UNIT-I

Digital Logic Design

A digital logic is an electronic technology that is used to storing, processing and generating a data in terms of positive (or) non-positive states.

Advantages of digital system over analog system :-

1. Ease of programming.
2. High reliable
3. Ease of design
4. High speed.
5. Reduction in Hardware Components
6. Less Cost.

Number System :-

Binary number system → 0 & 1
Decimal number system → 0 to 9
Octal number system → 0 to 7
Hexadecimal number system → 0 to F

Decimal	Binary	Octal	Hexa
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7

8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Conversions :-

Binary Conversions to decimal :-

Ex :- $(1010)_2 = (?)_{10}$

$$1 \quad 0 \quad 1 \quad 0$$

(Most signi-ficant bit) MSB LSB (least signi-ficant bit)

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

$$8 + 0 + 2 + 0 = (10)_{10}$$

1. $(1011)_2 = (?)_{10}$

Sol:- $1 \quad 0 \quad 1 \quad 1$

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$8 + 0 + 2 + 1$$

$$= (11)_{10}$$

2. $(10011010)_2 = (?)_{10}$

Sol:- $1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0$

$$1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

$$= 126 + 0 + 0 + 16 + 8 + 2$$

$$= (154)_{10}$$

3. $(111)_2 = (?)_{10}$

Sol:- $1 \quad 1 \quad 1$

$1\times2^2 + 1\times2^1 + 1\times2^0$

$4+2+2$

$=(7)_{10}$

4. $(\overset{4}{1}\overset{3}{1}\overset{2}{0}\overset{1}{1}\overset{0}{0}.\overset{-1}{1}\overset{-2}{0}\overset{-3}{1})$

↑ Binary decimal

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

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

$26+0.5+0.125$

$(26.625)_{10}$

5. $\overset{4}{1}\overset{3}{0}\overset{2}{1}\overset{1}{1}\overset{0}{1}.\overset{-1}{1}\overset{-2}{1}\overset{-3}{0}$

Sol:- $1\times2^4 \times 0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2} + 0\times2^{-3}$

$16+0+4+2+1+\frac{1}{2}+\frac{1}{4}$

$16+4+2+1+0.5+0.25$

$(23.75)_{10}$

Binary to Octal :-

Example :- $(10110010)_2 = (?)_8$

$\underbrace{10}_{2}\ \underbrace{110}_{6}\ \underbrace{010}_{2} = (262)_8$

1. $(10111011)_2 = (?)_8$

Sol:- $\underbrace{101}_{5}\ \underbrace{111}_{7}\ \underbrace{011}_{3} = (573)_8$

2. $111011 \cdot 11010 = (?)_8$

Sol:- $\underbrace{111}_{7}\ \underbrace{011}_{3} \cdot \underbrace{110}_{6}\ \underbrace{10}_{2} = (73 \cdot 62)_8$

Binary to Hexadecimal.

1. $(10110010)_2 = (?)_{16}$

Sol:- $\underbrace{1011}_{B}\ \underbrace{0010}_{2}$

$= (B2)_{16}$

2. $(101111011)_2 = (?)_{16}$

Sol:- $\underbrace{1}_{1}\ \underbrace{0111}_{7}\ \underbrace{1011}_{B}$

$= (17B)_{16}$

Binary Conversions :-

Decimal to Binary :-

1. $(625)_{10} = (?)_2$

Sol:- $(1001000001)_2$

$$\begin{array}{r|l}
2 & 625 \\
2 & 312 - 1 \\
2 & 156 - 0 \\
2 & 78 - 0 \\
2 & 39 - 0 \\
 & 1
\end{array}$$

2. $(18)_{10} = (?)_2$

Sol:-
$$\begin{array}{r|l} 2 & 18 \\ 2 & 9 - 0 \\ 2 & 4 - 1 \\ 2 & 2 - 0 \\ & 1 - 0 \end{array}$$

$= (10010)_2$

3. $(32)_{10}$

Sol:-
$$\begin{array}{r|l} 2 & 32 \\ 2 & 16 - 0 \\ 2 & 8 - 0 \\ 2 & 4 - 0 \\ 2 & 2 - 0 \\ & 1 - 0 \end{array}$$

$= (100000)_2$

4. $(267)_{10}$

Sol:-
$$\begin{array}{r|l} 2 & 267 \\ 2 & 133 - 1 \\ 2 & 66 - 1 \\ 2 & 33 - 0 \\ 2 & 16 - 1 \\ 2 & 8 - 0 \\ 2 & 4 - 0 \\ 2 & 2 - 0 \\ & 1 - 0 \end{array}$$

$= (100001011)_2$

5. $(758)_{10}$

Sol:-
$$\begin{array}{r|l} 2 & 758 \\ 2 & 379 - 0 \\ 2 & 189 - 1 \\ 2 & 94 - 1 \\ 2 & 47 - 0 \\ 2 & 23 - 1 \\ 2 & 11 - 1 \\ 2 & 5 - 1 \\ 2 & 2 - 1 \\ & 1 - 0 \end{array}$$

$= (101110110)_2$

Decimal to octal :-

1. $(758)_{10} = (?)_8$

Sol:-
$$\begin{array}{r|l} 8 & 758 \\ \hline 8 & 94 - 6 \\ \hline 8 & 11 - 6 \\ \hline & 1 - 3 \end{array}$$

$= (1366)_8$

Convert the Binary to octa and Hexadecimal and decimal.

1. $(1011101110 \cdot 1010011)_2 = (?)_8 = (?)_{16}$

Sol:- Octa :- $\underbrace{1}\,\underbrace{011}\,\underbrace{101}\,\underbrace{110} \cdot \underbrace{101}\,\underbrace{001}\,\underbrace{1}$

$1 \quad 3 \quad 5 \quad 6 \cdot 5 \quad 1 \quad 1$

$= (1356 \cdot 511)_8$

Hexa :- $\underbrace{10}\,\underbrace{1110}\,\underbrace{1110} \cdot \underbrace{1010}\,\underbrace{011}$

$2 \quad E \quad E \quad A \quad 3$

$(2EE \cdot A3)_{16}$

2. $(10111 \cdot 101)_2 = (?)_{10} = (?)_8 = (?)_{16}$

Sol:- Decimal :- $10111 \cdot 101$

$1 \times 2^4 \times 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \cdot 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$

$16 + 0 + 4 + 2 + 1 \cdot \frac{1}{2} + 0 + \frac{1}{8}$

$23 \cdot 0{\cdot}5 + 0{\cdot}125$

$= (23 \cdot 625)_{10}$

Octa :- $\underbrace{10}\,\underbrace{111} \cdot \underbrace{101}$

$2 \quad 7 \cdot 5$

$= (27.5)_8$

Hexa :- $\underbrace{1}_{1}\ \underbrace{0111}_{7}\ .\ \underbrace{101}_{5}$

$= (17.5)_{16}$

3. $(11011011111011 . 011010)_2 = (?)_8 = (?)_{16}$

Sol:- $\underbrace{1}_{1}\ \underbrace{101}_{5}\ \underbrace{101}_{5}\ \underbrace{111}_{7}\ \underbrace{011}_{3}\ .\ \underbrace{011}_{3}\ \underbrace{010}_{2}$

$= (15573.32)_8$

4. $(1111011 . 1011)_2 = (?)_{10} = (?)_8 = (?)_{16}$

Sol:- decimal :- 1111011.1011

$1\times2^6+1\times2^5+1\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0 . 1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$
$+1\times2^{-4}$

$= 64+32+16+8+0+2+1 . \frac{1}{2}+0+\frac{1}{8}+\frac{1}{16}$

$= 123 . 0.5+0.125+0.0625$

$= (123.6875)_{10}$

Octa :- $\underbrace{1}_{1}\ \underbrace{111}_{7}\ \underbrace{011}_{3}\ .\ \underbrace{101}_{5}\ \underbrace{1}_{1}$

$= (173.51)_8$

Hexa :-

$\underbrace{111}_{7}\ \underbrace{1011}_{B}\ .\ \underbrace{1011}_{B}$

$= (7B.B)_{16}$

Convert decimal to Binary.

1. $(144)_{10} = (?)_2$

Sol:-

$2\,|\,144$
$2\,|\,72 - 0$
$2\,|\,36 - 0$
$2\,|\,18 - 0$
$2\,|\,9 - 0$
$2\,|\,4 - 1$
$2\,|\,2 - 0$
$\;\;\;\,|\,1 - 0$

$= (10010000)_2$

2. $(144)_{10} = (?)_8$

Sol:-

$8\,|\,144$
$8\,|\,18 - 0$
$\;\;\;\,|\,2 - 2$

$= (220)_8$

3. convert 256 decimal number number into equalent odal and hexa decimal.

