# Number System Manual for Students

| Decimal Number | Octal Number | Hexadecimal Number | Binary Number Representation |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 01 |
| 3 | 3 | 3 | 11 |
| 4 | 4 | 4 | 100 |
| 5 | 5 | 5 | 101 |
| 6 | 6 | 6 | 110 |
| 7 | 7 | 7 | 111 |
| 8 | 10 | 8 | 1000 |
| 9 | 11 | 9 | 1001 |
| 10 | 12 | A | 1010 |
| 11 | 13 | B | 1011 |
| 12 | 14 | C | 1100 |
| 13 | 15 | D | 1101 |
| 14 | 16 | E | 1110 |
| 15 | 17 | F | 1111 |

**(Ravi K. Walia)**
**Assistant Professor & Incharge**
**Computer & Instrumentation Centre**
**Dr. Y. S. Parmar University of Horticulture & Forestry,**
**Nauni Solan INDIA (HP)**

# PREFACE

This document has been prepared for students at Dr. Y. S. Parmar University of Horticulture & Forestry, Nauni, Solan (HP) India. Most of the introductory courses in computer science and information technology include topic on number system and students face problems in working with number system.

The technique to represent and work with numbers is called number system. Decimal number system is the most common number system, other popular number systems include binary number system, octal number system and hexadecimal number system. This manual cover different number systems, conversion of number from one representation to other and mathematical computation using different numbers system as well as 1’s and 2’s complement notation for binary number. The best way to absorb the material is to try to solve different problems.

## Decimal Numbers

**Base** : 10

**Digits used** : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Each position represents a specific power of the base 10

In the decimal number system, there are ten possible values that can appear in each digit position, and so there are ten numerals required to represent the quantity in each digit position. The decimal numerals are the familiar zero through nine (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). When we write decimal (base 10) numbers, we use a positional notation system. Each digit is multiplied by an appropriate power of 10 depending on its position in the number: (from decimal point to the left we start with power of 10 as 0, 1, 2, 3 (i.e. $10^0$, $10^1$, $10^2$, $10^3$) and so on and from decimal point to right we start with power of 10 as -1, -2, -3 (ie.$10^{-1}$, $10^{-2}$, $10^{-3}$) and so on

**For example**:

$843 = 8 \times 10^2 + 4 \times 10^1 + 3 \times 10^0 = 8 \times 100 + 4 \times 10 + 3 \times 1 = 800+40+3=843$

$.25 = 2 \times 10^{-1} + 5 \times 10^{-2} = 2/10 + 5/100 = .2 + .05 = .25$

In a positional notation system, the number base is called the radix. Thus, the base ten system that we normally use has a radix of 10. The term radix and base can be used interchangeably. When writing numbers in a radix other than ten, or where the radix isn't clear from the context, it is customary to specify the radix using a subscript.

# Binary Numbers

Base : 2

Digits used : 0, 1

Each position represents a specific power of the base 2

The binary number system is also a positional notation numbering system with base 2. Each digit position in a binary number represents a power of two.

### *Conversion from Binary to Decimal number:*

Each binary digit is multiplied by an appropriate power of 2 based on the position in the number: (from binary point to the left we start with power of 2 as 0, 1, 2, 3 and so on and from decimal point to right we start with power of 2 as -1, -2, -3 and so on)

For example:

$101101_2 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= 1 \times 32 + 0 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$

$= 32 + 8 + 4 + 1 = 45$

$.1101_2 = 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}$

$= 1/2 + 1/4 + 0/8 + 1/16 = .5 + .25 + .0625 = .8125$

$101101.1101_2 = 45.8125$

## Octal Numbers

Base : 8

Digits used : 0, 1, 2, 3, 4, 5, 6, 7

Each position represents a specific power of the base 8

The octal number system is also a positional notation numbering system with base 8. Each digit position in octal number represents a power of 8.

