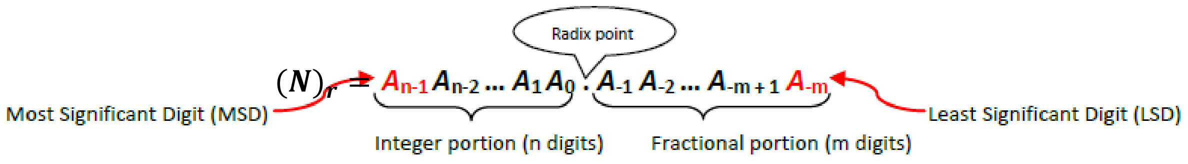
Unit-2
Number Systems

Unit-2
Number Systems

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 in general has two parts- integer and fractional. Set apart by a radix point (.), i.e.


Radix point
$(N)_r = A_{n-1} A_{n-2} \ldots A_1 A_0 . A_{-1} A_{-2} \ldots A_{-m+1} A_{-m}$
Most Significant Digit (MSD)
Least Significant Digit (LSD)
Integer portion (n digits)
Fractional portion (m digits)

$\boldsymbol{N}$ = a number and $\boldsymbol{r}$ = radix or base of number system

In general, a number expressed in base-r system has coefficients multiplied by powers of r:

$$A_{n-1}r^{n-1} + A_{n-2}r^{n-2} + \cdots + A_1r^1 + A_0r^0 + A_{-1}r^{-1} + A_{-2}r^{-2} + \cdots + A_{-m}r^{-m}$$

For example, $(3456.54)_{10}$ can be written as:

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

➢ There are mainly four number system which are used in digital electronics platform.

1. ***Decimal number system:***
   - The decimal number system contains ten unique digits from 0 to 9.
   - The base or radix is 10.

2. ***Binary number system:***
   - The binary number system contains two unique digits 0 and 1.
   - The base or radix is 2.

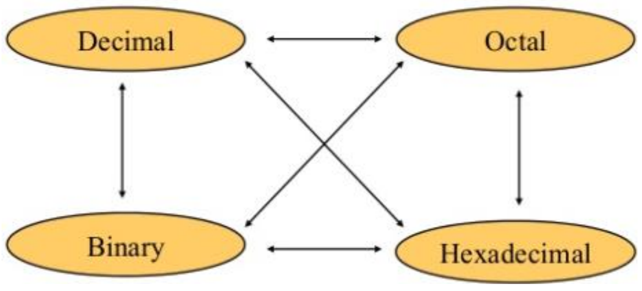
3. ***Octal number system:***
   - The octal number system contains eight unique digits from 0 to 7.
   - The base or radix is 8.

4. ***Hexadecimal number system:***
   - The hexadecimal number system contains sixteen unique digits: 0 to 9 and six letters A, B, C, D, E and F.
   - The base or radix is 16.

DECIMAL (BASE 10)	BINARY (BASE 2)	OCTAL (BASE 8)	HEXADECIMAL (BASE 16)
0	00000	0	0
1	00001	1	1
2	00010	2	2
3	00011	3	3
4	00100	4	4
5	00101	5	5
6	00110	6	6
7	00111	7	7
8	01000	10	8
9	01001	11	9
10	01010	12	A
11	01011	13	B
12	01100	14	C
13	01101	15	D
14	01110	16	E
15	01111	17	F
16	10000	20	10
Examples			
255	11111111	377	FF
256	100000000	400	100

Number Base Conversion

The possibilities:



Decimal to Other Base System

The decimal number can be an integer or floating-point integer. When the decimal number is a floating-point integer, then we convert both part (integer and fractional) of the decimal number in the isolated form (individually). There are the following steps that are used to convert the decimal number into a similar number of any base $'r'$.

Conversion of Integer part:

- Divide the given integer part of decimal number by base $'\boldsymbol{r}'$ successively and write down all the remainders till the quotient is zero.
- Write all the remainders starting with the MSB (Most Significant Bit) i.e. from bottom to LSB (Least Significant Bit) i.e. top.

Conversion of Fractional part:

- Multiply the given fractional part of decimal number by base $'\boldsymbol{r}'$ successively until the fractional part becomes zero.
- Note down the integer part starting from first.
  Note: If fractional part does not become zero then, result has been taken up to 6 places.

➢ Decimal to Binary Conversion

1. ***Convert*** $\mathbf{(51)_{10}}$ ***and*** $\mathbf{(152)_{10}}$ ***into binary.***

2	51	
2	25	1 (LSB)
2	12	1
2	6	0
2	3	0
2	1	1
	0	1 (MSB)

$(51)_{10} = (110011)_2$

2	152	
2	76	0 (LSB)
2	38	0
2	19	0
2	9	1
2	4	1
2	2	0
2	1	0
	0	1 (MSB)

$(152)_{10} = (10011000)_2$

2. ***Convert*** $\mathbf{(41.6875)_{10}}$ ***into binary***.