Sol:- Octal :-

$8\,|\,256$
$8\,|\,32 - 0$
$\;\;\;\,|\,4 - 0$

$= (400)_8$

Hexa :-

$16\,|\,256$
$16\,|\,16 - 0$
$\;\;\;\;\;|\,1 - 0$

$= (100)_{16}$

4. Convert into equalent Binary number.

$(0.11)_{10} = (?)_2$

Sol:-

0.11×2	0.22
0.22×2	0.44
0.44×2	0.88
0.88×2	1.76

$= (0.0001)_2$

5. $(0.21)_{10} = (?)_2$

Sol:-

0.21×2	0.42
0.42×2	0.84
0.84×2	1.68
0.68×2	1.36

$= (0.0011)_2$

6. 0.75

Sol:-

0.75×2	1.5
0.5×2	1
0×2	0
0×2	0

$= (0.1100)_2$

Conversion of Decimal to Binary.

1. $(26.125)_{10} = (?)_2$

Sol:- 26.125

$$\begin{array}{r|l} 2 & 26 \\ 2 & 13 - 0 \\ 2 & 6 - 1 \\ 2 & 3 - 0 \\ & 1 - 1 \end{array}$$

$= (11010)_2$

0.125×2	0.25
0.25×2	0.5
0.5×2	1
0×2	0

$= (0.0010)_2$

$= (26.125)_2 = (11010.001)_2$

2: $(46.625)_{10} = (?)_2$

Sol:- 46.625

$$\begin{array}{r|l} 2 & 46 \\ 2 & 23 - 0 \\ 2 & 11 - 1 \\ 2 & 5 - 1 \\ 2 & 2 - 1 \\ & 1 - 0 \end{array}$$

$= 101110$

0.625×2	1.25
0.25×2	0.5
0.5×2	1

$= 101$

$= (101110.101)_2$

Octal Conversion :-

1. $(17.12)_8 = (?)$

Sol:- $8 \lfloor 17$
$\lfloor 2 - 1$

0·12×2	0·24	0·12×8	0·96
0·24×2	0·48	0·96×8	7·68
0·48×2	0·96	0·68×8	5·44
0·96×2	1·92	0·44×8	3·52

(21·0001)

$(21.0753)_8$

1. $(256.72)_{10} = (?)_8 = (?)_{16}$

Sol:- Octal :-

$8 \lfloor 256$
$8 \lfloor 32 - 0$
$\lfloor 4 - 0$

0·72×8	5·76
0·76×8	6·08
0·08×8	0·64
0·64×8	5·12

$(400.5605)_8$

Hexa :-

$16 \lfloor 256$
$16 \lfloor 16 - 0$
$\lfloor 1 - 0$

0·72×16	11·52
0·52×16	8·32
0·32×16	5·12
0·12×16	1·92

$(100.B851)_{16}$

2. $(4072.84)_{10} = (?)_8 = (?)_{16}$

Sol:- $8 \lfloor 4072$
$8 \lfloor 509 - 0$
$8 \lfloor 63 - 5$
$\lfloor 7 - 7$

0·84×8	6·72
0·72×8	5·76
0·76×8	6·08
0·08×8	3·84 0·64

$= (7750.6560)_8$

Hexa:-

$$16\,\underline{|4072}$$
$$16\,\underline{|254} - 8$$
$$\underline{|15} - 14$$

0.84×16	13.44
0.44×16	7.04
0.04×16	0.64
0.64×16	10.24

$$(FE8 \cdot D70A)_{16}$$

3. $(210.124)_{10} = (?)_8 = (?)_{16}$

Sol:- $8\,\underline{|210}$
$$8\,\underline{|26} - 2$$
$$8\,\underline{|3} - 3$$

0.124×8	0.99
0.99×8	7.92
0.92×8	7.36
0.36×8	2.88

$$= (332.0772)_8$$

Hexa:-

$$16\,\underline{|210}$$
$$\underline{|13} - 2$$

0.124×16	2.006
0.006×16	0.096
0.096×16	1.536
0.536×16	8.576

0.124×16

$$= (D \cdot 2018)_{16}$$

Octal conversions.

⇒ Octal to demical.

1. $(17.12)_8 = (?)_{10}$

Sol:- $\overset{1}{1}\ \overset{0}{7}\ .\ \overset{-1}{1}\ \overset{-2}{2}$

$1 \times 8^1 + 7 \times 8^0 + 1 \times 8^{-1} + 2 \times 8^{-2}$

$= 1 \times 8 + 7 \times 1 + 1 \times \frac{1}{8} + 2 * \frac{1}{8^2}$

$= 8 + 7 + 0.125 + 2 * 0.0156.$

$= 15 + 0.156$

$= (15.156)_{10}.$

2. $(123.120)_8 = (?)_{10}$

Sol:- $\overset{2}{1}\ \overset{1}{2}\ \overset{0}{3}\ .\ \overset{-1}{1}\ \overset{-2}{2}\ \overset{-3}{0}$

$= 1 \times 8^2 + 2 \times 8^1 * + 3 \times 8^0 + 1 \times 8^{-1} + 2 \times 8^{-2} + 0 \times 8^{-3}$

$= 64 + 16 + 3 + 1 \times \frac{1}{8} + 2 \times \frac{1}{\cancel{1}64} + 0$

$= 83 + 0.156$

$= (83.156)_{10}$

3. $(4071.125)_8 \rightarrow (?)_{10}$

Sol:-

$\overset{3}{4}\ \overset{2}{0}\ \overset{1}{7}\ \overset{0}{1}\ .\ \overset{-1}{1}\ \overset{-2}{2}\ \overset{-3}{5}$

$= 4 \times 8^3 + 0 \times 8^2 + 7 \times 8^1 + 1 \times 8^0 + 1 \times 8^{-1} + 2 \times 8^{-2} + 5 \times 8^{-3}$

$= 4 \times 512 + 0 + 56 + 1 + \frac{1}{8} + \frac{2}{\cancel{1}64} + \frac{5}{512}$

$= 2048 + 57 + 0.125 + 0.031 + 9.76 \times 10^{-3}$

$= 2105.156 + 9.76 \times 10^{-3}$

$= (2105.16576)_{10}$

$= (2105.166)_{10}$

Octal to Binary conversions :-

1. $(27.05)_8 \rightarrow (?)_2$

Sol:-

2 7 . 0 5

010 111 000 101

$(010111.000101)_2$

2. $(42.53)_8 \rightarrow (?)_2$

Sol:-

4 2 . 5 3

100 010 101 011

$(100010.101011)_2$

3. $(4078.258)_8 \rightarrow (?)_2$

Sol:-

4 0 7 8 . 2 5 8

100 000 111 000 010 101 000

$(100000111000.010101000)_2$

4. $(726.357)_8 \rightarrow (?)_2$

Sol:- 7 2 6 . 3 5 7

111 010 110 011 101 111

$(111010110.011101111)_2$

Octal to Hexa decimal conversion :-

1. $(27.05)_8 \rightarrow (?)_{16}$

Sol:- 2 7 . 0 5

010 111 000 101

$(17.11)_{16}$

2. $(42.53)_8 \rightarrow (?)_{16}$.

Sol:- 4 2 . 5 3

100 010 101 011

(22 . AB)

3. $(256.625)_8 \rightarrow (?)_{16}$

Sol:-

2 5 6 . 6 2 5

010 101 110 110 010 101

$= (AE . CA1)_{16}$.

4. $(764.325)_8 \rightarrow (?)_{16}$

Sol:- 7 6 4 . 3 2 5

111 110 100 . 011 010 101

$= (1F4.6A1)_{16}$.

Hexadecimal conversions :-

Hexadecimal to decimal conversions.

