **Decimal to Binary conversion:**

A systematic way to convert a decimal number into equivalent binary is known as double-dabble method. This method involves successive division by 2 and recording the remainder (the remainder will be always 0 or 1). The division is stopped when we get a quotient of '0' with a remainder of 1. The remainders when read upwards give the equivalent binary number.

Example(1) convert decimal number 747 into its equivalent binary number.

Sol

| Division | Quotient | Remainder |
|---|---|---|
| $747/2$ | 373 | 1 |
| $373/2$ | 186 | 1 |
| $186/2$ | 93 | 0 |
| $93/2$ | 46 | 1 |
| $46/2$ | 23 | 0 |
| $23/2$ | 11 | 1 |

| | | |
|---|---|---|
| 5/2 | 2 | 1 |
| 2/2 | 1 | 0 |
| 1/2 | 0 | 1 |

Thus the binary equivalent of 747 is $1011101011_2$

H.W. → Convert decimal number 101 into its equivalent binary number.

→ Convert $3289_{10}$ into binary.

Soln

| | | |
|---|---|---|
| 2 | 3289 | |
| 2 | 1644 | remainder 1 |
| 2 | 822 | remainder 0 |
| 2 | 411 | remainder 0 |
| 2 | 205 | remainder 1 |
| 2 | 102 | remainder 1 |
| 2 | 51 | remainder 0 |
| 2 | 25 | remainder 1 |
| 2 | 12 | remainder 1 |
| 2 | 6 | remainder 0 |
| 2 | 3 | remainder 0 |
| 2 | 1 | remainder 1 |
| | 0 | remainder 1 |

$\therefore 3289_{10} = 110011011001_2$

HW → convert $10_{10}$ into binary.

HW → convert $25_{10}$ into binary

---

## Binary fractions.

So far we have discussed only whole numbers. However, to represent fractions is also important.

The decimal number 2568 is represented as $2568 = 2000+500+60+8$

$= 2\times10^3 + 5\times10^2 + 6\times10^1 + 8\times10^0$

Similarly 25.68 can be represented as

$25.68 = 20 + 5 + 0.6 + 0.08$

$= 2\times10^1 + 5\times10^0 + 6\times10^{-1} + 8\times10^{-2}$

(2) $0.238 = 0.2 + 0.03 + 0.008$

$= 2\times10^{-1} + 3\times10^{-2} + 8\times10^{-3}$

## * Conversion of Binary to Decimal:

In the binary system, the weights of the binary bits after the binary point, can be written as

$0.1011 = 0.1 + 0.00 + 0.001 + 0.0001$

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

$= 1\times\frac{1}{2} + 0\times\frac{1}{2^2} + 1\times\frac{1}{2^3} + 1\times\frac{1}{2^4}$

$= 0.5 + 0 + 0.125 + 0.0625$

$= 0.6875$ (decimal)

→ Determine the decimal numbers represented by the following binary numbers

(1) 111011.101 (2) 10101.10101 (3) 1100.1011

(4) 1001.0101 (5) 0.10101 (6) 11000.0011

Soln:

(1) $(111011.101)_2 = 1\times2^5+1\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$.

$= 32+16+8+0+2+1+\frac{1}{2}+0+\frac{1}{8}$

$= 59+0.5+0.125$

$= (59.625)_{10}$.

(3) $(1100.1011)_2 = 1\times2^3+1\times2^2+0\times2+0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}+1\times2^{-4}$

$= 8+4+0+0+\frac{1}{2}+0+\frac{1}{8}+\frac{1}{32}$

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

$= (12.6875)_{10}$.

---

## Conversion of decimal to binary.

To convert a decimal to binary, a method of successive multiplecation by 2 is used. After each multiplication, the integer part is noted seperately and the fraction is again multiplied by 2 till the remainder becomes zero.

Sometimes it is possible that the remainder doesnot become zero even after many stages. In such case, an approximation is made up to a certain

number of bits after the binary point.

A similar procedure is adopted for a number having both integer and fractions.

Binary fractions can be added, subtracted etc. as the decimal numbers.

→ Example:
Express the number 0.6875 into binary equivalent.

Sol$^{n}$:

| Fraction | Fraction×2 | Remainder new fraction | Integer. |
|---|---|---|---|
| 0.6875 | 1.375 | 0.375 | 1 (MSB) |
| 0.375 | 0.75 | 0.75 | 0 |
| 0.75 | 1.5 | 0.5 | 1 |
| 0.5 | 1 | 0 | 1 (LSB) |

The binary equivalent is 0.1011

→ Convert the decimal number 0.634 into its binary equivalent.