Conversion of integer part:

2	41	
2	20	1 (LSB)
2	10	0
2	5	0
2	2	1
2	1	0
	0	1 (MSB)

$(41)_{10} = (101001)_2$

Conversion of fractional part:

	integer	fraction
0.6875×2	1	0.3750
0.3750×2	0	0.7500
0.7500×2	1	0.5000
0.5000×2	1	0.0000

$(0.6875)_{10} = (0.1011)_2$

$\therefore \mathbf{(41.6875)_{10} = (101001.1011)_2}$

➢ Decimal to Octal Conversion

1. ***Convert*** $(\mathbf{125})_{\mathbf{10}}$ ***into octal.***

8	125	
8	15	5 (LSB)
8	1	7
	0	1 (MSB)

$(125)_{10} = (175)_8$

2. ***Convert*** $(\mathbf{153.513})_{\mathbf{10}}$ ***into octal.***

Conversion of integer part:

8	153	
8	19	1 (LSB)
8	2	3
	0	2 (MSB)

$(153)_{10} = (231)_8$

Conversion of fractional part:

	integer	fraction
0.513×8	4	0.104
0.104×8	0	0.832
0.832×8	6	0.656
0.656×8	5	0.248
0.248×8	1	0.984
0.948×8	7	0.584

$(0.513)_{10} = (0.406517)_8$

$\therefore\ (\mathbf{153.513})_{\mathbf{10}} = (\mathbf{231.406517})_{\mathbf{8}}$

3. ***Convert*** $(\mathbf{125.6875})_{\mathbf{10}}$ ***to octal.***

Conversion of integer part:

8	125	
8	15	5 (LSB)
8	1	7
	0	1 (MSB)

$(125)_{10} = (175)_8$

Conversion of fractional part:

	integer	fraction
0.6875×8	5	0.5000
0.5000×8	4	0.0000

$(0.6875)_{10} = (54)_8$

$\therefore\ (\mathbf{125.6875})_{\mathbf{10}} = (\mathbf{175.54})_{\mathbf{8}}$

➢ Decimal to Hexadecimal Conversion

1. ***Convert* $(\mathbf{2598})_{\mathbf{10}}$ *to hexadecimal.***

16	2598	
16	162	6 (LSB)
16	10	2
	0	10(A) (MSB)

$(2598)_{10} = (A26)_{16}$

2. ***Convert* $(\mathbf{952.62})_{\mathbf{10}}$ *to hexadecimal.***

Conversion of integer part:

16	952	
16	59	8 (LSB)
16	3	11(B)
	0	3 (MSB)

$(952)_{10} = (3B8)_{16}$

Conversion of fractional part:

	integer	fraction
0.62×16	9	0.92
0.92×16	14(E)	0.72
0.72×16	11(B)	0.52
0.52×16	8	0.32
0.32×16	5	0.12
0.12×16	1	0.92

$(0.513)_{10} = (0.9EB851)_{16}$

$\therefore \mathbf{(952.62)_{10} = (3B8.9EB851)_{16}}$

<u>Any Base to Decimal Conversion</u>

Converting from any base to decimal is done by multiplying each digit by its corresponding positional weights and summing.

➢ Binary to Decimal

1. ***Convert* $(\mathbf{10110})_{\mathbf{2}}$ *into decimal.***

$$\begin{aligned}(10110)_2 &= 1\times 2^4 + 0\times 2^3 + 1\times 2^2 + 1\times 2^1 + 0\times 2^0\\ &= 16+0+4+2+0\\ &= 22\end{aligned}$$
$\therefore \mathbf{(10110)_2 = (22)_{10}}$

2. ***Convert* $(\mathbf{1101.011})_{\mathbf{2}}$ *into decimal.***

$$\begin{aligned}(1101.011)_2 &= 1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0 + 0\times 2^{-1} + 1\times 2^{-2} + 1\times 2^{-3}\\ &= 8+4+0+1+0+0.25+0.125\\ &= 13.375\end{aligned}$$
$\therefore \mathbf{(1101.011)_2 = (13.375)_{10}}$

- **Octal to Decimal**

1. ***Convert*** $\mathbf{(724.25)_8}$ ***into decimal.***

$$\begin{aligned}(724.25)_8 &= 7 \times 8^2 + 2 \times 8^1 + 4 \times 8^0 + 2 \times 8^{-1} + 5 \times 8^{-2} \\ &= 448 + 18 + 4 + 0.25 + 0.0781 \\ &= 470.3281\end{aligned}$$

$\therefore \mathbf{(724.25)_8 = (470.3281\ )_{10}}$

2. ***Convert*** $\mathbf{(6301)_8}$ ***into decimal.***

$$\begin{aligned}(6301)_8 &= 6 \times 8^3 + 3 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 \\ &= 3072 + 192 + 0 + 1 \\ &= 3265\end{aligned}$$

$\therefore \mathbf{(6301)_8 = (3265\ )_{10}}$

- **Hexadecimal to Decimal**

1. ***Convert*** $\boldsymbol{(A0F9.0EB)_{16}}$ ***to decimal.***

$$\begin{aligned}(A0F9.0EB)_{16} \\ &= 10 \times 16^3 + 0 \times 16^2 + 15 \times 16^1 + 9 \times 16^0 + 0 \times 16^{-1} + 14 \\ &\quad \times 16^{-2} + 11 \times 16^{-3} \\ &= 40960 + 0 + 240 + 9 + 0 + 0.0546 + 0.0026 \\ &= 41209.0572\end{aligned}$$

$\therefore \boldsymbol{(A0F9.0EB)_{16} = (41209.0572\ \ )_{10}}$

2. ***Convert*** $\boldsymbol{(A9F5.DE)_{16}}$ ***to decimal.***

<u>Octal and Hexadecimal number to Binary</u>

- **Octal to Binary**

To convert Octal number to its Binary equivalent, each digit of given octal number is directly converted to its 3-bit binary equivalent.