1. $(A3.62)_{16} = (?)_{10}$

Sol: $A \times 16^1 + 3 \times 16^0 + 6 \times \frac{1}{16} + 2 \times \frac{1}{16^2}$

$= 10 \times 16 + 3 \times 1 + 0.375 + 0.007$

$= (163.382)_{10}$

2. $(1B1.625)_{16} \rightarrow (?)_{10}$

Sol:- $1 \times 16^2 + B \times 16^1 + 1 \times 16^0 + 6 \times 16^{-1} + 2 \times 16^{-2} + 5 \times 16^{-3}$

$= 256 + B \times 16 + 1 \times 1 + \cancel{1}6 \times \frac{1}{16} + 2 \times \frac{1}{16^2} + 5 \times \frac{1}{16^3}$

$= 256 + B \times 16 + 1 + \frac{6}{16} + \frac{2}{256} + \frac{5}{4096}$

$= 256 + 11 \times 16 + 1 + 0.375 + 0.007 + 0.001$

$= (433.383)_{10}$

3. $(41.01)_{16} \longrightarrow (?)_{10}$

Sol:- $4 \times 16^1 + 1 \times 16^0 + 0 \times 16^{-1} + 1 \times 6^{-2}$

$4 \times 16 + 1 \cdot \times 1 + 0 + \frac{1}{16^2}$

$64 + 1 + 0.003$

$(65.003)_{10}$

Scanned with CamScanner

Hexadecimal to Binary convention:-

1. $(F02 \cdot 25)_{16} = (?)_2$

Sol:- F 0 2 . 2 5

1111 0000 0010 0010 0101

$(111100000010 \cdot 00100101)_2$

2. $(BAC \cdot 0A)_{16} = (?)_2$

Sol:-

B A C . 0 A

1011 1010 1100 0000 1010

$(101110101100 \cdot 00001010)_2$

3. $(EFC \cdot 1D)_{16} = (?)_2$

Sol:- E F C . 1 D

1110 1111 1100 0001 1101

$(111011111100 \cdot 00011101)_2$

4. $(A10F \cdot 18)_{16} \rightarrow (?)_2$

Sol:-

A 1 0 F . 1 8

1010 0001 0000 1111 0001 1000

$(1010000100001111 \cdot 00011000)_2$

5. $(456 \cdot AC)_{16} \rightarrow (?)_2$

Sol:- 4 5 6 . A C

0100 0101 0110 1010 1100

$(010001010110 \cdot 10101100)_2$

Hexadecimal to Octal :-

1. $(F02.25)_{16} \rightarrow (?)_8$

Sol:- F 0 2 . 2 5

111 1 0000 0010 0010 0101

7 4 0 2 1 1 1

$(7402.111)_8$

2. $(BAC.0A)_{16} \rightarrow (?)_8$

Sol:- B A C . 0 A

1011 1010 1100 0000 1010

5 6 5 4 0 2 2

$(5654.022)_8$

3. $(EFC.1D)_{16} \rightarrow (?)_8$

Sol:- E F C . 1 D

1110 1111 1100 0001 1101

7 3 7 4 0 7 1

$(7374.071)_8$

4. $(A10F.18)_{16} \rightarrow (?)_8$

Sol:- A 1 0 F . 1 8

1010 0001 0000 1111 0001 1000

1 2 0 4 1 7 0 6 0

$(120417.060)_8$

Scanned with CamScanner

5. $(456 \cdot AC)_{16} \rightarrow (?)_8$.

Sol:-

4 5 6 . A C

0100 0101 0110 . 1010 1100

2 1 2 6 . 5 3 0

$(2126 \cdot 530)_8$

Binary Addition:-

1.
  1 1 0 1
+ 0 1 0 1
(1) (1)
1 0 0 1 0

2.
  1 1 1 0
+ 0 0 1 0
(1) (1)
1 0 0 0 0

3.
  1 1 1 1
+ 0 1 0 1
(1) (1) (1)
1 0 1 0 0

4.
  1 1 1 1
+ 0 1 1 1
(1) (1) (1)
1 0 1 1 0

5.
1 1 0 1
0 1 0 0
1 1 1 1
1 1
0 0

5.
  1 1 0 1
1 0 1 0 0
1 0 0 0 1

  1 0 0 0 1
1 1 1 1 1
1 1 1
1 0 0 0 0 0

Perform binary addition to the given number.

i) 257.01 + 582

ii) 582

Binary Subtraction :-

```
   1 0 0 0        1 0 0        1 0 0 1
 - 0 0 1 0      - 0 0 1            1 1
 ---------      -------      ---------
   0 1 1 0        0 1 1        0 1 1 0
 ---------      -------      ---------
```

```
   1 1 1 1        1 1 0 0
 - 0 1 0 1        0 0 1 1
 ---------      ---------
   1 0 1 0        1 0 0 1
 ---------      ---------
```

→ Subtract 10 from 1101

```
   1 1 0 1
 -     1 0
 ---------
   1 0 1 1
 ---------
```

Binary Multiplication :-

```
    1 0 1          1 1 0 1
   x  1 0           x  1 1
  -------       ----------
    0 0 0          1 1 0 1
  1 0 1 X        1 1 0 1 X
  -------       ----------
  1 0 1 0      1 0 0 1 1 1
  -------       ----------
```

→ Multiplying decimal 14 with 8 by converting into binary equalent form

```
          1 1 1 0
       x  1 0 0 0
       ----------
          0 0 0 0
        0 0 0 0 X
      0 0 0 0 X X
    1 1 1 0 X X X
  ---------------
    1 1 1 0 0 0 0
```

In Binary compliments we have 1's/1st compliment and 2's compliment method to apply for binary subtraction. Here, compliment means writing 0's to 1's and 1's for 0. Which is called as One's compliment and adding one to One's compliment is 2's compliment.

1'c / 1's Compliment :-

Ex :- 1100
1'c 0011

Ex :- 1011
1'c 0100

Ex :- 1110110111
0001001000

2'c / 2's Compliment :-

2'c compliment of a binary number is obtained by adding 1 to LSB of 1's compliment of a number.

Examples :-

```
    1011            110101            11011 1011
1's 0100       1's  001010       1's  0010000100
       1                 1                      1
    ----            ------            ----------
    0101            001011            0010000111

  110110          11101101            1001100
  001001          00010010            0110011
       1                 1                  1
  ------          --------            -------
  001010          00010011            0110100
```

Number	1's compliment	2's compliment
→ 2 | 46 2 | 23 − 0 2 | 11 − 1 2 | 5 − 1 2 | 2 − 1 1 − 0 101110	For 10110. 1's 010001	010001 + 1 ——— 010010
→ 2 | 26 2 | 13 − 0 2 | 6 − 1 2 | 3 − 0 1 − 1 11010	00101	00101 + 1 ——— 00110
→ 2 | 31 2 | 15 − 1 2 | 7 − 1 2 | 3 − 1 1 − 1 11111	00000	00000 + 1 ——— 00001
→ 2 | 17 2 | 8 − 1 2 | 4 − 0 2 | 2 − 0 1 − 0 10001	01110	01110 + 1 ——— 01111
→ 2 | 13 2 | 6 − 1 2 | 3 − 0 1 − 1 1101	0010	0010 + 1 ——— 0011

→ 2 | 36
2 | 18 - 0
2 | 9 - 0
2 | 4 - 1
2 | 2 - 0
| 1 - 0

100100

011011

```
 0011011
     + 1
 -------
  011100
```

Substract :-

37 - 25: 37 → 100101, 25 → 11001

```
    100101
(-) 011001
    ------
    001100
```

2 | 37
2 | 18 - 1
2 | 9 - 0
2 | 4 - 1
2 | 2 - 0
| 1 - 0
100101

2 | 25
2 | 12 - 1
2 | 6 - 0
2 | 3 - 0
| 1 - 1
11001

42 - 18: 42 → 101010, 18 → 10010

```
    101010
(-) 010010
    ------
     11000
```

46 - 12: 46 → 101110, 12 → 1100

```
    101110
      1100
    ------
    100010
```

```
  46
- 12
----
  34
```

2 | 34
2 | 17 - 0
2 | 8 - 1
2 | 4 - 0
2 | 2 - 0
| 1 - 0

50 - 29: 50 → 110010, 29 → 11101

```
    110010
(-)  11101
    ------
      0101
```

Scanned with CamScanner

Subtraction of Smaller number from Large Num using One's Compliment :-

Step-1 : Find the one's compliment of smaller number.
Step-2 :- Add the one's compliment no. to the larger no.
Step-3 :- If any carry exist that may add carry to the,

Example1:-

```
  1 1 1  -> 7
  1 0 1  -> 5
  -----
  -----
```

step-1

```
    1 1 1
  + 0 1 0  -> is compliment of 5
  -------
```

No is positive 1)0 0 1

step-2

```
    └──────> 1
  -------
    0 1 0  Actual Result.
  -------
```

Example 2: $(14)_{10} - (7)_{10}$

```
  1 1 1 0  -> 14
  0 1 1 1  -> 7
  -------
  -------
```

step-1

```
    1 1 1 0
  + 1 0 0 0  -> 1's Compliment of 7
  ---------
```

Carry ↙ 1)0 1 1 0

step-2

```
    └────────> 1
   +
  ---------
    0 1 1 1  Actual Result.
  ---------
```