### ***Conversion from Octal to Decimal number:***

Each octal digit is multiplied by an appropriate power of 8 based on the position in the number: (from octal point to the left we start with power of 8 as 0, 1, 2, 3 and so on and from octal point to right we start with power of 8 as -1, -2, -3 and so on)

For example:

$2305_8 = 2 \text{ x } 8^3 + 3 \text{ x } 8^2 + 0 \text{ x } 8^1 + 5 \text{ x } 8^0$

$= 2 \text{ x } 512 + 3 \text{ x } 64 + 0 \text{ x } 8 + 5 \text{ x } 1$

$= 1024 + 192 + 0 + 5 = 1221$

$.423_8 = 4 \text{ x } 8^{-1} + 2 \text{ x } 8^{-2} + 3 \text{ x } 8^{-3}$

$= 4/8 + 2/64 + 3/512$

$= .5 + .03125 + .005859375 = .537109375$

| Decimal Number | Binary Number Representation | Explanation Binary Number Representation | | | | |
|---|---|---|---|---|---|---|
| | | | $2^3 = 8$ | $2^2 = 4$ | $2^1 = 2$ | $2^0 = 1$ |
| 0 | 0 | | | | | 0 |
| 1 | 1 | | | | | 1 |
| 2 | 01 | | | | 1 | 0 |
| 3 | 11 | 2=2+1 | | | 1 | 1 |
| 4 | 100 | | | 1 | 0 | 0 |
| 5 | 101 | 5=4+1 | | 1 | 0 | 1 |
| 6 | 110 | 6=4+2 | | 1 | 1 | 0 |
| 7 | 111 | 7=4+2+1 | | 1 | 1 | 1 |

## Hexadecimal Numbers

Base : 16

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

(Where A=10, B=11, C=12, D=13, E=14 and F=15)

Each position represents a specific power of the base 16

The hexadecimal number system is also a positional notation numbering system with base 16. Each digit position in a binary number represents a power of 16.

### *Conversion from Hexadecimal to Decimal number*:

Each hexadecimal digit is multiplied by an appropriate power of 16 based on the position in the number: (from hexadecimal point to the left we start with power of 16 as 0, 1, 2, 3, 4 and so on and from hexadecimal point to right we start with power of 16 as -1, -2, -3, -4 and so on)

For example:

$2B5_{16} = 2 \times 16^2 + B \times 16^1 + 5 \times 16^0$

$= 2 \times 256 + 11 \times 16 + 5 \times 1$

$= 512 + 176 + 5 = 693$

$.8C_{16} = 8 \times 16^{-1} + C \times 16^{-2}$

$= 8/16 + 12/256$

$= .5 + .046875 = .546875$

$2B5.8C_{16} = 693.546875$

## *Number System table*

| Decimal Number | Octal Number | Hexadecimal Number | Binary Number Representation | Explanation<br>Binary Number Representation | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | $2^3$ =8 | $2^2$ =4 | $2^1$=2 | $2^0$ =1 |
| 0 | 0 | 0 | 0 | | | | | 0 |
| 1 | 1 | 1 | 1 | | | | | 1 |
| 2 | 2 | 2 | 01 | | | | 1 | 0 |
| 3 | 3 | 3 | 11 | 2=2+1 | | | 1 | 1 |
| 4 | 4 | 4 | 100 | | | 1 | 0 | 0 |
| 5 | 5 | 5 | 101 | 5=4+1 | | 1 | 0 | 1 |
| 6 | 6 | 6 | 110 | 6=4+2 | | 1 | 1 | 0 |
| 7 | 7 | 7 | 111 | 7=4+2+1 | | 1 | 1 | 1 |
| 8 | 10 | 8 | 1000 | 8=8+0 | 1 | 0 | 0 | 0 |
| 9 | 11 | 9 | 1001 | 9=8+1 | 1 | 0 | 0 | 1 |
| 10 | 12 | A | 1010 | 10=8+2 | 1 | 0 | 1 | 0 |
| 11 | 13 | B | 1011 | 11=8+2+1 | 1 | 0 | 1 | 1 |
| 12 | 14 | C | 1100 | 12=8+4 | 1 | 1 | 0 | 0 |
| 13 | 15 | D | 1101 | 13=8+4+1 | 1 | 1 | 0 | 1 |
| 14 | 16 | E | 1110 | 14=8+4+2 | 1 | 1 | 1 | 0 |
| 15 | 17 | F | 1111 | 15=8+4+2+1 | 1 | 1 | 1 | 1 |