Q. Convert $\mathbf{(367.52)_8}$ ***into Binary.***

$$\begin{aligned}(367.52)_8 &= 011\ \ 110\ \ 111\ .101\ \ 010 \\ &= 11110111.10101\end{aligned}$$

$\therefore \mathbf{(367.52)_8 = (11110111.10101)_2}$

- **Hexadecimal to Binary**
  To convert Hexadecimal number to its Binary equivalent, each digit of given hexadecimal number is converted to its 4-bit binary equivalent.

  Q. Convert $(3A9E.B0D)_{16}$ into Binary.

$$\begin{aligned}(3A9E.B0D)_{16} &= 0011\ \ 1010\ \ 1001\ \ 1110\,.\,1011\ \ 0000\ \ 1101\\ &= 11101010011110.101100001101\end{aligned}$$

$$\therefore \mathbf{(3A9E.B0D)_{16} = (11101010011110.101100001101\,)_2}$$

<u>**Binary to Octal and Hexadecimal Numbers**</u>

- **Binary to Octal**
  As $8 = 2^3$, for binary to octal conversion groups of 3 binary bits each are formed in the binary number. After forming groups, each group of three binary bits is converted to its octal equivalent.
  - For integer part of the binary number, the group of three bits is formed from right to left. In the binary fraction the group of three bits is formed from left to right. If there are not 3 bits available at last, just stuff '0' to make 3 bits group.

  1. ***Convert $\mathbf{(10110001101011.111100000110)_2}$ into octal.***

$$\begin{aligned}(10110001101011.111100000110)_2 &= 010\ \ 110\ \ 001\ \ 101\ \ 011.111\ \ 100\ \ 000\ \ 110\\ &= \ \ 2\quad\ 6\quad\ 1\quad\ 5\quad\ 3\ .\ 7\quad\ 4\quad\ 0\quad\ 6\\ &= 26153.7406\end{aligned}$$

$$\therefore \mathbf{(10110001101011.111100000110)_2 = (26153.7406)_8}$$

  2. ***Convert $\mathbf{(110101.101010)_2}$ into octal.***

$$\begin{aligned}(110101.101010)_2 &= 110\ \ 101\,.\,101\ 010\\ &= 65.52\end{aligned}$$

$$\therefore \mathbf{(110101.101010)_2 = (65.52)_8}$$

- **Binary to Hexadecimal**
  As $16 = 2^4$, for binary to hexadecimal conversion groups of 4 binary bits each are formed in the binary number. After forming groups, each group of four binary bits is converted to its hexadecimal equivalent.
  - For integer part of the binary number, the group of four bits is formed from right to left. In the binary fraction the group of four bits is formed from left to right. If there are not 4 bits available at last, just stuff '0' to make 4 bits group.

1. ***Convert*** $(\mathbf{10110001101011.111100000110})_2$ ***into hexadecimal.***

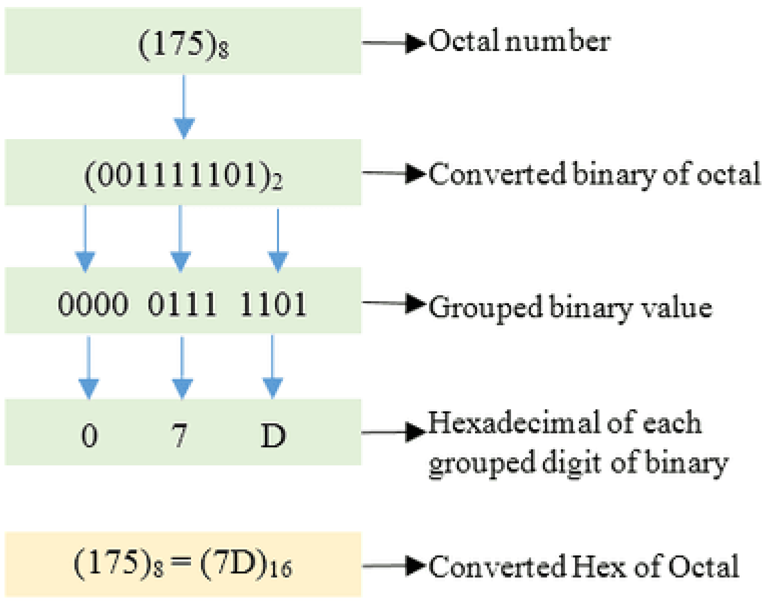
$$\begin{aligned}(10110001101011.111100000110)_2 &= 0010\ \ 1100\ \ 0110\ \ 1011.1111\ \ 0000\ 0110\\ &= \quad 2 \quad\ C \quad\ 6 \quad\ B \quad\ F \quad\ 0 \quad\ 6\\ &= 2C6B.F06\end{aligned}$$

$\therefore (\mathbf{10110001101011.111100000110})_2 = (2C6B.F06)_{16}$

<u>Conversion of Octal and Hexadecimal numbers</u>

- **Octal to Hexadecimal**
  Steps to convert from octal to its hexadecimal equivalent:
  - Each digit of given octal number is converted into its 3-bit binary equivalent.
  - Now, form the groups of 4 binary bits to obtain its hexadecimal equivalent.

