Unit 1 Number System

(A) Number System :→ Number systems are systems in mathematics that are used to express numbers in various form.

The four most common number system are :-

(I) **Decimal number system** (Base-10)

(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

Example → $(987.25)_{10}$

(II) **Binary number system** (Base-2)

(0, 1)

Example :→ $(10101.011)_2$

(III) **Octal number system** (Base-8)

The octal number system uses digits

(0, 1, 2, 3, 4, 5, 6, 7)

Example – $(769.2)_8$

(IV) **Hexadecimal number system** (Base-16)

$$\begin{array}{cccccccccccccccc} 0, & 1, & 2, & 3, & 4, & 5, & 6, & 7, & 8, & 9, & A, & B, & C, & D, & E, & F \\ & & & & & & & & & & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ & & & & & & & & & & 10 & 11 & 12 & 13 & 14 & 15 \end{array}$$

Example → $(7F2.C4)_{16}$

(1) Convert Binary number $(11010.11)_2$ to Decimal number.

Sol: → 11010.11

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

$= 16 + 8 + 0 + 2 + 0 + 0.5 + 0.25$

$= (26.75)_{10}$

(2) Convert octal number $(630.4)_8$ to decimal number.

Sol: → $630.4 = 6\times8^2 + 3\times8^1 + 0\times8^0 + 4\times8^{-1}$

$= (408.5)_{10}$

(3) Convert Hexadecimal number 63.FD7 into decimal number.

Sol: → 63.FD7

$= 6\times16^1 + 3\times16^0 + 15\times16^{-1} + 13\times16^{-2} + 7\times16^{-3}$

$= (99.99)_{10}$

...ert Decimal number $(41)_{10}$ into Binary number.

Sol: →

2	41	
2	20	1
2	10	0
2	5	0
2	2	1
2	1	0
	0	1

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

(5) Convert Decimal number $(153)_{10}$ into Octal number.

Sol: →

8	153	
8	19	1
8	2	3
	0	2

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

(6) Convert decimal number $(0.6875)_{10}$ into Binary number.

Sol:→

$$0.6875 \times 2 = 1 + 0.3750$$

$$0.3750 \times 2 = 1 + 0.7500$$

$$0.75 \times 2 = 1 + 0.50$$

$$0.50 \times 2 = 1.00$$

Ans → $(0.6875)_{10} = (0.1011)_2$

(7) Convert decimal number $(0.513)_{10}$ into octal number.

Sol:→

$$0.513 \times 8 = 4 + 0.104$$

$$0.104 \times 8 = 0 + 0.832$$

$$0.832 \times 8 = 6 + 0.656$$

$$0.656 \times 8 = 5 + 0.248$$

$(0.513)_{10} = (0.4065)_8$

Convert decimal number $(789.25)_{10}$ into octal number.

Sol: →

8	789	
8	98	5
8	12	2
8	1	4
	0	1

$0.25 \times 8 = 2.00$

$0.00 \times 8 = 0.00$

$$(789.25)_{10} = (1425.20)_8$$

(9) Convert octal number $(372.67)_8$ into Hexadecimal number

Sol: →

372 . 6 7

011 111 010 110 111

011111010 . 11011100

FA . DC

$(372.67)_8$

$= (FA.DC)_{16}$

(10) Convert Hexadecimal number $(128.0D)$ into octal number.

Sol: → $(128.0D)_{16} = (450.032)_8$

128 . 0 D

0001 0010 1000 0000 1101

000100101000.00001101 0

450.032

(B) Complements

Complements → There are two Complements forms : 1's Complement form and 2's Complement form.

Most digital Computers do Subtraction by the 2's Complement method but some do it by the 1's Complement method. The advantage of performing Subtraction by the Complement method is reduction in the hardware. Instead of having separate digital Circuits for addition and Subtraction, only adding Circuits is needed. That is Subtraction is also performe by adders only.

(i) 1's Complement's of 1011000 is
= 0100111

(ii) The 2's Complement's of 1101100 is

Sol: → 1's Complement's of 1101100 is
= 0010011

2's Complement's of
$$\begin{array}{r} 0010011 \\ +\ 1 \\ \hline 0010100 \end{array}$$

(iii) The 9's Complement's of 546700 is

Sol: →
$$\begin{array}{r} 999999 \\ 546700 \\ \hline 453299 \end{array}$$

(IV) The 10's Complements of 012398

Sol:→

9's of 012398 =

$$\begin{array}{r} 999999 \\ 012398 \\ \hline 987601 \\ \hline \end{array}$$

10's of 012398 =

$$\begin{array}{r} 987601 \\ +\ 1 \\ \hline 987602 \end{array}$$

10's and 2's are Called r's Complement.

1's and 9's are Called (r-1)'s Complement.

(C) Binary addition and Substraction →

(i) If X = 101101 , Y = 100111

Determine X+Y and X−Y.

Sol:→

$$\begin{array}{rrl} X = & 101101 & (45) \\ Y = & 100111 & (39) \\ \hline X+Y = & 1010100 & (84) \end{array}$$

Rules for addition

(i) 1 + 1 = 0 , with Carry 1

(ii) 0 + 0 = 0 .

(iii) 1+1+1 = 1 with Carry 1

$X = 101101$ (45)

$Y = 100111$ (39)

$X-Y = 000110$ (6)

<u>Rules for binary Subtraction</u> →

$0-0=0$

$1-1=0$

$1-0=1$

(iii) If $X = 11001$, $Y = 10011$ then Determine $X+Y$ and $X-Y$.