# Conversion of Binary, Octal & Hexadecimal Numbers

## *From Binary to Octal*

Starting at the binary point and working left, separate the bits into groups of **three** and replace each group with the corresponding **octal** digit and similarly for binary fraction starting at binary point and working right, separate the bits into group of **three** and replace each group with the corresponding **octal** digit (add additional zeros to the left of integer portion or right of fraction portion if required)

$10001011.10101_2 =$ 010 001 011. 101 010 $= 213.52_8$

## *From Binary to Hexadecimal*

Starting at the binary point and working left, separate the bits into groups of **four** and replace each group with the corresponding **hexadecimal** digit and similarly for binary fraction starting at binary point and working right, separate the bits into group of **four** and replace each group with the corresponding **hexadecimal** digit ( add additional zeros to the left of integer portion or right of fraction portion if required).

$10001011.110001_2 =$ 1000 1011. 1100 0100 $= 8D.C4_{16}$

## *From Octal to Binary*

Replace each **octal** digit with the corresponding **3-bit** binary string.

$213.24_8 =$ 010 001 011.010 100 $= 10001011.010100_2$

### *From Hexadecimal to Binary*

Replace each **hexadecimal** digit with the corresponding **4-bit** binary string.

$8B.A5_{16}$ = 1000 1011. 1010 0101 = $10001011.10100101_2$

## Conversion of Decimal Numbers

### *From Decimal to Binary*

### *For Integer part conversion*

Divide number by 2; keep track of remainder; repeat with dividend equal to quotient until zero; first remainder is binary LSB and last is MSB.

e.g. to convert 139 to binary

| Dividend | Quotient | Remainder |
|---|---|---|
| 139/2 | 69 | 1 LSB |
| 69/2 | 34 | 1 |
| 34/2 | 17 | 0 |
| 17/2 | 8 | 1 |
| 8/2 | 4 | 0 |
| 4/2 | 2 | 0 |
| 2/2 | 1 | 0 |
| 1/2 | 0 | 1 MSB |

139= 10001011

The conversion from decimal integers to any base-r system is same to the above example, except that division is done by r instead of 2**.**

## *For Fraction part conversion*

Multiply the fraction by two; keep track of integer part; repeat with multiplier equal to product fraction until the fraction part is zero; first integer is MSB , last is the LSB ( conversion may not be exact, a repeated fraction). The conversion from decimal fraction to any base-r system is similar except the multiplication is done by r instead of 2.

Convert (0.7854) 10 to binary.

| | |
|---|---|
| 0.7854<br>x 2 | Integer part |
| 1.5708 | 1 MSB |
| 0.5708<br>x 2 | |
| 1.1416 | 1 |
| 0.1416<br>x 2 | |
| 0.2832 | 0 |
| 0.2832<br>x 2 | |
| 0.5664 | 0 LSB |

$0.7854 = (0.1100)_2$

## *From Decimal to Octal*

## *For Integer part conversion*

Divide number by 8; keep track of remainder; repeat with dividend equal to quotient until zero; first remainder is binary LSB and last is MSB. e.g. to convert decimal 846 to octal

| Dividend | Quotient | Remainder |
|---|---|---|
| 846/8 | 105 | 6 LSB |
| 105/8 | 13 | 1 |
| 13/8 | 1 | 5 |
| 1/8 | 0 | 1 MSB |

846 = $1516_8$

## *For Fraction part conversion*

Multiply the fraction by eight; keep track of integer part; repeat with multiplier equal to product fraction; first integer is MSB , last is the LSB; conversion may not be exact; a repeated fraction. Convert $(0.44140625)_{10}$ to octal.

| 0.44140625<br>x 8 | Integer part |
|---|---|
| 3.53125 | 3 MSB |
| 0.53125<br>x 8 | |
| 4.25 | 4 |
| 0.25<br>x 8 | |
| 2.00 | 2 LSB |