  E.g.

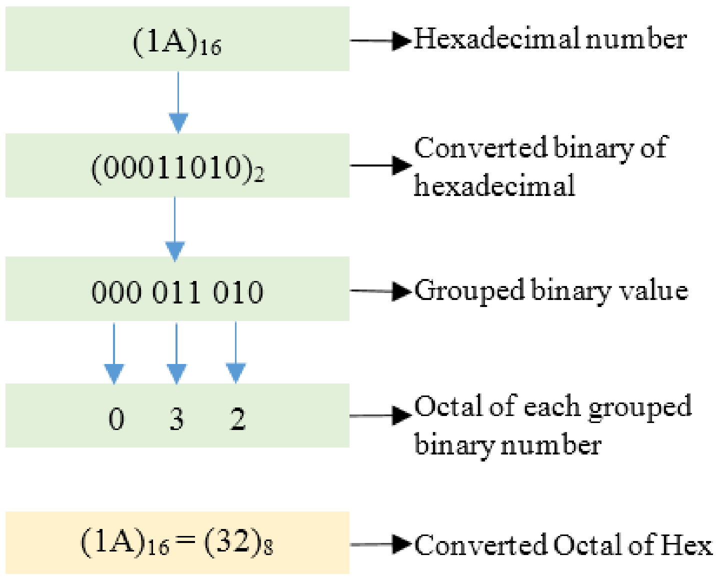


Q. Convert $(\mathbf{673.124})_8$ ***into hexadecimal.***

$$\begin{aligned}(673.124)_8 &= (\underline{110}\ \underline{111}\ \underline{011}\ .\ \underline{001}\ \underline{010}\ \underline{100})_2\\ &= (\underline{0001}\ \underline{1011}\ \underline{1011}\ .\ \underline{0010}\ \underline{1010})_2\\ &= (1BB.2A)_{16}\end{aligned}$$

- **Hexadecimal to Octal**
  Steps to convert from hexadecimal to its octal equivalent:
  - Each digit of given hexadecimal number is converted into its 4-bit binary equivalent.
  - Now, form the groups of 3 binary bits to obtain its octal equivalent.

  E.g.



Q. Convert $(B9F.AE)_{16}$ ***into octal.***

$$\begin{aligned}(B9F.AE)_{16} &= (1011\ 1001\ 1111\ .\ 1010\ 1110)_2\\ &= (101\ 110\ 011\ 111\ .\ 101\ 011\ 100)_2\\ &= (5637.534)_8\end{aligned}$$

<u>Unsigned and Signed numbers</u>

<u>*Unsigned numbers:*</u>

- Numbers without any positive and negative sign.
- Represents only magnitude.

<u>*Signed magnitude numbers:*</u>

- In binary number system, both +ve and –ve values are possible.
- In this, we use 0's and 1's to represent every number. The representation of number is known as signed number.
  - 0 → +ve number
  - 1 → -ve number

E.g.

+7 = 0111

-7 = 1111

This kind of representation for signed number is called signed magnitude representation.

Note: For +ve numbers, the result is the same as the unsigned binary representation.

The signed numbers are represented in three ways: Sign-magnitude form, 1's complement form and 2's complement form

1. ***Sign-Magnitude form***

   In this form, a binary number has a bit for a sign symbol. If this bit is set to 1, the number will be negative else the number will be positive if it is set to 0. Apart from this sign-bit, the n-1 bits represent the magnitude of the number.

   Syntax:

Sign Bit	Actual binary

   E.g. +7 = 0111
   -7 = 1111

2. ***1's Complement***

   By inverting each bit of a number, we can obtain the 1's complement of a number. The negative numbers can be represented in the form of 1's complement. In this form, the binary number also has an extra bit for sign representation as a sign-magnitude form.

   Syntax:

Sign Bit	1's complement of actual binary

   E.g. 7 = 111
   1's complement = 000
   - 7 = 1000

3. ***2's Complement***

   By inverting each bit of a number and adding plus 1 to its least significant bit, we can obtain the 2's complement of a number. The negative numbers can also be represented in the form of 2's complement. In this form, the binary number also has an extra bit for sign representation as a sign-magnitude form.

   Syntax:

Sign Bit	2's complement of actual binary

   E.g. 7 = 111
   1's complement = 000
   2's complement = 000 + 1 = 001
   - 7 = 1001

<u>**Floating-Point Representation**</u>

A floating-point number is represented by the triple:

- **S** is the **sign bit** (0 is +ve and 1 is –ve)
  - Representation is called sign and magnitude.
- **E** is the **Exponent field** (signed)
  - Very large numbers have large positive exponents
  - Very small close-to-zero numbers have negative exponents
  - More bits in exponent field increases range of values
- **F** is the **fraction field** (fraction after binary point)
  - More nits in fraction field improves the precision of FP numbers.