Sol$^{n}$.

| Fraction | Fraction×2 | Remainder new fraction | Integer. |
|---|---|---|---|
| 0.634 | 1.268 | 0.268 | 1 (MSB) |
| 0.268 | 0.536 | 0.536 | 0 |
| 0.536 | 1.072 | 0.072 | 1 |
| 0.072 | 0.144 | 0.144 | 0 |
| 0.144 | 0.288 | 0.288 | 0 |
| 0.288 | 0.576 | 0.576 | 0 |
| 0.576 | 1.152 | 0.152 | 1 (LSB) |

It is seen that it is not possible to get a zero as remainder even after 7 stages.

approximation can be made and the process terminated here.

The binary equivalent is $0.10100001$.

It is important to note that the decimal equivalent of $0.10100001$ (binary) is $0.6328125$ (decimal)

→ Convert the decimal number $39.12$ into binary.

Soln:

Taking the integer part first:

| | | |
|---|---|---|
| 2 | 39 | |
| 2 | 19 | remainder 1 (LSB) |
| 2 | 9 | remainder 1 |
| 2 | 4 | remainder 1 |
| 2 | 2 | remainder 0 |
| 2 | 1 | remainder 0 |
| | 0 | remainder 1 (MSB) |

$(39)_{10} = (111001)_2$

Taking the fractional part:

| Fraction | Fraction × 2 | Remainder new fraction | Integer |
|---|---|---|---|
| 0.12 | 0.24 | 0.24 | 0 (MSB) |
| 0.24 | 0.48 | 0.48 | 0 |
| 0.48 | 0.96 | 0.96 | 0 |
| 0.96 | 1.92 | 0.92 | 1 |
| 0.92 | 1.84 | 0.84 | 1 |
| 0.84 | 1.68 | 0.68 | 1 |
| 0.68 | 1.36 | 0.36 | 1 (LSB) |

This fraction cannot be converted into exact from number. The process can be terminated here.

The result is 0.0001111.

Adding the binary equivalent of 39 and 0.12

$$\begin{array}{r} 100111.0000000 \\ 0.0001111 \\ \hline 100111.0001111 \end{array}$$

$\therefore (39.12)_{10} = (100111.0001111)_2$.

→ Express the following decimal numbers in the binary form.

(a) 25.5 (b) 10.625 (c) 0.6875.

## * Binary Arithmetic :-

We all 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 fact, binary arithmetic is much simpler than decimal arithmetic because here only two digits 0 and 1 are involved.

### Binary addition :-

The rules of binary addition are given by table:

| 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 two rows above, there is no carry, that is, carry = 0, whereas in the fourth row a carry is produced (Since the largest digit possible is 1), i.e, Carry = 1, and similar to decimal addition it is added to the next higher binary position.

→ Add the binary numbers:

(i) 1011 and 1100 (ii) 0101 and 1111

Sol$^{n}$:

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

↑ carry

Sol$^{n}$:

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

carry ← carry.

→ Add the binary numbers:

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

Sol$^{n}$:

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

← Two pair of 1's in the previous column.

← one pair of 1's in the previous column.

Even number of 1's in the column.

carry ↑ ↑ ↑ ↑ ↑ ← odd number of 1's in the column.

∴ the sum = 111110010.

From the above example, we observe the following:

(i) If the number of 1's to be added in a column is even then the sum bit is 0, and if the number of 1's to be added in a column is add then the sum bit is 1.

(ii) Every pair of 1's in a column produces a carry (1) to be added to the next higher bit column.

=

Binary subtraction:

The rules of binary subtraction are given in table:

| Minuend | subtrahend | Difference | Borrow. |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

Except in the second row above, the borrow = 0. when the borrow = 1, as in the second row, this is subtracted from the next higher binary bit as it is done in decimal subtraction.

→ Subtract 0111 from 1010.

Sol$^{n}$:

$$\begin{array}{r} 1010 \\ -\ 0111 \\ \hline 0011 \end{array}$$

The least significant digit in the first number is '0'. So we borrow 1 from next digit and subtract 1 to give 1.

Now in the second column we have '0'. So we again borrow 1 from the next higher column and subtract 1 to give 1. In the third column, we borrow 1 from the next higher column and 1 − 1 gives '0'.

In the fourth column, 0 (after lending) − 0 gives 0

(1) $\begin{array}{r} 11001 \\ -1110 \\ \hline 1011 \end{array}$ (2) $\begin{array}{r} 1011 \\ -0110 \\ \hline 0101 \end{array}$ (3) $\begin{array}{r} 100 \\ -001 \\ \hline 011 \end{array}$.

## Binary Multiplication:

The four basic rules for binary multiplication are:

$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

The method of binary multiplication is similar to decimal multiplication.

In binary, each partial product is either zero (multiplication by 0) or exactly same as the multiplicand (multiplication by 1).

Examples:

(1) Multiply 1001 by 1101

Sol$^n$:

$$\begin{array}{r} 1001 \\ \times\ 1101 \\ \hline 1001 \\ 0000 \\ 1001 \\ 1001 \\ \hline 1110101 \end{array}$$

1001 ← Multiplicand; 1101 ← Multiplier; 1001, 0000, 1001, 1001 ← partial products; 1110101 ← final product.