0.44140625 = $(0.342)_2$

## *From Decimal To Hexadecimal*

## *For Integer part conversion*

Divide number by 16; keep track of remainder; repeat with dividend equal to quotient until zero; first remainder is binary LSB and last is MSB. e.g. to convert 2619 to Hexadecimal

| Dividend | Quotient | Remainder |
|---|---|---|
| 2619/16 | 163 | 11 (Hexa B) LSB |
| 163/16 | 10 | 3 |
| 10/16 | 0 | 10 (in Hexa A) |
| 1/8 | 0 | 1 MSB |

$2619 = A3B_{16}$

## *For Fraction part conversion*

Multiply the fraction by sixteen; keep track of integer part; repeat with multiplier equal to product fraction; first integer is MSB , last is the LSB; conversion may not be exact; a repeated fraction. Convert (0.609375) to hexadecimal.

| | |
|---|---|
| 0.609375<br>x 16 | Integer part |
| 9.75 | 9 MSB |
| 0.75<br>x 16 | |
| 12.00 | 12 (C in Hexa) LSB |

$0.609375 = (0.9C)_{16}$

## Binary Number System

### Addition

0 + 0 = 1

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 plus a carry of 1 to next higher bit/column

```
  100111
 + 11011
------------
 1000010
------------
```

### Subtraction

0 - 0 = 0

0 - 1 = 1 with borrow from the next higher digit/coulm

1 - 0 = 1

1 - 1 = 0

```
   10101           1011100    (92)
  -01110          -0111000    (-56)
------------      --------------
   00111           0100100    (36)
------------      -------------
```

## Multiplication in Binary Number

Multiplicand 1000 Multiplier 1001

```
   1000
 x 1001
 --------------
   1000
  0000
 0000
1000
----------------
Product 1001000
```

## Division in Binary Number

Divisor 1000 Dividend 1001010

```
        1001_____(Quotient)
 1000  | 1001010
        -1000
        ----------
            10
            101
            1010
           -1000
         -------------
            10  Remainder
```

## Complements of Number:

There are two types of complements for a number of base (also called radix) r, called r's complement and (r- 1)'s complement. For decimal numbers there are 10's and 9's complement. For binary numbers there are 2's and 1's complements

**9's complement:** The 9's complement is obtained by subtracting each digit of the number from 9 (the highest digit value).

9's complement of 256

```
    9  9  9  9
  - 0 -2 -5 -6
 ---------------
   9  7  4   3
```

**10's complement:** Adding 1 in the 9's complement produces the 10's complement.

10's complement of 0256 = 9743+1 = 9744

## Binary Arithmetic

1's complement for a binary number is obtained by subtracting each binary digit of the number from the digit 1.

**1's complement:** The 1's complement is obtained by changing binary digit 0 to 1 and 1 to 0. The 1's complement of 1010 is 0101

| | |
|---|---|
| Number is | 1 0 1 0 |
| 1's complement is | 0 1 0 1 |

**2's complement**: 2's complement of binary number is obtained by adding 1 to 1's complement

```
Number is                 1 0 1 0
1's complement  is        0 1 0 1
2's complement is         0 1 0 1
                        +       1
                        ---------------
                          0 1 1 0
                        ----------------
```

The 2's complement can also be obtained by not complementing the least significant zeros till the first 1 is encountered, this 1 is also not complemented. After this 1 the rest of all the bits are complemented on the left.

## Addition and Subtraction using 1's complement

For addition and subtraction, in 1's complement the negative number is represented in 1's complement notation and both numbers are added, if 1 bit is carried out of sign bit it is added to the result.