S	**Exponent**	**Fraction**

Value of a floating-point number = $(-1)^{S} \times val(F) \times 2^{val(E)}$

Binary Addition

Addition rules:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 with carry 1

1 + 1 + 1 = 1 with carry 1

Example:

```
0011010 + 001100 = 00100110

   1 1          carry
 0 0 1 1 0 1 0  = 26₁₀
+0 0 0 1 1 0 0  = 12₁₀
 -------------
 0 1 0 0 1 1 0  = 38₁₀
```

Binary Subtraction

Subtraction rules:

0 - 0 = 0

1 - 0 = 1

0 - 1 = 1 with borrow 1

1 - 1 = 0

Example:

0011010 - 001100 = 00001110

```
     1 1        borrow
 0 0 ~~1 1~~ 0 1 0  = 26₁₀
-0 0 0 1 1 0 0  = 12₁₀
 -------------
 0 0 0 1 1 1 0  = 14₁₀
```

Binary Multiplication

Multiplication rules:

0 x 0 = 0

0 x 1 = 0

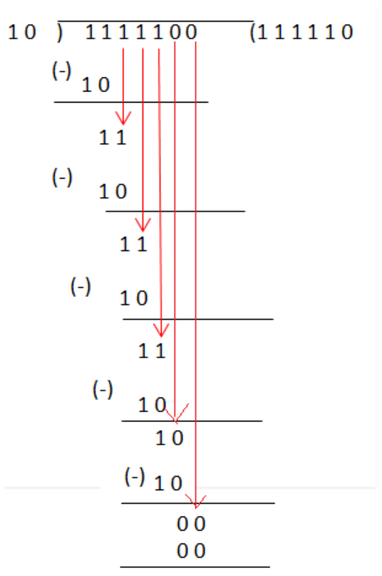
1 x 0 = 0

1 x 1 = 1

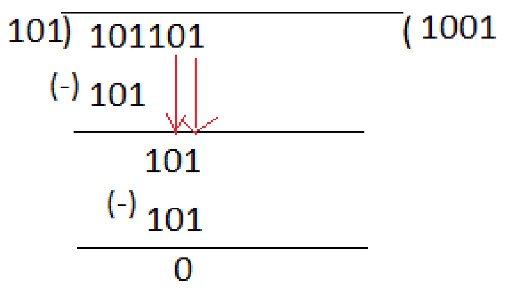
Example:

```
      1 0 0 1
      x 1 0 1
    ---------
      1 0 0 1
    0 0 0 0
+ 1 0 0 1
  -----------
  1 0 1 1 0 1
```

Binary Division



```
10 ) 1111100 (111110
(-) 10
    ----
     11
(-)  10
     ----
      11
(-)   10
      ----
       11
(-)    10
       ----
        10
(-)     10
        ----
        00
        00
        ----
```



```
101) 101101 ( 1001
 (-) 101
     ------
       101
   (-) 101
       ------
         0
```

Complements

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulations.

There are two types of complements for each base$-r$ system:
a) The $r's$ complement and
b) The $(r-1)'s$ complement.

- ✓ $r's$ complement is known as 10's complement in base 10 and 2's complement in base 2.
- ✓ $(r-1)'s$ complement is known as 9's complement in base 10 and 1's complement in base 2.

➢ $r's$ complement

Given a positive number N in base r with an integer part of n digits, the $r's$ complement of N is defined as

The $r's$ complement of $N = \begin{cases} r^n - N, & if\ N \neq 0 \\ 0, & if\ N = 0 \end{cases}$

E.g.

10's complement of 52520 = 10^5-52520 = 47480

10's complement of 0.3267 = 10^0-0.3267 = 0.6733

10's complement of 25.639 = 10^2-25.639 = 74.361

2's complement of $(101100)_2$ = $(2^6)_{10}$-$(101100)_2$ = $(1000000\text{-}101100)_2$ = 010100

2's complement of $(0.0110)_2$ = $(2^0)_{10}$-0.0110 = 0.1010

➢ $(r-1)'s$ complement

Given a positive number N in base r with an integer part of n digits and a fractional part of m digits, the $(r-1)'s$ complement of N is defined as:

The $(r-1)'s$ complement of $N = r^n - r^{-m} - N$

E.g.

9's complement of $(52520)_{10}$ = $(10^5\text{-}10^0\text{-}52520)$ = 47479

9's complement of $(0.3267)_{10}$ is $(10^0\text{-}10^{-4}\text{-}0.3267)$ = 0.6732

9's complement of $(25.693)_{10}$ is $(10^2\text{-}10^{-3}\text{-}25.693)$ = 74.306

1's complement of $(101100)_2$ is $(2^6\text{-}2^0)_{10}$-$(101100)_2$ = 111111-101100 = 010011

1's complement of $(0.0110)_2$ is $(1\text{-}2^{-4})_{10}$-$(0.0110)_2$ = 0.1001

<u>Subtraction with Complements</u>

- **<u>Subtraction with r's Complements</u>**
  Subtraction of two positive numbers $(M - N)$, both of base r, may be done as follows:
  Step-1: Add the minuend M to the $r's$ complement of subtrahend N.
  Step-2: Inspect the result obtained in step 1 for an end carry:
  (a) If an end carry occurs, discard it.
  (b) If an end carry does not occur, take the r's complement of the number obtained in step 1 and place a negative sign in front.