(2) (i) Multiply $1011_2$ by $101_2$

(ii) Convert $1011_2$ and $101_2$ into decimal number. Multiply them, convert the result into binary and Compare with the result of (i).

Sol$^n$ (i)

$$\begin{array}{r} 1011 \\ \times\ 101 \\ \hline 1011 \\ 0000 \\ 1011 \\ \hline 110111 \end{array}$$

(ii) 1 0 1 1 Binary number

8 4 2 1 Write weights

8 ~~4~~ 2 1 Cross out weights under zero

8 + 2 + 1 = 11 Add weights.

1 0 1 Binary number

4 2 1 Write weights

4 ~~2~~ 1 Cross out weights under zero.

4 + 1 = 5 Add weights.

| | | |
|---|---|---|
| 2 | 55 | |
| 2 | 27 | remainder 1 |
| 2 | 13 | remainder 1 |
| 2 | 6 | remainder 1 |
| 2 | 3 | remainder 0 |
| 2 | 1 | remainder 1 |
| | 0 | remainder 1 |

$\therefore (55)_{10} = (110111)_2$.

$\therefore$ The result is same as in part (i)

H.W. → Multiply $11101_2$ by $10001_2$

**Binary Division:**

Binary division is obtained using the same procedure as decimal division.

Examples

→ (1) Divide 1110101 by 1001

Soln:

```
1001) 1110101 (1101
      1001
      ----
       1011
       1001
       ----
       001001
          1001
          ----
          0000
```

Ans: 1101

→ (2) Divide 10100111 by 1001

Soln

```
1001) 10100111 (1001
      1001
      -------
       0001011
          1001
          ----
          00101
```

remainder 101
quotient 1001.

(3) →

(i) Divide 110110 by 1.01

(ii) Convert 110110 and 101 into equivalent decimal number obtain division, convert results into binary and compare the results with those in part (i).

Sol^n: (i) 101) 110110 (101 quotient.

101

00111

101

100 remainder

(ii) 1 1 0 1 1 0 Binary number

32 16 8 4 2 1 write weights

32 16 ~~8~~ 4 2 ~~1~~ cross out weights under zero

32+16+4+2 = 54 Add weights

and 1 0 1 Binary number

4 2 1 write weights

4 ~~2~~ 1 cross out weights under zero.

4+1=5 Add weights.

NOW 5) 54 (10 → quotient

50

4 → remainder.

Quotient

| | | |
|---|---|---|
| 2 | 10 | |
| 2 | 5 | remainder 0 |
| 2 | 2 | remainder 1 |
| 2 | 1 | remainder 0 |
| | 0 | remainder 1 |

Remainder.

| | | |
|---|---|---|
| 2 | 4 | |
| 2 | 2 | remainder 0 |
| 2 | 1 | remainder 0 |
| | 0 | remainder 1 |

∴ The quotient is 1010 and the remainder is 100.

These are the same as in part (i).

## Signed Binary numbers:

On the decimal number system a plus (+) sign is used to denote a positive number and a minus (−) sign for denoting a negative number. The plus sign is usually dropped, and the absence of any sign means that the number has positive value. This representation of numbers is known as signed number.

As is well known, digital circuits can understand only two symbols, 0 and 1; therefore, we must use the same symbols to indicate the sign of the number also.

Normally, an additional bit is used as the sign bit and it is placed as the most significant bit

A '0' is used to represent a positive number and a 1 to represent a negative number.

for example:

An 8-bit signed number 01000100 represents a positive number and its value (magnitude) is $(1000100)_2 = (68)_{10}$

The left most '0' (MSB) indicates that the number is positive.

On the other hand, in the signed binary form, 11000100 represents a negative number with magnitude $(1000100)_2 = (68)_{10}$.

the left most position (MSB) indicates

that the number is negative and the other seven bits give its magnitude.

This kind of representation for signed numbers is known as sign-magnitude representation.

---

→ Find the decimal equivalent of the following binary numbers assuming sign-magnitude representation of the binary numbers.

(a) 101100 (b) 001000 (c) 0111 (d) 1111.

Sol$^{n}$: (a) 101100.

Sign bit is 1, which means the number is negative.

Magnitude = 01100

$= 0\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 0\times2^0$

$= 0 + 8 + 4 + 0 + 0$

$= (12)_{10}$.

$\therefore (101100)_2 = (-12)_{10}$.

(b) 001000

Sign bit is 0, which means the number is positive.

Magnitude = 01000

$= 0\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 0\times2^0$

$= (8)_{10}$

$\therefore (001000)_2 = (+8)_{10}$

→ Express in 16-bit signed binary system.

(a) +8, (b) −8 (c) +165 (d) −165.