Scanned with CamScanner

Example 3 : $(36)_{10} - (16)_{10}$

```
2 | 36
2 | 18 - 0
2 | 9  - 0
2 | 4  - 1
2 | 2  - 0
    1  - 0
```

```
2 | 16
2 | 8 - 0
2 | 4 - 0
2 | 2 - 0
    1 - 0
```

36 - 100100
16 - 10000

```
  1 0 0 1 0 0
    1 0 0 0 0
  -----------
      0 1 0 0
```

```
Step-1 :-   1 0 0 1 0 0
          + 0 1 1 1 1 1   → 1's compliment of 16.
          -------------
Carry ↙   1)1 0 0 0 1 1
Step-2 :-   └──────────→ 1
          -------------
              1 0 1 0 0   — Actual Result.
```

Example 4: $(1125)_{10} - (135)_{10}$

```
2 | 1125
2 | 562 - 1
2 | 281 - 0
2 | 140 - 1
2 | 70  - 0
2 | 35  - 0
2 | 17  - 1
2 | 8   - 1
2 | 4   - 0
2 | 2   - 0
    1   - 0
```

```
2 | 135
2 | 67 - 1
2 | 33 - 1
2 | 16 - 1
2 | 8  - 0
2 | 4  - 0
2 | 2  - 0
    1  - 0
```

1125 - 10001100101

135 - 10000111

```
  10001100101
  00001111000
  -----------
  10011011101
```

```
   10001100101
   11101111000
  ------------
  101111011101
   └─────────→1
  ------------
   01111011110
```

Subtraction of Larger number from Smaller :-

Step-1 :- Find the one's compliment of a larger number

Step-2 :- Add one's compliment to the smaller number.

Step-3 :- To get the answer in true form takes one's compliment of the answer and asign negative sign.

Example:- 7 − 14.

111 − 1110

Step-1

```
 0001
 0111
 0001
 ----
 1000
```

1st compliment

−(0111)

Example :- 2

8−16 — 8 − 1000, 16 − 10000 ⇒

```
 01000
 10000  1's
```

```
 01000
 01111
 -----
 10111
```

1's compliment → −(01000)

Example :- 3

5−14 — 5−0101, 14−1110 → 1's ⇒

```
 0101
 0001
 ----
 0110
```

1's compliment → −(1001)

Example :- 4

10010 − 11010 ⇒

```
 10010
 00101
 -----
 10111
```

1's compliment → −(01000)

Example :- 5

220−450 — 220 − 11011100, 450 − 111000010 ⇒

```
 011011100
 111000010
```

```
 011011100
 000111101
 ---------
 100011001
```

−(011100110) ← 100011001

$(101011)_2 - (111001)_2$

1's compliment of $(111001)_2 \rightarrow 000110$

$$\begin{array}{r} 101011 \\ +\ 000110 \\ \hline 110001 \end{array}$$

1's compliment of $110001 \rightarrow -(001110) \rightarrow -14$

* subtraction Using 2's compliment :-

case(i) Smaller number from Larger number :-

(i) $(14)_{10} - (7)_{10}$

$14 \rightarrow 1110$

$7 \rightarrow 0111$

2's compliment of 7

1's comp of $7 \rightarrow 1000$

$$\begin{array}{r} 1000 \\ 0001 \\ \hline \end{array}$$

2's comp of $7 \rightarrow 1001$

$$\begin{array}{r} 1110 \\ 1001 \\ \hline \boxed{1}\,0011 \end{array} = (7)_{10}$$

2) subtract $(27)_{10}$ from $(43)_{10}$

$$\begin{array}{r|l} 2 & 27 \\ \hline 2 & 13 - 1 \\ \hline 2 & 6 - 1 \\ \hline 2 & 3 - 0 \\ \hline & 1 - 1 \end{array} \qquad \begin{array}{r|l} 2 & 43 \\ \hline 2 & 21 - 1 \\ \hline 2 & 10 - 1 \\ \hline 2 & 5 - 0 \\ \hline 2 & 2 - 1 \\ \hline & 1 - 0 \end{array}$$

$(011011)_2$ $\qquad$ $(101011)_2$

1's compliment of 27 → 100100

→ 00001

2's compliment of 27 100101

$$\begin{array}{r} 101011 \\ 100101 \\ \hline 1]\,010000 \end{array} = (16)_{10}$$

3) $(000.101)_2$ from $(111.0101)_2$

1's compliment of 0000101 → 1111010

00000 01

2's compliment → 1111011

$$\begin{array}{r} 111111 \\ \hline 1110101 \\ +\ 1111011 \\ \hline 1]\,1110000 \end{array} = (112)_{10}$$

neglect

64 32 16 8 4 2 1

Procedure :- (L.N − S.N)

⇒ subtraction of smaller number from Larger number by using 2's compliment.

→ Find 2's compliment of smaller number

→ Add 2's compliment to the Larger number

→ If there is any carry, discard the carry.

1) $(26)_{10} - (36)_{10}$ using 2's compliment

2	36	
2	18	0
2	9	0
2	4	1
2	2	0
	1	0

2	26	
2	13	0
2	6	1
2	3	0
	1	1

$(100100)_2 \rightarrow (36)_{10}$

2's compliment of 36 ⇒
$$\begin{array}{r} 011011 \\ 000001 \\ \hline 011100 \end{array}$$

$$\begin{array}{r} 011010 \\ +\ 011100 \\ \hline 110110 \end{array}$$

2's compliment of result ⇒
$$\begin{array}{r} 001001 \\ +\ \quad 1 \\ \hline -(001010) \end{array}$$

$-(10)_{10}$

2) $(57)_{10} - (98)_{10}$

2	57	
2	28	1
2	14	0
2	7	0
2	3	1
	1	1

$(111001)_2$

2	98	
2	49	0
2	24	1
2	12	0
2	6	0
2	3	0
	1	1

$(1100010)_2$

1's compliment of 98 → 0011101

$$\begin{array}{r} 0011101 \\ +\quad 1 \\ \hline 0011110 \end{array}$$

2's compliment → 0011110

$$\begin{array}{r} 0011110 \\ 111001 \\ \hline 1010111 \end{array}$$

2's compliment of result → $-(0101000)_2$

$$\begin{array}{r} +\quad 1 \\ \hline \end{array}$$

$= -(0101001)_2$

* subtraction of Larger from smaller using 2's compliment

step-1 :- Find the 2's compliment of Larger number

step-2 :- Add 2's compliment to the smaller number.

step-3 :- Find the 2's compliment to the result, to get answer in true form take negative sign.

$$\begin{array}{r} 9 \\ -8 \\ \hline 1 \end{array}$$

9's compliment and 10's compliment :-

Find 9's complement and 10's compliment of $(45)_{10}$

9's compliment $\Rightarrow$ $\begin{array}{r} 99 \\ -45 \\ \hline 54 \end{array}$

10's complement = 9's complement + 1

= 54 + 1

= 55

Find 9's complement and 10's complement for $(9356)_{10}$

9's complement $\Rightarrow$ $\begin{array}{r} 9999 \\ -9356 \\ \hline 643 \end{array}$

