- **<u>Number system</u>:**

  For any number, which contains *(n+1)* integral digits and *k* fraction digits, then it is represented by:

  $d_n d_{n-1} d_{n-2} d_{n-3} \ldots . d_2 d_1 d_0 . d_{-1} d_{-2} d_{-3} \ldots . d_{-k}$

  For any number with base *'b',* its decimal equivalent is:

  $(d_n \times b_n) + (d_{n-1} \times b_{n-1}) + .. + (d_0 \times b_0) + (d_{-1} \times b_{-1}) + \ldots + (d_{-k} \times b_{-k})$

  There are different number system, divided as per the uses required by a computer to process, transmit and store data in a better way.

  There are 4 types of number system:

  1. Binary system (0, 1)
  2. Octal System (0, 1, 2, 3, 4, 5, 6, 7)
  3. Decimal System (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
  4. Hexadecimal System (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

- **Binary number system:**
  - Base or Radix = 2

    (Base: The number of symbols used in a number system is called its base or its radix)
  - Symbols are '0' and '1'.
  - Binary digits are also known as 'Bits' in short.
  - This number is used for **processing data** in computers.
  - Any binary number with 'n' integral digits and 'k' fractional digits can be written as-

    $b_n\ b_{n-1}\ b_{n-2} \ldots b_2\ b_1\ b_0\ .\ b_{-1}\ b_{-2} \ldots b_{-k}$

    Where b_n = Most significant bit;

    b_{-k} = Least significant bit.

    Examples = $(10)_{10} = (1010)_2$, $(5)_{10} = (101)_2$, $(1)_{10} = (1)_2$, etc.
- **Octal number system:**
  - Base or radix = 8

- Symbols are 1, 2, 3, 4, 5, 6, and 7.
- These numbers are used for **transmitting data** in computers.
- Any octal number with 'n' integral and 'k' fractional digits can be written as-

  $o_n\ o_{n-1}\ o_{n-2} \ldots o_2\ o_1\ o_0\ .\ o_{-1}\ o_{-2} \ldots o_{-k}$

  Examples = $(10)_{10} = (12)_8$, $(5)_{10} = (5)_8$, $(17)_{10} = (21)_8$, etc.

- **Hexadecimal number system:**
  - Base or radix = 16
  - Symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
  - These numbers are used for **storing data** in computers.

    Example: $(10)_{10} = (10)_{16}$, $(15)_{10} = (F)_{16}$, $(17)_{10} = (11)_{16}$, etc.

- **Decimal number system:**
  - Base or radix = 10
  - Symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.
  - These numbers are used for **showing results to user** in computers.
  - Numbers are written as-

    0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, etc.

- **<u>Conversion of number systems:</u>**

  1. Binary to decimal:

  Binary number- $b_{n-1}\ b_{n-2} \ldots b_2\ b_1\ b_0\ .\ b_{-1}\ b_{-2} \ldots b_{-k}$

  Decimal number- $(b_{n-1} \times 2^{n-1}) + (b_{n-2} \times 2^{n-2}) + \ldots + (b_0 \times 2^0) + (b_{-1} \times 2^{-1}) + (b_{-2} \times 2^{-2}) + \ldots + (b_{-k} \times 2^{-k})$.

  Example: $(11011)_2 = (27)_{10}$

  2. Binary to octal:

a. Make triads (collection of 3) from last integral digit and first decimal digit, and complete the last triad by putting zeros.
b. Convert each triad to their respective decimal equivalent value.
c. The result is the octal form of the given number.

Example: $(1011111010100000)_2 = (137240)_8$

- **Binary to hexadecimal:**
  a. Make groups of 4 bits from left of binary point, and complete last group by putting zeroes.
  b. Convert each group into decimal value and then corresponding hexadecimal values of the decimal value.
  c. The resultant number is a hexadecimal form of given number.

  Example: $(1100\ 0011\ 0000\ 1100)_2$

  12 3 0 12

  C 3 0 C

  $= (C30C)_{16}$

- **Octal to binary:**
  a. Convert every digit of octal into binary numbers of 3 bits each.
  b. Arrange them into original order.
  c. The resultant number is binary form of given octal number.

  Example = $(256)_8$

  010 101 110

  = $(010101110)_2$

- **Octal to hexadecimal:**

  Octal number -> Binary number -> Hexadecimal number

- **Hexadecimal to binary:**
    a. Convert each digit of hexadecimal number to binary number of 4 bits.
    b. Combining them we will get the required number.
- **Hexadecimal to octal:**
  Hexadecimal number -> Binary number -> Octal number.
- **<u>Addition of numbers:</u>**
- **Addition of decimals:**
    a. Arrange both numbers in ordered way.
    b. Starting from the last, add each digit and write the result.
    c. If carry, then add it to next digit's sum and repeat the process.
    d. Resulting number is the answer.
- **Addition of binary:**
    a. Use these results to add:
       0 + 0 = 0
       1 + 0 = 1
       0 + 1 = 1
       1 + 1 = 0 (With carry 1)
       1 + 1 + 1 = 1 (With carry 1)
    b. Add carry to next digit's sum.
- **Addition of octal or hexadecimal:**
    a. Arrange the numbers and add each digit from last.
    b. If carry, add it to next digit's sum and repeat the process.

- **<u>r's complement of a number:</u>**
  r's complement of a number can be given as:

  r's complement = $r^n - N$

  where r = base/radix of number system,
  n = number of digit
  N = number

- **<u>(r-1)'s complement of a number:</u>**

  (r-1)'s complement of a number be given a:

  (r-1)'s complement = r^n – N – 1

  Now, we can also write:

  r's = (r-1)'s + 1

  SHORT TRICK TO FIND r's AND r-1's:

  - Find (r-1)'s by subtracting each digit from highest symbol of given number system.
  - Add 1 to get r's compliment.

  SHORT TRICK TO FIND 2's AND 1's complement of Binary:

  - Find 1's by flipping each bit.
  - Add 1 to get 2's compliment.

- **<u>Rules for subtraction:</u>**
- **Using r's complement (A-B):**

  a. Adjust digits of A and B.
  b. Find r's complement of B.
  c. Add A and B'.
  d. If carry is present then ignore it;
     Otherwise Recompliment the sum and put a –ve sign to the produced result.
- **Using (r-1)'s complement:**

  a. Adjust A and B digits.
  b. Find (r-1)'s complement of B.
  c. Add A and B'.
  d. If carry is present then add the carry with the sum and produce the result;
     Otherwise recompliment the sum and put a –ve sign to produce result,

- **General method:**
  a. Adjust A and B digits.
  b. Subtract each digits of A from B from last one-by-one.
  c. The final number is the required result.

  NOTE: Computer uses the r's complement method ONLY, and other methods are just theoretical methods.

- **<u>ASCII (American Standard code for Information Interchange):</u>**
  It is a seven bit code and there are 2^7 (=128) different combinations of 0's and 1's to encode lowercase, uppercase and special characters besides 10 decimal symbols.
  It is extensively used for printer and terminals that interface the small computer system.
  Such as- A = 65, B = 66, etc.

- **<u>EBCDIC (Extended Binary Coded Decimal Interchange Code):</u>**
  It is 8 bit code in which there are 2^8 different combinations of 0's and 1's that are used to encode all the characters of ASCII and also some other characters.