E.g.

1. Using 10's complement, subtract (72532-3250).

Let, $M = 72532$ and
$N = 03250$
10's complement of $N = (10^5 - 03250) = 96750$
Now,

$$\begin{array}{r} 72532 \\ \underline{+96750} \\ 1\ 69282 \end{array}$$

Here, end carry occurred. So discard it.
So, answer = 69282

2. ***Using 10's complement, subtract (3250-72532).***

Let, $M = 03250$ and
$N = 72532$
10's complement of $N = (10^5 - 72532) = 27468$
Now,

$$\begin{array}{r} 03250 \\ \underline{+27468} \\ 30718 \end{array}$$

Here, no end carry. .
So, answer = $-$(10's complement of 30718) = $-\left(10^5 - 30718\right) = -69282$

3. Using 2's complement, subtract (1000100-1010100).

Let, $M = 1000100$ and
$N = 1010100$
2's complement of $N = (2^7)_{10} - (1010100)_2 = 10000000 - 1010100 = 0101100$
Now,

$$\begin{array}{r} 1000100 \\ \underline{+0101100} \\ 1110000 \end{array}$$

No end carry. .
So, answer = $-$(2's complement of 1110000) = $-[(2^7)_{10} - (1110000)_2] = -10000$

4. ***Subtract (1110.111-1010.101) using 2's complement.***

Let, $M = 1110.111$ and
$N = 1010.101$
2's complement of $N = (2^4)_{10} - (1010.101)_2 = 10000 - 1010.101 = 0101.011$
Now,

$$\begin{array}{r} 1110.111 \\ \underline{+0101.011} \\ 1\;0100.010 \end{array}$$

End carry occurred. So discard it.
So, answer $= 0100.010$

➢ **<u>Subtraction with (r-1)'s Complements</u>**

The subtraction of $M - N$, both positive number in base r, may be calculated in the following manner:
Step-1: Add the minuend M to the $(r-1)'s$ complement of the subtrahend N.
Step-2: Inspect the result obtained in step 1 for an end carry.
(a) If an end carry occurs, add 1 to the list significant digit (end-round carry)
(b) If an end carry does not occurs, take the $(r-1)'s$ complement of the number obtained in step 1 and place a negative sign in front.

E.g.

1. Using 9's complement, subtract (453.35-321.17).

Let, $M = 453.35$ and
$N = 321.17$
9's complement of $N = 10^3 - 10^{-2} - 321.17 = 678.82$
Now,

$$\begin{array}{r} 453.35 \\ \underline{+678.82} \\ 1\;132.17 \\ \underline{+1} \\ 132.18 \end{array}$$

End carry occurred.
End round carry.

∴ Answer $= 132.18$

2. Using 9's complement, subtract (3250 – 72532).

Let, $M = 03250$ and
$N = 72532$
9's complement of $N = 10^5 - 10^0 - 72532 = 27467$
Now,

$$\begin{array}{r} 03250 \\ \underline{+27467} \\ 30717 \end{array}$$

Here, no end carry. .
So, answer = $-$(9's complement of 30717) = $-\left(10^5 - 10^0 - 30718\right) = -69282$

3. Using 1's complement, subtract (1000100-1010100).

Let, $M = 1000100$ and
$N = 1010100$
1's complement of $N = 0101011$
Now,

$$\begin{array}{r} 1000100 \\ \underline{+0101011} \\ 1101111 \end{array}$$

No end carry. .
So, answer = $-(1\text{'s complement of } 1101111) = -10000$

4. Subtract (1010100 - 1000100) using 1's complement.

Let, $M = 1010100$ and $N = 1000100$
1's complement of $N = 0111011$

$$\begin{array}{r} 1010100 \\ \underline{+0111011} \\ 1\,0001111 \\ \underline{+1} \\ 0010000 \end{array}$$

End carry occurred.
End round carry.

$\therefore$ Answer = 10000

Binary Codes

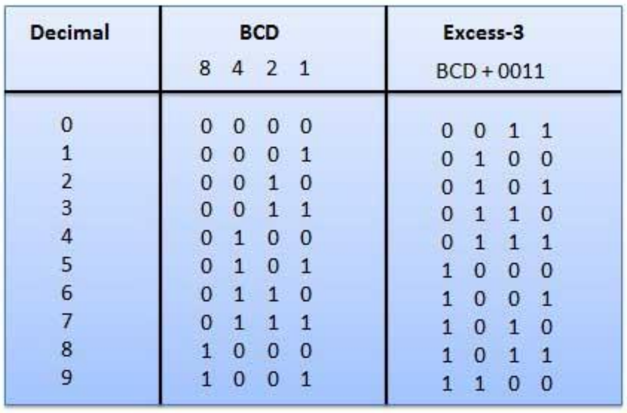
When numbers, letters or words are represented by a special group of binary symbols/combinations, we say that they are being encoded and the group of symbols is called a code. Some familiar binary codes are: Decimal Codes, Error-detection Codes, The Reflected Code, Alphanumeric Codes etc.