Soln: (a)

| | |
|---|---|
| 2 | 8 |
| 2 | 4 remainder 0 |
| 2 | 2 remainder 0 |
| 2 | 1 remainder 0 |
| | 0 remainder 1. |

The binary number is 1000.

For the 16 bit system, we use 16 bits, 0 (which stands for +) in the left most position, 1000 in the last 4 bits and 0 in the remaining 11 position.

∴ The signed 16 bit binary number is

+8 = 0000 0000 0000 1000

(b) Using 1. in the left most position (to represent the '−' sign). The rest of the representation is the same as in part (a)

∴ −8 = 1000 0000 0000 1000.

---

→ Represent $(-17)_{10}$ in sign − magnitude.

Soln:

| | |
|---|---|
| 2 | 17 |
| 2 | 8 remainder 1 |
| 2 | 4 remainder 0 |
| 2 | 2 remainder 0 |
| 2 | 1 remainder 0 |
| | 0 remainder 1 |

Using 1 in the left most position (for '−' sign).

$(-17)_{10} = (10001)$.

## 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 the one's complement of the first number. In fact, both the numbers are complement of each other.

If one of these numbers is positive, then the other number is negative with the same magnitude and vice-versa.

**For example:**

$(0101)_2$ represents $(+5)_{10}$, whereas $(1010)_2$ represents $(-5)_{10}$ in this representation. This method is widely used for representing signed numbers.

In this representation also, MSB is '0' for positive numbers and 1 for negative numbers.

→ Find the one's complement of the following binary numbers.

(a) 010011001 (b) 11011010.

**Sol":** (a) one's complement of the binary number 010011001 is 101100110

(b) One's complement of the binary number 11011010 is 00100101.

→ Represent the following numbers in one's complement form.

(c) +15 and -15.

Sol": In one's complement representation

(a) $(+7)_{10} = (0111)_2$

and $(-7)_{10} = (1000)_2$

(b) $(+8)_{10} = (01000)_2$

and $(-8)_{10} = (10111)_2$

(c) $(+15)_{10} = (01111)_2$

$(-15)_{10} = (10000)_2$.

From the above examples, it can be observed 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 $-(2^{n-1}-1)$.

## Two's complement Representation:

If 1 is added to 1's complement of a binary number, the resulting number is known as the two's complement of the binary number.

For example:

2's complement of 0101 is 1011.

Since 0101 represents $(+5)_{10}$, therefore, 1011 represents $(-5)_{10}$ 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.

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 $-2^{n-1}$.

The below table gives Sign-magnitude, 1's and 2's complement numbers represented by 4-bit binary numbers.

Sign-magnitude, 1's and 2's complement representation using four bits:

| Decimal number | 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 |
| -8 | — | — | 1000 |
| -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 | 1111 |
| -0 | 1000 | 1111 | — |

From the table, it is observed that the maximum positive number is 0111 = +7, where as the maximum negative number is 1000 = −8 using four bits in 2's complement format.

It is also observed that the 2's complement of the 2's complement of a number is the number itself.

For example:

Let the binary number 0101 = A (say)

Now number A = 0101

1's complement $\bar{A}$ = 1010

Add 1 — 1

2's complement A' = 1011

1's complement of A' is $\bar{A'}$ = 0100

2's complement of $\bar{A'}$ is A'' = 0101

$$\left(\because \begin{array}{r} 0100 \\ +\quad 1 \\ \hline 0101 \end{array}\right)$$

∴ The 2's complement of the 2's complement of a number is the number itself.

→ Find the 2's complement of the numbers.

(i) 01001110 (ii) 00110101

Sol:

(i) Number 01001110

1's complement 10110001

Add 1 — 1

10110010

(iii) Number 00110101

1's complement 11001010

Add 1 1

11001011

From the above example, we observe the following:

(1) If the LSB of the number is 1, its 2's complement is obtained by changing each '0' to '1' and '1' to '0' except the least significant bit

(2) If the LSB of the number is '0', its 2's complement is obtained by scanning the number from the LSB to MSB bit by bit and retaining the bits as they are up to and including the occurence of the first 1 and complement all other bits.

→ Find two's complement of the numbers.

(i) 01100100 (ii) 10010010 (iii) 11011000 (iv) 01100111

Sol: Using the rules of conversion given above, we obtain

(i) Number → 01100100

2's complement → 10011100

(ii) Number → 10010010

2's complement → 01101110

(iii) Number → 11011000

2's complement → 00101000

(iv) Number → 01100111

2's complement → 10011001

→ Represent $(-17)_{10}$ in
(i) Sign-magnitude
(ii) One's complement
(iii) two's complement.

Sol$^n$:

| | |
|---|---|
| 2 | 17 |
| 2 | 8 remainder 1 |
| 2 | 4 remainder 0 |
| 2 | 2 remainder 0 |
| 2 | 1 remainder 0 |
| | 0 remainder 1 |

$(17)_{10} = (10001)_2$.