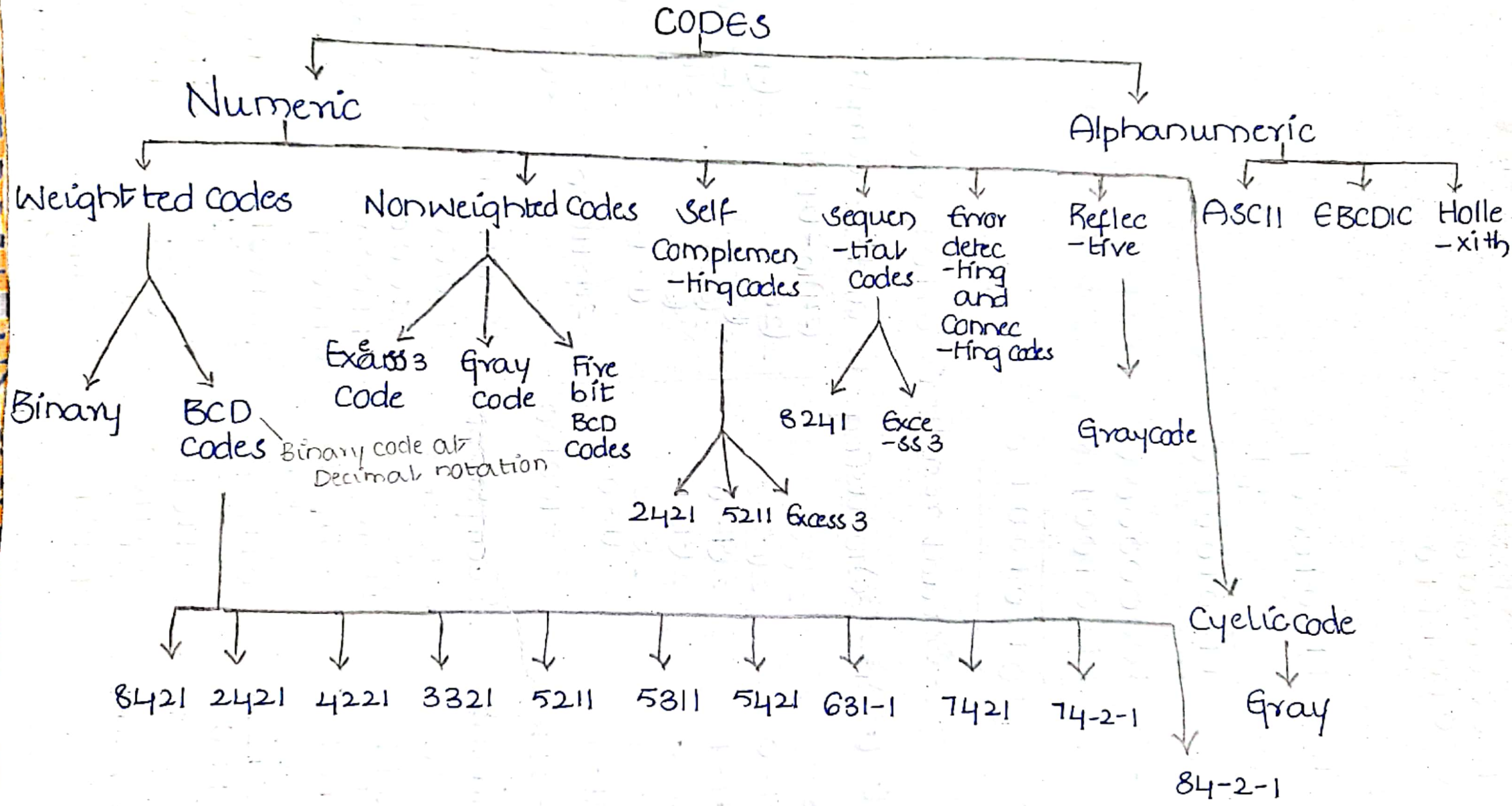
10's complement = 9's comp + 1

= 643 + 1

= 644

CODES

CLASSIFICATION OF CODES.



Alphanumeric Codes:-

The codes which consist of both . The most commonly used codes are ASCII (American standard code

Weighted codes:-

Weighted codes are classified into two categories

1. positive weighted code and. Ex:- 8421, 2421, 4221, 3321, 5211, 5311
2. Negative weighted code Ex:- 631-1, 72-2-1, 84-2-1.

Decimal Number	8421	2421
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	1011
6	0110	1100
7	0111	1101
8	1000	1110

8421 BCD code is widely used and it is common practice to refer to it simply as BCD code. In this code each decimal digi-t 0 to 9 is code by fourbit binary number it is also called natu-ral binary code because 8,4,2, and 1 weights are attached to it. And it is a weighted code and it is also called sequential code. The main advantage of this code is of conversion from 2end from decimal and binary.

Note:- It is less efficient then the pure binary in the scense that it require more bits.

For Example:- the decimal no. 14 can be represented as 1110 in pure binary but the same 14 in BCD can be prepresented as

14		25		84	
0001	0100	0010	0101	1000	0100

Disadvantages:-

The Arthmetic opeartions are more complex then they are in pure binary as there are six illegeal combination and they are not part of 8421 BCD code system.

- 8421 as rules of binary addition and subtraction which donot apply for entire 8421 number but only to the individ-ual fourbit Groups.

BCD Addition:-

1. If a decimal number is given that each digit of decimal number should be represented in binary coded BCD Form.
2. performing addition.
3. Verifying the sum. If sum if equal to (or) less then 9 no correction is needes and the sum is your actual output.
4. If it is Greater than 9 (or) if any carry is Generated from fourbit sum then the sum is invalid.
5. To correct (or) To make a Valid sum add 6 to the fourbit sum. If a carry is Generated from the addition add to the next higher BDC digit.
6. After this you will get actual result.

Decimal number	8-4-2-1 Binary code
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

⇒ Write BCD code for $(68)_{10}$

1) $(68)_{10}$

0110 1000

2) $(245)_{10}$

2	4	5
0010	0100	0101

* BCD addition:-

→ sum ≤ 9 with carry 0 (valid sum)

→ sum > 9 with carry 0 } Invalid sum → Add 6.

→ sum ≤ 9 with carry 1 }

case (i)

sum ≤ 9 with carry 0

(i)
$$\begin{array}{r} 6 \\ +3 \\ \hline 9 \end{array} \qquad \begin{array}{r} 0110 \\ +\ 0011 \\ \hline 1001 \end{array} \rightarrow 9$$

(ii)
$$\begin{array}{r} 5 \\ +2 \\ \hline 7 \end{array} \qquad \begin{array}{r} 0101 \\ +\ 0010 \\ \hline 0111 \end{array} \rightarrow 7$$

case (i)

Sum > 9 with carry 0

$$\begin{array}{r} 6 \\ +8 \\ \hline 14 \end{array} \qquad \begin{array}{r} 0110 \\ +\ 1000 \\ \hline 1110 \\ \hline 14 \end{array} \qquad \begin{array}{r} 1110 \\ +\ 0110 \\ \hline 0001\ 0100 \\ 1 \quad\ \ 4 \end{array}$$

0001 0100

$$\begin{array}{r} 7 \\ +\ 8 \\ \hline 15 \end{array} \qquad \begin{array}{r} 0111 \\ +\ 1000 \\ \hline 1111 \\ \hline 15 \end{array} \qquad \begin{array}{r} 111 \\ 1111 \\ 0111 \\ \hline 1\,]\,0110 \end{array} \qquad \begin{array}{r} 11 \\ 1111 \\ 0110 \rightarrow 6 \\ \hline 1\,]\,0101 \\ \hline 1 \quad 5 \end{array}$$

0001 0101

→ If the sum is invalid, add '6' to get the valid sum

case(iii):-

sum ≤ 9 with carry 1 (invalid sum)

$$\begin{array}{r} 8 \\ +\ 9 \\ \hline 17 \end{array} \qquad \begin{array}{r} 1000 \\ +\ 1001 \\ \hline 1\,]\,0001 \end{array} \qquad \begin{array}{r} 0001\,0001 \\ 0000\,0110 \\ \hline 0001\,0111 \end{array}$$

→ $(24)_{10} + (18)_{10}$

$$\begin{array}{r} 1 \\ \hline 24 \\ +\ 18 \\ \hline 42 \end{array} \rightarrow \begin{array}{r} 0010\ 0100 \\ 0001\ 1000 \\ \hline 0011\ 1100 \\ \hline 3 \quad\ 12 \end{array} \qquad \begin{array}{r} 1111 \\ \hline 0011\,1100 \\ +\ 0000\,0110 \\ \hline 0100\,0010 \\ \hline 4 \qquad 2 \end{array}$$

→ Perform addition on

$(58)_{10} + (18)_{10}$ in 8421 digit

```
          11      
 58  →  0101 1000          0111 0000
+18  →+ 0001 1000          0000 0110
 76     0111 0000          0111 0110
                             7    6

          1111
→ 65 →  0110 0101          1011 1101
+ 58    0101 1000          0000 0110
 123    1011 1101          1100 0011
                            12    3

  1100 0011
  0110
0001 0010 0011
1     2    3
```

$\rightarrow (589)_{10} + (199)_{10}$

```
  11          11    11    1
  5 8 9  →  0101  1000  1001
+ 1 9 9     0001  1001  1001
  7 8 8     0111  0010  0010
              7     2     2

                    11
  0111  0010  0010
+ 0000  0000  0110
  0111  0010  1000
+ 0000  0110  0000
  0111  1000  1000
    7     8     8
```

Scanned with CamScanner

* Step-1 :-
Add two BCD numbers using ordinary addition

* Step-2 :-
If 4 bit sum ≤ 9. no correction is needed
The sum is in proper BCD form.

* Step-3 :-
If 4 bit sum is greater than 9 (or) carry is generated from 4 bit sum, then the sum is invalid.

* Step-4 :-
To correct the invalid sum add 6 to the 4-bit sum. $(0110)_2 - (6)_{10}$

Step-5 :-

1)
$$\begin{array}{r} 113 \\ +\ 101 \\ \hline 214 \end{array} \qquad \begin{array}{r} 0001\,0001\,0011 \\ +\ 0001\,0000\,0001 \\ \hline \underbrace{0010}_{2}\,\underbrace{0001}_{1}\,\underbrace{0100}_{4} \end{array}$$

* BCD Substraction :-

Subtraction using 9's complement :-

→ Find the 9's complement of the negative number

→ Add 9's complement to the minuend.