For addition and subtraction, in 2's complement the negative number is represented in 2's complement notation and both numbers are added, if 1 bit is carried out of sign bit it is discarded

For the purpose of all example assume register size is 8 bit (1 position for sign bit and 7 position for magnitude)

| Number | 1's complement representation | 2's complement representation |
|---|---|---|
| +5 | 0 000 0101 | 0 000 0101 |
| -5 | 1 111 1010 | 1 111 1011 |
| +25 | 0 001 1001 | 0 001 1001 |
| -25 | 1 110 0110 | 1 110 0111 |
| +30 | 0 001 1110 | 0 001 1110 |
| -30 | 1 110 0001 | 1 110 0010 |
| +55 | 0 011 0111 | 0 011 0111 |
| -55 | 1 100 1000 | 1 100 1001 |

## 1's Complements Notations

25 + 35

| Decimal Number | Carry bit | Sign Bit | 1's complement notation | |
|---|---|---|---|---|
| +25<br>+30 | | 0<br>0 | 001 1001<br>001 1110 | |
| +55 | | 0 | 011 0111 | |

25 - 30

| Decimal Number | Carry bit | Sign Bit | 1's complement notation | |
|---|---|---|---|---|
| +25<br>-30 | | 0<br>1 | 001 1001<br>110 0001 | |
| -5 | | 1 | 111 1010 | Which is 1's complement of -5 |

-25 + 30

| Decimal Number | Carry bit | Sign Bit | 1's complement notation | |
|---|---|---|---|---|
| -25<br>+30 | | 1<br>0 | 110 0110<br>001 1110 | |
| | 1 | 0 | 000 0100<br>+ 1 | Add carry bit |
| +5 | | 0 | 000 0101 | |

-25 - 30

| Decimal Number | Carry bit | Sign Bit | 1's complement notation | |
|---|---|---|---|---|
| -25<br>-30 | | 1<br>1 | 110 0110<br>110 0001 | |
| | 1 | 1 | 100 0111<br>+ 1 | Add carry bit |
| -55 | | 1 | 100 1000 | Which is 1's complement of -55 |

## 2’s Complements Notations

+25 + 30

| Decimal Number | Carry bit | Sign Bit | 2’s complement notation | |
|---|---|---|---|---|
| +25<br>+30 | | 0<br>0 | 001 1001<br>001 1110 | |
| +55 | | 0 | 011 0111 | |

+25 - 30

| Decimal Number | Carry bit | Sign Bit | 2’s complement notation | |
|---|---|---|---|---|
| +25<br>-30 | | 0<br>1 | 001 1001<br>110 0010 | |
| -5 | | 1 | 111 1011 | Which is 2’s complement of -5 |

-25 + 30

| Decimal Number | Carry bit | Sign Bit | 2’s complement notation | |
|---|---|---|---|---|
| -25<br>+30 | | 1<br>0 | 110 0111<br>001 1110 | |
| | 1 | 0 | 000 0101 | Discard the carry bit |
| +5 | | 0 | 000 0101 | |

-25 - 30

| Decimal Number | Carry bit | Sign Bit | 2’s complement notation | |
|---|---|---|---|---|
| -25<br>-30 | | 1<br>1 | 110 0111<br>110 0010 | |
| | 1 | 1 | 100 1001 | Discard the carry bit |
| -55 | | 1 | 100 1001 | Which is 2’s complement of -55 |

If the **carry into the sign bit is not equal to the carry out of the sign bit** then overflow must have occurred.

| Decimal Number | Carry bit | Sign Bit | 2's complement notation |
|---|---|---|---|
| +65<br>+75 | | 0<br>0 | 100 0001<br>100 1011 |
| +140 | | 1 | 000 1100 |

The expected result is +140 but the binary sum is a negative number and is equal to –116, which obviously is a wrong result. This has occurred because of overflow.

## Signed 2's is preferred over signed 1's notation

Signed 2's complement there is just one zero and there is no positive or negative zero.

## Data Representation

+0 in 2's Complement Notation: 0 000 0000

-0 in 1's complement notation: 1 111 1111

Add 1 for 2's complement: 1

Discard the Carry Out 1 0 000 0000

For signed 1's complement representation in 8 bit register range is

$= (2^7 - 1)$ to $-(2^7 - 1)$

$= (128–1)$ to $-(128–1)$

= 127 to –127

But, for signed 2's complement we can represent +127 to –128.

The – 128 is represented in signed 2's complement notation as 10000000