Using 0 in the left most position for '+' sign.

$(+17)_{10} = (0\,10001)_2$

∴ The minimum number of bits required to represent $(+17)_{10}$ in signed number format is six.

Therefore, $(-17)_{10}$ is represented by

(i) 110001 in sign-magnitude representation

(ii) 101110 in 1's complement representation.

(iii) 101111 in 2's complement representation

---

→ Find the largest positive and negative numbers which can be stored with 8 bits.

Sol$^n$: The largest positive number is $(2^{8-1}-1) = 127$

$0111\,1111 = +127$.

The largest negative number is

1000 0000 = −128

Thus, with 8 bits we can store numbers between −128 and +127.

## 2's complement Addition, Subtraction:

The use of 2's complement representation has simplified the computer hardware for arithmetic operations. When A and B are to be added, the B bits are not inverted so that we get,

$$S = A + B. \quad \text{——} (1)$$

when B is to be subtracted from A, the computer hardware forms the 2's complement of B and then adds it to A.

Thus $S = A + B'$

$= A + (-B) = A - B.$ ——(2)

Equations (1) and (2) represent algebraic addition and subtraction. A and B may represent either positive or negative numbers.

**Example:** If A = −24 and B = +16, (a) represent A & B in 8-bit 2's complement (b) find A+B (c) find A−B.

**Soln:** (a)

| | |
|---|---|
| 2 | 24 |
| 2 | 12 remainder 0 |
| 2 | 6 remainder 0 |
| 2 | 3 remainder 0 |
| 2 | 1 remainder 1 |

↑

$24 = (11000)_2$

$= (0001\ 1000)_2$ which is 8-bit form.

$24 = 0001\ 1000$

$\overline{24} = 1110\ 0111$ (1's complement)

$+\qquad\qquad 1$

$-24 = 1110\ 1000$ (2's complement)

Now

| | |
|---|---|
| 2 \| 16 | |
| 2 \| 8 | remainder 0 |
| 2 \| 4 | remainder 0 |
| 2 \| 2 | remainder 0 |
| 2 \| 1 | remainder 0 |
| 0 | remainder 1 |

$B = +16 = (10000)_2$

$= (00010000)_2$ (8-bit form)

(b) $-24 = 1110\ 1000$

$+16 = 0001\ 0000$

$-8 = 1111\ 1000$

(c) $B = 0001\ 0000$

$\overline{B} = 1110\ 1111$ (1's complement)

$+\qquad\qquad 1$

$B' = 1111\ 0000$ (2's complement)

Now $-24 = 1110\ 1000$

$+(-16) = 1111\ 0000$

$-40 = 1101\ 1000$

Note : For the above problem:

2's complement : The signed binary numbers required too much electronic circuitry for addition and subtraction.

Therefore, positive decimal numbers are expressed in sign-magnitude form but negative decimal numbers are expressed in 2's complement.

---

Double precision numbers :-

→ Most present day computers are 16 bit. In these computers the numbers from +32,767 to −32,768 can be stored in each register.

→ To store numbers greater than these numbers, double precision system is used.

→ In this method two storage locations are used to represent each number.

The format is

| | | |
|---|---|---|
| First word | S | High order bits |
| Second word | 0 | Low order bits |

where 'S' is the sign bit and '0' is a zero. Thus numbers with 31 bit length can be represented in 16 bit registers.

→ For still bigger numbers triple precision can be used. In triple precision 3 word lengths (each 16 bit) is used to represent each number

## Floating point Numbers :-

A 16-bit computer cannot store a positive number larger than 32767. What if we want to handle a fractional number like 35.7812 or a large number like 987654321 ?

Such numbers are stored and processed in what is known as exponential form. These numbers have an embedded decimal point and are called floating point numbers or Real numbers.

For example:

35.7812 can be expressed $0.357812 \times 10^2$.

Similarly, the number 987654321 can be expressed as $0.987654 \times 10^9$.

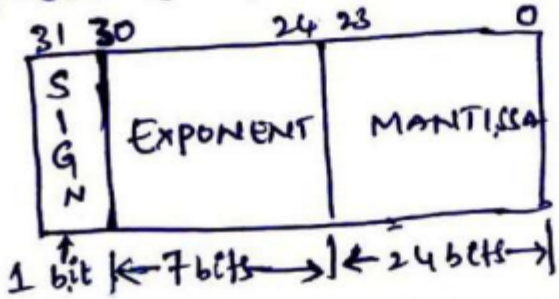
By writing a large number in exponential form, we lose some digits.

If 'x' is a real number, its floating point form representation is

$$x = f \times 10^E.$$

where the number f is called mantissa and E is the exponent.

Floating point numbers are stored differently. The entire memory location is divided into three fields (or) parts as shown below.