→ If an carry add it to the result

7 → minuend
- 5 → subrahend.
2

0111
0001

7
- 5
2 (9-5) = 4 ⇒ 0100
1011 → (11) invalid

1011
0110
1]0001
0001
0010

0111
0100
1011

1011
0110
1]0001
+ 0001
0010 ⇒ 2

→ $(9)_{10} - (5)_{10}$

9 → 1001
- 5 (9-5) = 4 0100
4 1101 → 13 (invalid)
+ 6

1101
add 6 → 0110
1]0011
0001
0100
↓
4

→ 79 → 0111 1001
- 26 (99-26) 0110 0011
53 = 73 1100 1100
12

1100 1100
0000 0110
1100 0010
0110
1]0100 0010
1
0100 0011
5 3

$\rightarrow (89)_{10} - (34)_{10}$

99
34
$\overline{65}$

8 9	$\rightarrow$	0000	1001
− 3 4		0011	0100
5 5		1011	1101

$(99 - 34) = 65$

1011	1101
	0110
1100	0001

1000	1001
0110	0101
1110	1110
	0110
1111	0100
0110	
1 0101	0100
	1
0101	0101
5	5

89 =
99-34 = 65 =>

1000	1001
0110	0101
1110	1110
	0110
1111	0100
0110	
1 0101	0100
	1
0101	0101
5	5

$\rightarrow$ perform $(24)_{10} - (56)_{10}$ in BCD using 9's complement

2 4	$\rightarrow$ 24	$\Rightarrow$ 24	$\Rightarrow$	0010	0100
− 5 6	$\rightarrow$ (99−56)	$\Rightarrow$ 43		0100	0011
− 3 2				0110	0111
				6	7

Scanned with CamScanner

9's complement of 67 ⇒ -(32)

1001
1
1010

* subtraction using 10's complement :-

step-1 :- Find the 10's complement of the negative numbers.

step-2 :- Add 2 numbers using addition 10's complement of positive number and negative number.

step-3 :- If any carry is generated neglect it otherwise if carry = 0 then take 10's complement of result.

10's comp = 9's comp + 1

$(8-2)_{10}$

1000
-0010

1000
1000
1 | 0000 → Invalid.
0110
0110 — 6

neglect carry

10's complement of result

1000
1000
1 | 0000 In

1 | 0000
0110
0110 → 6

(9-5)

1
1001
0101
1110 — 14
0110
1 | 0100

1000
0111
1101

10's = $\frac{9}{-2}$ 7+1 = 8

1000
1000
1 | 0000
0110
0110
6 —

Scanned with CamScanner

Perform subtraction $(88)_{10} - (64)_{10}$

$$\begin{array}{r} 9 \\ -4 \\ \hline 5 \end{array}$$

~~0101~~
1011

$$\begin{array}{rrl} 88 & & 10001000 \\ \underline{64} & + & 00110110 \\ \underline{24} & & \overline{10111110} \\ & & 0110 \\ & & \overline{11000100} \\ & & 0110 \\ & & \overline{\underbrace{0010}_{2}\underbrace{0100}_{4}} \end{array}$$

0110 1100 1000

$$\begin{array}{r} 999 \\ 604 \\ \hline 395 \end{array}$$

Non-weighted codes :-

There are two types of non-weighted codes.

They are.

1) Excess-3 code (XS-3)

2) Gray code

Excess-3 code = 3 + BCD code

Ex:- = 0011 + BCD code

Excess-3code of

0010 is 0101

Decimal	8-4-2-1 code	2-4-2-1 code	Excess-3
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	0101	1000
6	0110	0110	1001
7	0111	0111	1010
8	1000	1110	1011
9	1001	1111	1100

complements

→ also called as self complementary / reflective code

→ It is a non weighted code where each bit of the code does not have any weight.

→ It is modified form of the BCD code.

→ It is obtained by adding 3 (i.e) (0011) to each digit of the decimal number.

Find the Excess-3 code (XS-3)

$(12)_{10}$

0001	0010
0011	0011
0100	0101
(4	5)

```
  1100
+ 0011
  1111
```

Find the excess-3 code $(592)_{10}$ & $(403)_{10}$

0101	1001	0010
0011	0011	0011
1000	1100	0101
8	12	5

403

0111 0001 0110

Binary Representation of Signed numbers

There are two ways to represent the signed numbers.

1) sign magnitude form $\rightarrow$ $(-(2^{n-1}-1)$ to $+(2^{n-1}-1))$

2) complement form. $\rightarrow$ 1's complement $\rightarrow$ $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

2's Complement $\rightarrow$ -2^{n-1} to $+(2^{n-1}-1)$

sign-magnitude form:-

For unsigned Binary no's

Range = $(0$ to $2^n-1)$.

Ex:-

$4 \rightarrow 100$

$+4 \rightarrow \boxed{0 \mid 100}$

$-4 \rightarrow \boxed{1 \mid 100} \rightarrow$ magnitude

MSB/sign bit

$\rightarrow$ In sign-magnitude we are adding one more extra bit on the MSB side to represent a number positive or negative

If MSB = 0, number is +ve

If MSB = 1, number is -ve

There are two types of complement forms

1) sign 1's complement form

2) sign 2's complement form.

Represent -5 in three ways.

i) sign-magnitude form.

ii) sign-1's complement form.

iii) sign-2's complement form.

1) sign-magnitude form

$$-5 - \underbrace{\overbrace{\boxed{1}}^{\text{sign bit}}\boxed{101}}_{\text{magnitude}}$$

2) $-5 - \boxed{1\,|\,010}$ sign 1's complement

3) $-5 - \boxed{1\,|\,011}$ sign 2's complement

In all forms +5, the code is not going to be changed i.e $\boxed{0\,|\,101}$

Express (-45) in

1) sign-magnitude, 1's complement & 2's complement form

2	45	
2	22	– 1
2	11	– 0
2	5	– 1
2	2	– 1
	1	– 0

~~$(110101)_2$~~

$(101101)_2$

sign - magnitude → 1 | 101101

sign - 1's comp → 1 | 010010

- 2's comp → 1 | 010011

For -51 in all the three forms with 8 bits

2	51
2	25 - 1
2	12 - 1
2	6 - 0
2	3 - 0
	1 - 1

$(110011)_2$

sign - magnitude → 1 | 0110011 for -51

sign - 1's comp of -51 → 1 | 1001100

sign - 2's comp of -51 → 1 | 1001101

Express -6 in 8 bits. in sign - magnitude form

0 | 0000110

Sign - magnitude of -6 → 1 | 0000110

1 | 0000110

sign 1's comp of -6 → 1 |

1 | 1111001

~~Sig~~

1 | 1111001

1 | 1111010

Sign 2's comp of -6 → 1 | 1111010

Range of numbers:-

$$-2^{n-1} \text{ to } 2^{n-1}-1$$

Where n = no. of bits

$n=4$ and $n=8$

$\boxed{n=4}$

$$\left[-2^{4-1} \text{ to } 2^{4-1}-1\right] = (-8 \text{ to } +7)$$

$\boxed{n=8}$

$$\left[-2^{8-1} \text{ to } 2^{8-1}-1\right] = (-128 \text{ to } +127).$$

→ Write the decimal number for the signed binary numbers. in 1) Sign magnitude form.
2) sign 1's comple form & 2's complefom

Binary number	sign - magnitude form	sign 1's comp	Sign 2's comp
01101	$\boxed{0 \mid 1101}$ = 13	+13	+13
010111	$\boxed{0 \mid \overset{16\,8\,4\,2\,1}{10111}}$ = 23	+23	+23

1101010 $\quad$ 32 16 8 4 2 1 | 1 | 101010 | (−42) $\quad$ 16 8 4 2 1 | 1 | 010101 | −21 $\quad$ 16 8 4 2 1 | 1 | 010110 | −22

→ write the binary numbers using 16 digits.

		sign-mag	sign-1's comp
a)	+ 1001010	0 | 000000001001010	0 | 111111110110101
b)	− 1111.0000	1 | 00000001111.0000	1 | 11111110000 1111

→ Represent the numbers in 8 bits

1) +4 2) −14, 3) +23, 4) −64 in all three forms.