Decimal Codes

The representation of decimal digits by binary combinations is known as decimal codes. Binary codes from decimal digits require minimum of four bits. Numerous different codes can be obtained by arranging four or more bits in ten distinct possible combinations. Some decimal codes are-
- BCD
- Excess-3

➢ Binary-Coded-Decimal (BCD) Code

If each digit of a decimal number is represented by its binary equivalent, the result is a code called binary coded decimal (BCD). It is possible to assign weights to the binary bits according to their positions. The weights in the BCD code are 8,4,2,1.



Decimal	BCD 8 4 2 1	Excess-3 BCD + 0011
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

- **<u>Excess-3 Code</u>**
  This is an unweighted code. Its code assignment is obtained from the corresponding value of BCD after the addition of 3.

Q. Convert decimal number 23 to Excess-3 code.

So, according to excess-3 code we need to add 3 to both digit in the decimal number then convert into 4-bit binary number for result of each digit. Therefore,

$$\begin{array}{rr} 2 & 3 \\ \underline{+3} & \underline{3} \\ 5 & 6 \end{array}$$

$5\ \ 6 = 0101\ 0110$ which is required excess-3 code for given decimal number 23

Q. Convert decimal number 15.46 into Excess-3 code.

According to excess-3 code we need to add 3 to both digit in the decimal number then convert into 4-bit binary number for result of each digit. Therefore,

$$\begin{array}{rrrr} 1 & 5. & 4 & 6 \\ \underline{+3} & \underline{3.} & \underline{3} & \underline{3} \\ 4 & 8. & 7 & 9 \end{array}$$

48.79 = 0100 1000.0111 1001 which is required excess-3 code for given decimal number 15.46.

<u>Error-detection Codes</u>

- An error detection codes can be used to detect errors during transmission. A parity bit is an extra bit included with a message to make the total number of 1's either odd or even.
- For a message of four bits parity (P) is chosen so that the sum of all 1's is odd (in all five bits) or the sum of all 1's is even. In the receiving end, all the incoming bits (in this case five) are applied to a "parity-check" network for checking.
- An error is detected if the check parity does not correspond to the adopted one. The parity method detects the presence of one, three or any odd combination of errors. An even combination of errors is undetectable. Additional error-detection schemes may be needed to take care of an even combination of errors.

Message	P (Odd)	Total Message	Message	P (even)	Total Message
0000	1	10000	0000	0	00000
0001	0	00001	0001	1	10001
0010	0	00010	0010	1	10010
0011	1	10011	0011	0	00011
0100	0	00100	0100	1	10100
0101	1	10101	0101	0	00101
0110	1	10110	0110	0	00110
0111	0	00111	0111	1	10111
1000	0	01000	1000	1	11000
1001	1	11001	1001	0	01001
1010	1	11010	1010	0	01010
1011	0	01011	1011	1	11011
1100	1	11100	1100	0	01100
1101	0	01101	1101	1	11101
1110	0	01110	1110	1	11110
1111	1	11111	1111	0	01111

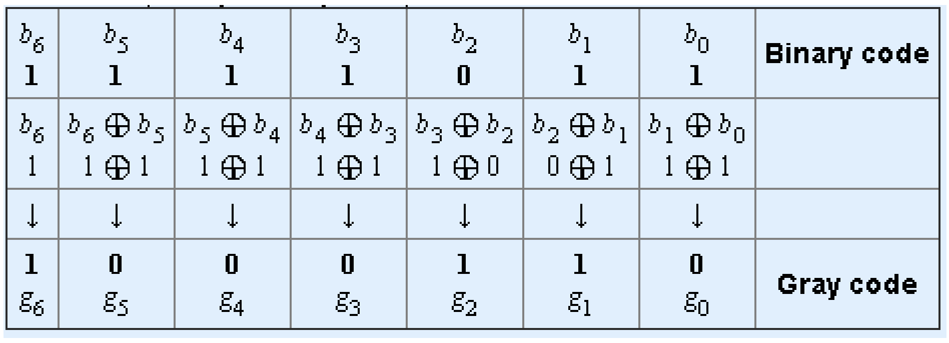
The Reflected Code/Gray Code

- The Reflected code, also called Gray code is unweighted and is not an arithmetic code; that is, there are no specific weights assigned to the bit positions.
- It is a binary numeral system where two successive values differ in only one bit (binary digit).
- For instance, in going from decimal 3 to decimal 4, the Gray code changes from 0010 to 0110, while the binary code changes from 0011 to 0100, a change of three bits. The only bit change is in the third bit from the right in the Gray code; the others remain the same.

Decimal Digit	Binary	Reflected or gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Q. Convert $(123)_{10}$ to gray code.

Binary code of $(123)_{10}$ = $(1111011)_2$



b_6 **1**	b_5 **1**	b_4 **1**	b_3 **1**	b_2 **0**	b_1 **1**	b_0 **1**	**Binary code**
b_6 1	$b_6 \oplus b_5$ $1 \oplus 1$	$b_5 \oplus b_4$ $1 \oplus 1$	$b_4 \oplus b_3$ $1 \oplus 1$	$b_3 \oplus b_2$ $1 \oplus 0$	$b_2 \oplus b_1$ $0 \oplus 1$	$b_1 \oplus b_0$ $1 \oplus 1$	
↓	↓	↓	↓	↓	↓	↓	
1 g_6	**0** g_5	**0** g_4	**0** g_3	**1** g_2	**1** g_1	**0** g_0	**Gray code**

∴ $(123)_{10}$ = $(1000110)_{GrayCode}$

Alphanumeric Code

- In order to communicate, we need not only numbers, but also letters and other symbols. In the strictest sense, alphanumeric codes are codes that represent numbers and alphabetic characters (letters). Most such codes, however, also represent other characters such as symbols and various instructions necessary for conveying information.
- The ASCII is the most common alphanumeric code.