Floating point representation.

The first part (1 bit) is reserved for the sign, the second part (7 bits) for the exponent of the number, and the third (24 bits) for the mantissa of the number.

Typically, floating numbers use a field width of 32 bits where 24 bits are used for the mantissa and 7 bits for the exponent.

Thus, we can represent very small fractions or very large numbers within the computer using the floating point representation.

**for example:**

Here the mantissa has a 10 bit length and exponent has 6 bit length.

| Mantissa | Exponent |
|---|---|
| 0111001101 | 100111 |

Floating point format fig (1)

→ The left most bit of mantissa is sign bit.

→ The binary point is to the right of this sign bit.

→ The 6-bit exponent has a base of 2. The exponent can represent 0 to 63.

→ To express negative exponents the number $(32)_{10}$ (ie $(100000)_2$) has been added to the exponent. It is known as excess -32 format notation and is a common floating point format.

Examples of exponent in excess -32 format are (20)

| Actual exponent | Binary representation in excess-32 format |
|---|---|
| -32 | 000000 |
| -1 | 011111 |
| 0 | 100000 |
| +7 | 100111 |
| +15 | 101111 |
| +31 | 111111 |

The number represented in fig (1) is

Mantissa $+0.111001101$

Exponent $100111$.

Subtracting 100000 from exponent, we get

$0001.11$.

The number is $0.111001101 \times 2^{7}$

$= (1110011.01)_2$

$= (115.25)_{10}$.

→ What does the floating point number $0110100000|010101$ represent.

Sol$^n$: Mantissa is $+0.110100000$

Exponent is $010101$.

Subtracting 100000 from exponent, we get

$110101$.

The given number is $+0.110100000 \times 2^{-11}$.

$= (0.00000000000110100000)_2$

$= (+0.000396728)_{10}$

## Octal-to-Decimal conversions:

Any octal number can be converted into its equivalent decimal number by using the weights assigned to each octal digit position as given in Table-1. (pg.2 after).

→ For example:

Convert the octal number 1054 into decimal number.

Soln: 1054 octal number.
$8^3\ 8^2\ 8^1\ 8^0$ weights

$$\therefore (1054)_8 = 1\times 8^3 + 0\times 8^2 + 5\times 8 + 4\times 8^0$$
$$= 512 + 0 + 40 + 4$$
$$= (556)_{10}.$$

HW → Convert the octal number $(1502)_8$ into decimal number

→ Convert $(6327.4051)_8$ into its equivalent decimal number.

Soln: Using the weights given in Table-1, we obtain

$$(6327.4051)_8 = 6\times 8^3 + 3\times 8^2 + 2\times 8^1 + 7\times 8^0 + 4\times 8^{-1} + 0\times 8^{-2} + 5\times 8^{-3} + 1\times 8^{-4}$$
$$= 3072 + 192 + 16 + 7 + \frac{4}{8} + 0 + \frac{5}{512} + \frac{1}{4096}$$
$$= (3287.5100098)_{10}.$$

$$\therefore (6327.4051)_8 = (3287.5100098)_{10}.$$

## Decimal-to-octal conversion:

The conversion from decimal to octal (base-10 to base-8) is similar to the conversion procedure for base-10 to base-2 conversion.

The only difference is that the number 8 is used

in place of 2 for division in the case of integers, and for multiplication in the case of fractional numbers.

Examples:

→ Convert decimal number 574 into octal.

Soln:

| | |
|---|---|
| 8 | 574 |
| 8 | 71 remainder 6 |
| 8 | 8 remainder 7 |
| 8 | 1 remainder 0 |
| | 0 remainder 1 |

$\therefore (574)_{10} = (1076)_8$.

→ (a) convert $(247)_{10}$ into octal

(b) Convert $(0.6875)_{10}$ into octal

(c) convert $(468)_2$ into octal.

(d) convert $(3287.5100098)_{10}$ into octal

Soln: (b)

| Fraction | Fraction×8 | Remainder new fraction | Integer |
|---|---|---|---|
| 0.6875 | 5.5000 | 0.5000 | 5 |
| 0.5000 | 4.0000 | 0.0000 | 4 |

$\therefore (0.6875)_{10} = (.54)_8$

(d) Integer part:

| | |
|---|---|
| 8 | 3287 |
| 8 | 410 remainder 7 |
| 8 | 51 remainder 2 |
| 8 | 6 remainder 3 |
| | 0 remainder 6 |

$\therefore (3287)_{10} = (6327)_8$

Fractional part :

| Fraction | fraction × 8 | Remainder new fraction | Integer. |
|---|---|---|---|
| 0.5100098 | 4.0800784 | 0.0800784 | 4 |
| 0.0800784 | 0.6406272 | 0.6406272 | 0 |
| 0.6406272 | 5.1250176 | 0.1250176 | 5 |
| 0.1250176 | 1.0001408 | 0.0001408 | 1 |

Thus $(0.5100098)_{10} \simeq (0.4051)_8$

$\therefore (3287.5100098)_{10} = (6327.4051)_8$

Note: Conversion for fractional numbers may not be exact.
In general, an approximate equivalent can be determined by terminating the process of multiplication by eight at the desired point.