* Gray code :-

Decimal number	8-4-2-1 code	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

→ It is also called unique distant code

1) +4

Sign-magnitude form → $\boxed{0 \mid 0000100}$

1's comp form of +4 → $\boxed{0 \mid 1111011}$

2's comp form of +4 → $\boxed{0 \mid 1111100}$

2) −14

Sign-magnitude form → $\boxed{1 \mid 0001110}$

1's comp form of −14 → $\boxed{1 \mid 1110001}$

2's comp form of −14 → $\boxed{1 \mid 1110010}$

3) +23

Sign-magnitude form of +23 → $\boxed{0 \mid 0010111}$

16 8 4 2 1

23 → 1 0 1 1 1

1's comp of +23 → $\boxed{0 \mid 1101000}$

2's comp of +23 → $\boxed{0 \mid 1101001}$

1) convert $(101101110)_2$ to gray code

→ gray code is a non-weighted code, it is also called as unique distance code. because the two adjacent numbers differ by 1 bit

Gray code → (111011001)

2) convert $(0001011010)_2$ to gray

gray code → 0001110111

3) convert Gray code (1011010) to Binary

1 0 1 1 0 1 0

⊕ ⊕ ⊕ ⊕ ⊕ ⊕

1 1 0 1 1 0 0

(1101100)

4) convert Grey to (8-4-2-1) binary

(0100101011)

0111001101 → Binary code

5) (1011101) → Gray

(1101001) → Binary

*Error detecting and correcting codes:-

When the binary information is transmitted from one circuit to the other circuit and error may occur due to the presence of noise. It means that there is a change from 0 to 1 (or) 1 to 0. In order to detect and correct errors we are using error detecting and correcting codes.

*Parity bit:-

A parity bit is an extra bit added to the binary message to make the number of ones either even or odd. The binary message including the parity bit is transmitted and checked at the reciever end for the errors.

* Even parity:- In Even parity the added bit should make the total ~~even~~ number of ones even.

* odd parity:- In odd parity the added bit should make the total number of ones odd.

Message	Message with even Parity	Message with odd Parity.
000	0	1
001	1	0
010	1	0
011	0	1
100	1	0
101	0	1
110	0	1
111	1	0

There are two types of codes to detect and correct the errors.

1) block parity → detecting error
2) Hamming code → detecting & correcting error

*Block parity:-

check the errors if any with even parity in the block

↓ parity column.

4x8

1 0 1 1	0 1 0 1	1 → no error
0 1 1 1	0 1 0 1	0 → error (2nd row).
1 1 0 0	1 1 0 1	1 → no error
1 0 0 1	1 0 1 1	1 → no error
Parity row → 1 0 1 1	0 1 1 0	1 → error

no no error (3) no error no no no

When binary information is transmitted, the bits are arranged in rows and columns. along with parity bit (when binary) for each row and column. forms the block parity

→ check the errors in block parity with odd parity

1	0	0	0	0	1	0	1	0	→ no error
0	1	0	1	0	1	0	1	1	→ no error
1	1	0	0	1	1	0	1	1	→ error (3)
1	0	0	1	1	0	1	1	0	→ no error
0	1	0	1	1	0	0	1	1	
↓ no error	↓ no error	↓ error (3)	↓ error (4)	↓ no error	↓ no error	↓ no error	↓ no error		

* Error detecting and correcting code:-

Hamming code:-

Encode the binary number 1011 into 7-bit Even parity hamming code

Msge bits / Information bits = 1011 = 4

$$\boxed{2^p \geq x+p+1}$$

$x \Rightarrow$ no. of information bits $= 4$

$p \Rightarrow$ no. of parity bits $= 3$

$p=1 \Rightarrow 2^1 \geq 4+1+1 \Rightarrow 2 \geq 6$

$p=2 \Rightarrow 2^2 \geq 4+2+1 \Rightarrow 4 \geq 7$

$p=3 \Rightarrow 2^3 \Rightarrow 4+3+1 \Rightarrow 8 \geq 8$

Total bits = information bits + parity bits

$= 4+3$

$= 7$ bits (hamming code)

Step-2:-

Locating the parity bits. in ascending powers of '2'

$2^0 = 1 = P_1$

$2^1 = 2 = P_2$

$2^2 = 4 = P_4$

7	6	5	4	3	2	1
D_7	P_6	D_5	P_4	D_3	P_2	P_1

Step-3:-

Bit position table

Bit designation	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit location	7	6	5	4	3	2	1
Binary number	111	110	101	100	011	010	001
Information bits	1	0	1	-	1	-	-
Parity bits				0		0	1

$P_1 = 1, 3, 5, 7 = 111 = 1$

$P_2 = 2, 3, 6, 7 = 101 = 0$

$P_4 = 4, 5, 6, 7 = 101 = 0$

7-bit haming code = 1010101

→ Encode the binary 1110 into 7-bit odd Parity hamming code

Steps:-

1) $$2^p \geq x + p + 1$$

$x \rightarrow$ Data bits = 4

$p \rightarrow$ parity bits = 3

$2^1 \geq 4+1+1 \Rightarrow 2 \geq 6$

$2^2 \geq 4+2+1 \Rightarrow 4 \geq 7$

$2^3 \geq 4+3+1 \Rightarrow 8 \geq 8$

2) Locating the parity bits

D_7	D_6	D_5	P_4	D_3	P_2	P_1
7	6	5	4	3	2	1

3) bit position table.

Bit designation	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit location	7	6	5	4	3	2	1

Binary number	111	110	101	100	011	010	001
Information bits.	1	1	1	–	0	–	–
Parity bits				0		1	1

$P_1 \rightarrow P_1, 3, 5, 7 \rightarrow 0\,1\,1 \rightarrow 1$

$P_2 \rightarrow 2, 3, 6, 7 \rightarrow 0\,1\,1 \rightarrow 1$

$P_4 \rightarrow 3, 4, 5, 6, 7 \rightarrow \cancel{0}\,1\,1 \rightarrow 0$

7-bit hamming code

1110011

→ Encode the binary number 11001 into 9-bit odd parity hamming code

1) . $x = 5$. $2^4 \geq 5 + 4 + 1$, $16 \geq 10$

2) Locating the parity bits

$D_9\ P_8\ D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$

3) bit position table

Bit designation	D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit location	9	8	7	6	5	4	3	2	1

Binary number 1001 1000 0111 0110 0101 0100 0011

Data bits 1 - 1 0 0 - 1 0010

Parity bits 0 0 1

$P_1 \rightarrow 1011 \rightarrow 0$

$P_2 \rightarrow 1101 \rightarrow 1$

$P_4 \rightarrow 0001 \rightarrow 0$

$P_8 \rightarrow 0001 \rightarrow 0$

hamming code → 101000 110

→ Determine the single error correction code [hamming code] for the information 10111 for odd parity.

→ Determine hamming code for information bits 11000 for even parity

→ Assume that even parity hamming code 0110011 which is transmitted and 0100011 which is recieved. The reciever does not know what was transmitted. Determine ~~will~~ bit location where error has occured in recieved code

Bit location table

Bit designation.	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit location	7	6	5	4	3	2	1
Binary number	111	110	101	100	011	010	001
Data bits				-		-	-
parity bits							
Recived code	0	1	0	0	0	1	1

LSB

$P_1 \Rightarrow 1,3,5,7 \rightarrow 1000 \rightarrow$ wrong $\rightarrow 1$

$P_2 = 2,3,6,7 \rightarrow 1010 \rightarrow$ Correct $\rightarrow 0$

$P_4 = 4,5,6,7 \rightarrow 0010 \rightarrow$ wrong $\rightarrow 1$

MSB

101 = 5

Error is at 5^{th} bit

Replace '0' with 1

0110011

→ Hamming code 101101101 is recieved, correct it if any errors. there are four parity bits and odd parity is used.

Bit Location table

Bit designation	D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit position	9	8	7	6	5	4	3	2	1
Binary number	1001	1000	0111	0110	0101	0100	0011	0010	0001
Recieved code	1	0	1	1	0	1	1	0	1

$P_1 \rightarrow 1, 3, 5, 7, 9 \rightarrow 11011 \rightarrow 1$

$P_2 \rightarrow 2, 3, 6, 7, \rightarrow 0111 \rightarrow 0$

$P_4 \rightarrow 4, 5, 6, 7 \rightarrow 1011 \rightarrow 0$

$P_8 \rightarrow 8, 9 \rightarrow 01 \rightarrow 0$

$0001 \rightarrow 1^{st}$ position.

Replace '1' with '0'.