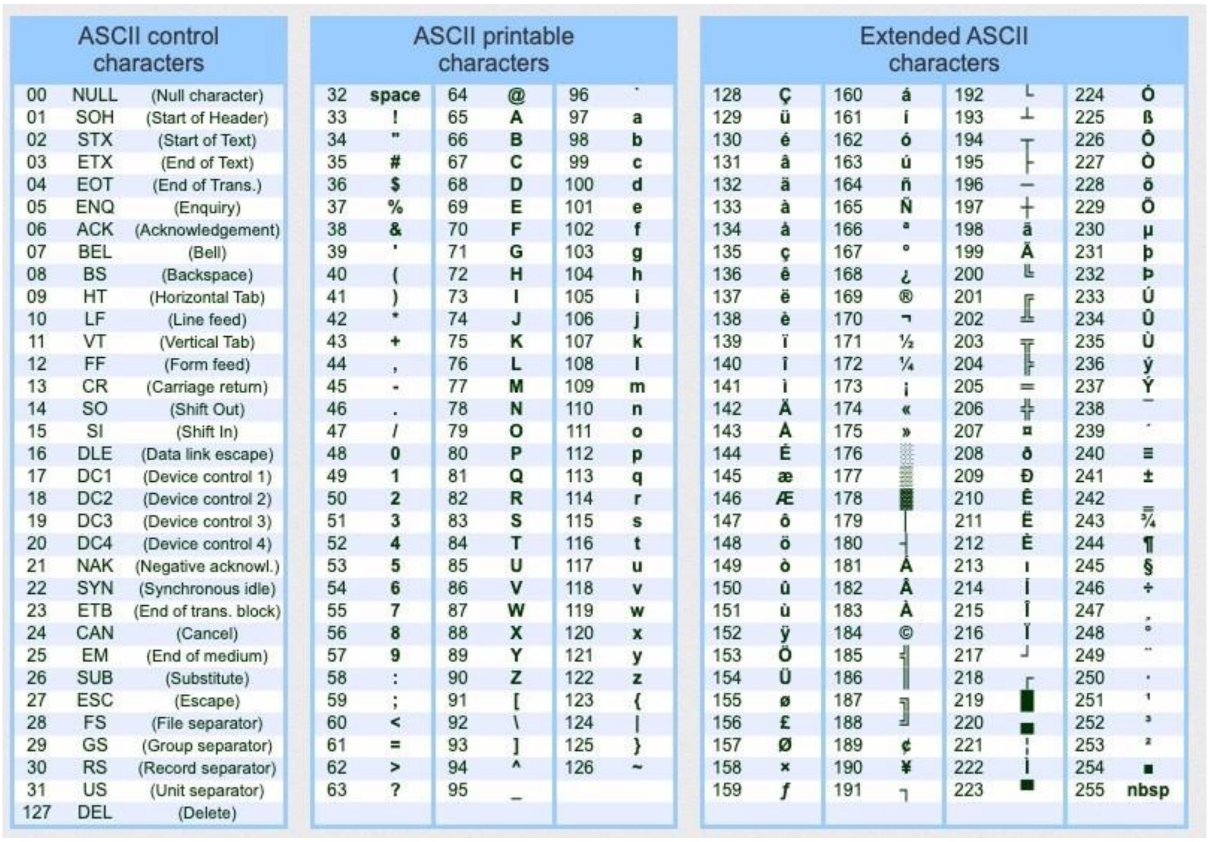
➢ ASCII Code

ASCII is the abbreviation for **American Standard Code for Information Interchange**. ASCII is a universally accepted alphanumeric code used in most computers and other electronic equipment. Most computer keyboards are standardized with the ASCII. When we enter a letter, a number, or control command, the corresponding ASCII code goes into the computer.

- ASCII has 128 characters and symbols represented by a 7-bit binary code. Actually, ASCII can be considered an 8-bit code with the MSB always 0. This 8-bit code is 00 through 7F in hexadecimal.
- The first thirty-two ASCII characters are non-graphic commands that are never printed or displayed and are used only for control purposes. Examples of the control characters are ""null," "line feed," "start of text," and "escape."
- The other characters are graphic symbols that can be printed or displayed and include the letters of the alphabet (lowercase and uppercase), the ten decimal digits, punctuation signs and other commonly used symbols.

➢ Extended ASCII characters

In addition to the 128 standard ASCII characters, there are an additional 128 characters that were adopted by IBM for use in their PCs (personal computers). Because of the popularity of the PC, these particular extended ASCII characters are also used in applications other than PCs and have become essentially an unofficial standard. The extended ASCII characters are represented by an 8-bit code series from hexadecimal 80 to hexadecimal FF.



ASCII control characters		
00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters					
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters							
128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

<u>References</u>:

- ***M. Morris Mano, "Digital Logic & Computer Design"***