Octal-to-Binary conversion:
Octal numbers can be converted into equivalent binary numbers by replacing each octal digit by its 3-bit equivalent binary.

Forexample: convert $(71)_8$ into an equivalent binary number.

octal number 7 1
↓ ↓
Binary number 111 001
i.e., binary equivalents of 7 and 1 are 111 and 001 respectively
$\therefore (71)_8 = (111001)_2$

→ Convert $(736)_8$ into an equivalent binary number

Soln: 7 3 6 octal number
↓ ↓ ↓
111 011 110 Binary number
i.e., the binary equivalents of 7, 3 and 6 are 111
011 and 110 respectively
$\therefore (736)_8 = (111\ 011\ 110)_2$

The following table gives octal number and their binary equivalents for decimal numbers 0 to 15.

Binary and decimal equivalents of octal numbers

| Octal | Decimal | Binary |
|---|---|---|
| 0 | 0 | 000 |
| 1 | 1 | 001 |
| 2 | 2 | 010 |
| 3 | 3 | 011 |
| 4 | 4 | 100 |
| 5 | 5 | 101 |
| 6 | 6 | 110 |
| 7 | 7 | 110 |
| 10 | 8 | 001000 |
| 11 | 9 | 001001 |
| 12 | 10 | 001010 |
| 13 | 11 | 001011 |
| 14 | 12 | 001100 |
| 15 | 13 | 001101 |
| 16 | 14 | 001110 |
| 17 | 15 | 001111 |

## Binary-to-octal conversion:

Binary numbers can be converted into equivalent octal numbers by making groups of three bits starting from LSB and moving towards MSB for integer part of the number and then replacing each group of three bits by its octal representation.

For fractional part, the groupings of three bits are made starting from the binary point.

→ convert $(1001110)_2$ to its octal equivalent.

Sol$^n$: $(1001110)_2 = (\underbrace{001}_{1}\ \underbrace{001}_{1}\ \underbrace{110}_{6})_2$

$= (1\ 1\ 6)_8$

$= (116)_8$.

→ convert the following binary numbers into octal numbers.

(i) 110101 (ii) 010100110 (iii) 111000101 (iv) 1001101.1011

(v) 1100111 0001. 0001011 11001

(vi) 10110111 10. 11001010011

(vii) 111110001. 10011001101.

Sol$^n$:

(i) $(110101)_2 = (\underbrace{110}_{6}\ \underbrace{101}_{5})_2$

$= (65)_8$

(ii) $(0.10100110)_2 = (0.\underbrace{101}_{5}\ \underbrace{001}_{1}\ \underbrace{100}_{4})_2$

$= (0.514)_8$

(iv) $(1001101.1011)_2 = (\underbrace{001}_{1}\ \underbrace{001}_{1}\ \underbrace{101}_{5}\ \underbrace{101}_{5}\ \underbrace{100}_{4})_2$

$= (115.54)_8$

From the above examples we observe that in forming the 3-bit groupings, 0's may be required to complete the first (most significant digit) group in the integer part and the last (least significant digit) group in the fractional part

## Hexadecimal to Decimal Conversion:

- One method to convert a hexadecimal number into its decimal equivalent is to first convert hexadecimal to binary and then convert binary to decimal.
- A direct conversion of hexadecimal into decimal is also possible.

Since the base of a hexadecimal is 16, the weights of different bits are $16^0, 16^1, 16^2$ etc. starting with the bit on the extreme right.

The decimal equivalent of a hexadecimal number equals the sum of all digits multiplied by their weights.

For example:

| E | 7 | F | 6 | Hexadecimal |
|---|---|---|---|---|
| $16^3$ | $16^2$ | $16^1$ | $16^0$ | weights |

$$E7F6 = (E \times 16^3) + (7 \times 16^2) + (F \times 16^1) + (6 \times 16^0)$$
$$= (14 \times 4096) + (7 \times 256) + (15 \times 16) + (6 \times 1)$$
$$= 57344 + 1792 + 240 + 6$$
$$= (59382)_{10}.$$

→ Obtain decimal equivalent of hexadecimal number $(3A.2F)_{16}$.

Sol: $(3A.2F)_{16} = (3 \times 16^1) + (A \times 16^0) + (2 \times 16^{-1}) + (F \times 16^{-2})$

$$= 48 + (10 \times 1) + \frac{2}{16} + \frac{15}{16^2}$$
$$= (58.1836)_{10}.$$

Note: The fractional part may not be an exact equivalent, may give a small error.