$101101100 \rightarrow$ corrected code

→ The recieved code is 1001001 check for errors with Even parity

Bit designation	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit position	7	6	5	4	3	2	1
Binary no.	111	110	101	100	011	010	001
Recieved code	1	0	0	1	0	0	1

$P_1 \rightarrow 1,3,5,7 \rightarrow 1001 \rightarrow 0$

$P_2 \rightarrow 2,3,6,7 \rightarrow 0001 \rightarrow 1$

$P_4 \rightarrow 4,5,6,7 \rightarrow 1001 \rightarrow 0$

$010 \rightarrow 2$

Error at 2^{nd} position

$1001011 \rightarrow$ correct hamming code

→ Recieved code is 0111001.

check for Error with odd parity

bit desi	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit pos	7	6	5	4	3	2	1
Bi no	111	110	101	100	011	010	001
Recieved code	0	1	1	1	0	0	1

$P_1 \rightarrow 1, 3, 5, 7 \rightarrow 1010 \rightarrow 1$

$P_2 \rightarrow 2, 3, 6, 7 \rightarrow 0010 \rightarrow 0$

$P_4 \rightarrow 4, 5, 6, 7 \rightarrow 1110 \rightarrow 0$

$001 \rightarrow 1^{st}$ position

corrected code $\rightarrow$ 0111000

$\rightarrow$ 101011011101, Even parity

Bit des.	D_{12}	D_{11}	D_{10}	D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit pos	12	11	10	9	8	7	6	5	4	3	2	1
Bin. no	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Rec code	1	0	1	0	1	1	0	1	1	1	0	1

$P_1 \rightarrow 1, 3, 5, 7, 9, 11 \rightarrow 111100 \rightarrow 0$

$P_2 \rightarrow 2, 3, 6, 7, 10, 11 \rightarrow 010110 \rightarrow 1$

$P_4 \rightarrow 4, 5, 6, 7, 12 \rightarrow 11011 \rightarrow 0$

$P_8 \rightarrow 8, 9, 10, 11, 12 \rightarrow 10101 \rightarrow 1$

$1010 \rightarrow 10^{th}$ position

1000110111101 → corrected code.

* Floating point representation :-

Floating point representation is classified into three fields.

1) Sign
2) Exponent
3) Mantissa

IEEE has established stad. rep. of floating pt. no's.

The standard representation of binary number is

$\pm 1.M \times 2^{E}$

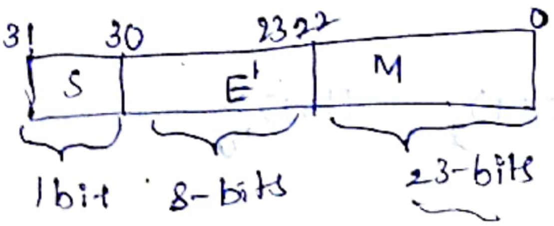
M → mantissa , E → Exponent

There are two types of representations.

1) 32-bit representation IEEE-754 (single precision floating pt standard uses 8-bit Exponent with bias 127) & 23-bit significand.
2) 64-bit representation (IEEE-754 double precision stand. uses 11-bit exponent (1023) & a 52-bit significand

→ As the binary point is variable and automatically adjusted. Therefore the binary point is said to be floating and the numbers are called floating point numbers.

* 32-bit representation :-



S → sign 1

E → Exponent 8

M → Mantissa 23

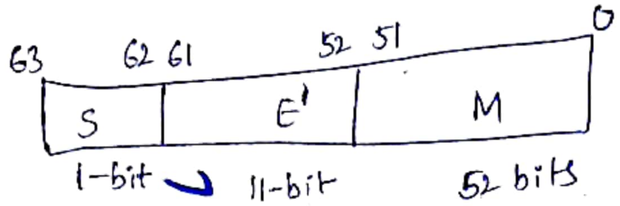
$\frac{24}{8}{32}$

$E' = E + 127$

$n = 8$

$2^{n-1} - 1 = 2^{8-1} - 1 = 2^{7} - 1 = 128 - 1 = 127$

2. 64-bit Representation:-



$E' = E + 1023$ $\qquad 2^{n-1} - 1 = 2^{11-1} - 1 = 2^{10} - 1 = 1023$

Represent $(10001.001)_2$ in 32-bit & 64-bit representati.

10001.001

$= 1.0001001 \times 2^4$

$= \pm 1.M \times 2^E$

$\boxed{M = 0001001}$, $\boxed{E = 4}$

$E' = E + 127 = 131$

$S = 0$

2 | 131
2 | 65 - 1
2 | 32 - 1
2 | 16 - 0
2 | 8 - 0
2 | 4 - 0
2 | 2 - 0
1 - 0

$(10000011)_2 = (131)_{10}$

0	10000011	00010010000000000000000000

64-bit

$E' = E + 1023 = 4 + 1023 = 1027$

2	1027
2	513 − 1
2	256 − 1
2	128 − 0
2	64 − 0
2	32 − 0
2	16 − 0
2	8 − 0
2	4 − 0
2	2 − 0
	1 − 0

$(1000,0000,011)_2$

63 62 61 52 51 0

0	10000000011	0001001 45 0's
1-bit	11-bit	52-bits

(Q) Represent $(12.5)_{10}$ in floating point 32, 64-bit rep

2	12.
2	6 − 0
2	3 − 0
	1 − 1

$0.5 \times 2 = 1.0 \rightarrow 1$

$(1100.1)_2$

$(1100)_2$

$= \boxed{1.1001 \times 2^3}$

$M = 1001$, $\boxed{E = 3}$

$E' = E + 127$
$= 3 + 127$
$= 130$

0	10000010	1001 19 0's

$$
\begin{array}{r|l}
2 & 130 \\
2 & 65 - 0 \\
2 & 32 - 1 \\
2 & 16 - 0 \\
2 & 8 - 0 \\
2 & 4 - 0 \\
2 & 2 - 0 \\
 & 1 - 0
\end{array}
$$

64-bit :-

$E' = 3 + 1023 = 1026$

$$
\begin{array}{r|l}
2 & 1026 \\
2 & 513 - 0 \\
2 & 256 - 1 \\
2 & 128 - 0 \\
2 & 64 - 0 \\
2 & 32 - 0 \\
2 & 16 - 0 \\
2 & 8 - 0 \\
2 & 4 - 0 \\
2 & 2 - 0 \\
 & 1 - 0
\end{array}
$$

$(10000000010)_2$

0	10000000010	1001 48 0's

→ $(11.1000.11.00.110)_2$

32-bit representation :-

$= 1.1100011000110 \times 2^9$

$\Rightarrow$ $\boxed{\text{Mantissa} = 11000.1100110}$, $\boxed{\text{Exponent} = 9}$

$E' = E + 127 = 9 + 127 = 136$

2	136	
2	68	0
2	34	0
2	17	0
2	8	1
2	4	0
2	2	0
	1	0

$(10001000)_2$

0	10001000	110001100110 11	10 0's bits

* 64-bit Representation :-

$E' = E + 1023 = 9 + 1023 = 1032$

2	1032	
2	516	0
2	258	0
2	129	0
2	64	1
2	32	0
2	16	0
2	8	0
2	4	0
2	2	0
	1	0

$(10000001000)_2$

0	10000001000	11000110000110	40 0's

Range

1) unsigned number $\rightarrow$ 0 to $2^n - 1$ = 0 to 15

Min Max

for $\boxed{n = 4 \text{ bits}}$

2) signed numbers :-

a) sign magnitude representation $\rightarrow$

$-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$ = -7 to $+7$

b) signed 1's complement representation

$\rightarrow$ $-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$ = -7 to $+7$

c) signed 2's complement representation

$\rightarrow$ $-(2^{n-1})$ to $+2^{n-1} - 1$ = -8 to $+7$

* overflow :-

* overflow never occurs if the two numbers are having different signs.

9 − 5 = 9 + (2's comp of 5)

$$\begin{array}{r} 9 \\ -5 \\ \hline 4 \end{array} \rightarrow \begin{array}{r} 1001 \\ 1011 \\ \hline 0100 \end{array}$$

1011 $\rightarrow$ 2's comp of 5

0100 $\rightarrow$ +4 (no overflow)

$$\begin{array}{rr} & \overset{1\,1}{111} \\ +7 \rightarrow & 111 \\ +7 & 111 \\ \hline & 1\,]\,1\,1\,0 \end{array} \rightarrow \text{overflow}$$

-ve number.

$$\begin{array}{rr} +6 \rightarrow & 110 \\ +5 \rightarrow & 101 \\ \hline & 1\,]\,0\,1\,1 \end{array} \rightarrow \text{overflow}$$

+ve number

0111

1001

$-7 \rightarrow$ 2's comp of $7 \rightarrow 1001$

$-7 \rightarrow$ 2's comp of $7 \rightarrow +1001$

$\overline{-14}$

$$1\,]\,0\,0\,1\,0$$

+ve

overflow occured

→ overflow occurs when the two numbers are having same sign.