NUMBER SYSTEM

$246 = 2x10^{2} + 4x10^{1} + 6x10^{0}$ $0.416 = 4x10^{-1} + 1x10^{-2} + 6x10^{-3}$

General expression for any Number System:

$$N = d_n x R^n + d_{n-1} x R^{n-1} + ... + d_i x R^i + ... + d_0 R^0$$

d_n: Digit at the nth position R: Base or Radix of the system

N : Positional value

Observe the pattern :

Binary: 0, 1, 10, 11, 100, 101, ...

Octal: 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, ...

Fractional Decimal to Binary:

$$0.375 \times 2 = 0.75 = 0 + 0.75$$

0.75 x 2 = 1.5 =
$$|\mathbf{1}|$$
 + 0.5 0.375₁₀ = 0.011₂

$0.5 \times 2 = 1.0 = |\mathbf{1}| + 0$

Fractional Binary to Decimal:

. 1 0 1 1

$$\frac{1}{2^{1}}x1 + \frac{1}{2^{2}}x0 + \frac{1}{2^{3}}x1 + \frac{1}{2^{4}}x1$$
 0.1011₂ = 0.6875₁₀
0.5 + 0 + 0.125 + 0.0625

BCD (Binary Coded Decimal):

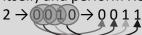
Different from Binary. Each digit is converted to binary.

Decimal: 12 Decimal: 33
Binary: 1100 Binary: 100001
BCD: 10010 BCD: 110011

1 2 - 0001 0010 3 3 - 0011 0011

Gray Code:

First bit remains same. From next bits, consider pair of bits (the previous one and itself) and perform XOR operation on them.



Excess-3 Code:

Convert to BCD. Add 3 (11) to every block. (Shortcut: add 3 to every digit. Then convert to BCD)

 $22 \rightarrow 0010 \ 0010 \ (BCD)$ +0011 0011 (+ 3 3)

0101 0101 \rightarrow Excess-3 code for 22

25.35 +33.33 (shortcut) →58.68

Excess-3 code for 25.35: 0101 1000.0110 1000

Octal to Binary:

Each octal digit → 3 bit equivalent binary representation

7 0 5
$$\longrightarrow$$
 Octal \downarrow \downarrow \downarrow \downarrow 705₈=111000101₂

111 000 101 —— Binary

Each hexadecimal digit \rightarrow 4 bit equivalent binary representation

1 0 A F
$$\downarrow$$
 \downarrow \downarrow 10AF₁₆ = 0001000010101111₂ 0001 0000 1010 1111

Binary to Octal:

Group bits in threes starting from the right. Convert to octal.

Binary to Hexadecimal:

Hexadecimal to Binary:

Group bits in fours starting from the right. Convert to hex.

Octal to Hexadecimal / Hexadecimal to Octal:

Use binary intermediate.

Representation of Signed Numbers:

Signed Magnitude form:

Signed Bit | Actual Binary

0 → +	<u>0</u>	<u>111</u>	<u>1</u>	<u>111</u>
1 → -	+	7	-	7
Using 8 bits:	$6_{10} = 00000110_0$			

Using 8 bits: $6_{10} = 00000110_2$ $-6_{10} = 10000110_2$ (1000 0000₂ = -0 this is invalid)

Complement form:

 $9 \to 1001$

1's complement \rightarrow 0110

2's complement \rightarrow 0111 (Add 1 to 1's complement)

1's complement: Signed Bit 1's component of Actual Binary

2's complement: Signed Bit 2's component of Actual Binary

 $+7 \rightarrow 0 \ 111$

 $-7 \rightarrow 1 111$ (Signed magnitude form)

1 000 (1's complement form)

1 001 (2's complement form)

*The maximum number that can be represented with a **k-bit** 2's complement notation is $2^{k-1}-1$

*A positive number remains same in 2's complement form

*Advantages of 2's complement form:

- Binary codes can be added regardless of the sign of the number

- Both +0 and -0 has the same representation