→ Convert the hexadecimal number $.8A3_{16}$ into decimal equivalent (a) by first converting into binary (b) directly.

→ Decimal-to-Hexadecimal Conversion:

To convert the decimal to binary

and then convert binary to hexadecimal.

- The direct method is successive division by 16 and to write the hexadecimal equivalents of remainder.

For example:

→ Convert decimal number 5390 into hexadecimal

| | |
|---|---|
| 16 \| 5390 | |
| 16 \| 336 | remainder E |
| 16 \| 21 | remainder 0 |
| 16 \| 1 | remainder 5 |
| \| 0 | remainder 1 |

↑

The hexadecimal equivalent is 150E.

---

→ Convert the following decimal numbers into hexadecimal numbers.

(a) 95.5 (b) 675.625 (c) 268 (d) 5741

Sol: (a) Integral part:

| | |
|---|---|
| 16 \| 95 | |
| 16 \| 5 | remainder 15 = F |
| \| 0 | remainder 5 |

↑

i.e; $(95)_{10} = (5F)_{16}$.

Fractional part:

| Fraction | Fraction × 16 | Remainder, new fraction | Integer |
|---|---|---|---|
| 0.5 | 8.0 | 0.0 | 8 |
| 0.0 | 0.0 | 0.0 | 0 |

i.e; $(0.5)_{10} = (0.8)_{16}$.

$\therefore (95.5)_{10} = (5F.8)_{16}$

---

## Binary and decimal equivalents of hexadecimal numbers:

| Hexa decimal | Decimal | 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 |
| A | 10 | 1010 |
| B | 11 | 1011 |
| C | 12 | 1100 |
| D | 13 | 1101 |
| E | 14 | 1110 |
| F | 15 | 1111 |

## Hexadecimal to Binary conversion:-

Hexadecimal numbers can be converted into equivalent binary numbers by replacing each hex digit by its equivalent 4-bit binary number.

**Example:** Convert $(2F9A)_{16}$ to equivalent binary number.

**Sol$^n$:** Using the above table, write the binary equivalent of each hex digit.

$$(2F9A)_{16} = (0010\ 1111\ 1001\ 1010)_2$$
$$= (0010111110011010)_2$$

## Binary-to-hexadecimal Conversions:-

Binary numbers can be converted into the equivalent hexadecimal numbers by making groups of four bits starting from LSB and moving towards MSB for integer part and then replacing each group of four bits by its hexadecimal representation.

For the fractional part, the above procedure is repeated starting from the bit next to the binary point. and moving towards the right.

→ convert the following binary numbers to their equivalent hex numbers.

(a) 1010011010111 (b) 0.0001111010111101.

(c) 11001110001.000101111001 (d) 1011011110.1100101001

(a) $(1010011010111)_2 = \underset{2}{0010}\ \underset{9}{1001}\ \underset{A}{1010}\ \underset{F}{1111}$

$= (29AF)_{16}$.

(b) $(0.0001111010111101)_2 = 0.\underset{1}{0001}\ \underset{E}{1110}\ \underset{B}{1011}\ \underset{4}{0100}$

$= (0.1EB4)_{16}$

(c) $(11001110001.000101111001)_2$

$= \underset{6}{0110}\ \underset{7}{0111}\ \underset{1}{0001} . \underset{1}{0001}\ \underset{7}{0111}\ \underset{9}{1001}$

$= (671.179)_{16}$.

From the above examples, we observe that in forming 4-bit groupings. 0's may be required to complete the first (MSB) group in the integer part and the (last) (LSB) group in the fractional part.

## Conversion from Hex to octal and vice-versa

Hexadecimal numbers can be converted to equivalent octal numbers and octal numbers can be converted to equivalent hex numbers by converting the hex/octal number to equivalent binary and then to octal/hex, respectively.

→ convert the following hex numbers to octal numbers.

(a) $(A72E)_{16}$ (b) $(0.BF85)_{16}$

Soln: (a) $(A72E)_{16} = (1010\ 0111\ 0010\ 1110)_2$

$= (\underbrace{001}_{1}\ \underbrace{010}_{2}\ \underbrace{011}_{3}\ \underbrace{100}_{4}\ \underbrace{101}_{5}\ \underbrace{110}_{6})_2$

$= (123456)_8$

(b) $(0.BF85)_{16} = (0.1011\ 1111\ 1000\ 0101)_2$

$= (0.\underbrace{101}_{5}\ \underbrace{111}_{7}\ \underbrace{111}_{7}\ \underbrace{000}_{0}\ \underbrace{010}_{2}\ \underbrace{100}_{4})_2$

$= (0.577024)_8$

→ convert $(247.36)_8$ to equivalent hex number.

Soln: $(247.36)_8 = (010\ 100\ 111.011\ 110)_2$

$= (0\ 1010\ 0111.0111\ 1000)_2$

$= (A7.78)_{16}$