Sol: →

$X = 11001$ (25)

$Y = 10011$ (19)

$X+Y$ 101100 (44)

$X = 11001$ (25)

$Y = 10011$ (19)

$X-Y = 10110$ (6)

(iii) Given the two number $X = 1010100$ and $Y = 1000011$, Perform the subtraction
(a) $X-Y$ (b) $Y-X$ using 2's Complements.

(a) $X = 1010100$

2's Complement of $y = +\ 0111101$

Sum $\overline{\text{(1)}0010001}$

Discard Carry $= 0010001$ (Ans)

$X-Y = 0010001$

(b) $Y-X$

$Y = 1000011$

2's of $X = +\ 0101100$

~~XX~~ Sum $= \overline{1101111}$

There is no Carry

$Y-X = -(2\text{'s of } 1101111)$

$= -(0010001)$

(IV) Given that two binary number $X = 1010100$ and $Y = 1000011$ Perform the subtraction (a) $X-Y$ (b) $Y-X$ using 1's Complements.

Sol: → (a) $(X-Y)$ →

$$X = 1010100$$

$$1\text{'s of } Y = 0111100$$

Sum $\quad$ 1 0010000

Add Carry

$$\begin{array}{r} 0010000 \\ +1 \\ \hline 0010001 \end{array}$$

$X - Y = 0010001$ (Ans)

(b) $Y - X$ →

~~X = 1010100~~

~~1's of Y = 0111100~~

~~010000~~

$$Y = 1000011$$

$$1\text{'s of } X = 0101011$$

Sum $\quad 1101110$

There is no Carry

$$Y - X = -(1\text{'s of } 1101110)$$

$$= -(0010001)$$

Note → This process is applicable in 9's also.

(V) Using 10's complement, Subtract

72532 − 3250 . Determine (X−Y)

Sol:→ $X = 72532$, $Y = 3250$

$X = 72532$

10's of $Y = 96750$

Sum (1)69282

Discard Carry

$X - Y = 69282$ (Ans)

(VI) Using 10's Complement, Subtract

3250 − 72532 Determine (Y−X)

Sol:→ $Y = 3250$, $X = 72532$

$Y = 03250$

10's of of $X = 27468$

Sum 30718

There is no Carry

Ans − (10's of 30718)

= −(69282)

(c) Signed binary Number → Positive integers (including zero) can be represented as unsigned numbers. However, to represent negative integers, we need a notation for negative values.

Arithmetic Addition and Substraction Using 2's Complement find the following

(i)

+6	00000110	
+13	00001101	(Sum)
+19	00010011	(Sum)

(ii)

−6 (2's of 6)		11111010
+13	~~Sum~~	00001101
+7	Sum	100000111

Ans → 00000111

Discard Carry

(iii)

+6	~~00000110~~		00000110
−13	(2's of 13)		11110011
−7		(Sum)	11111001

(iv) -6 (2's of 6) 11111010

-13 11110011

-19 (Sum) 1 11101101

Discard Carry

Ans → 11101101

In each of the four cases the operation performed is addition with the sign bit included. Any carry out of the sign bit position is discarded.

(V) -4 (11111100)

$+11$ (00001011)

$+7$ (00000111)

Binary Multiplication →

Rules for Binary Multiplication

=

$0 \times 0 = 0$

$1 \times 1 = 1$

$1 \times 0 = 0$

$0 \times 1 = 0$

(I) Multiply $(1101)_2$ by 110

Sol: →

```
     1101
   + 110
   ------
     0000
    1101
   1101
   ------
  1001110   (Ans →
```

(II) Multiply 1011·101 by 101·01

Sol: →

```
        1011 · 101
         101 · 01
     ---------------
         1011 · 101
       0 0000   00
     1 0 1 1 1 · 0  1
   0 0 0 0 0 · 0 0
  1 0 1 1 1 · 0 1
 ---------------------
  ~~+ + + 0 0 0 0 0 0 0 0 0 1~~
  1 1 1 1 0 1 · 0 0 0 0 1
 ---------------------
```

Binary Division →

(1) Divide $(110101.11)_2$ by $(101)_2$

```
101 ) 110101.11 ( 1010.11
      101
      -----
      00110
        101
        -----
        00111
          101
          -----
            101
            101
            -----
             ×
```

(D) <u>Binary Code</u> → Digital data is represented, stored and transmitted as groups of binary digits also known as binary code.

~~(1) Binary Coded Decimal Code (BCD Code)~~

Since the Computer Can accept only binary Values, we must represent the decimal digits by means of a code that Contain 1's and 0's.

Binary Codes

- Weighted Code
  - (i) Binary (8421 code)
  - (ii) BCD
- Non weighted Code
  - → Excess 3
  - → Gray

<u>Weighted Codes</u> → In weighted Codes, each digit is assigned a specific weight according to its position.

<u>Non weighted Code</u> → In non-weighted Codes are not appositionally weighted.

BCD Code (→ Binary Coded decimal Code)

A decimal number in BCD is the same as its equivalent binary number only when the number is between 0 and 9. A BCD number greater than 10 looks different from its equivalent binary number, even though both contain 1's and 0's.

Example → Consider decimal 185 and its corresponding value in BCD and binary.

$(185)_{10}$ → BCD — $(0001\ 1000\ 0101)_{BCD}$

Binary → $(10111001)_2$

Other Binary Codes →

Decimal digit	BCD 8421	Excess 3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Excess - 3 is an unweighted code in which each coded combination is obtained from the corresponding binary value plus 3.

Gray Code (Cyclic Code) → The advantage of gray code over binary code numbers is that only one bit in the code group changes when going from one number to the next. For example, in going from 7 to 8, the Gray Code changes from 0100 to 1100. Only first bit from left changes from 0 to 1, the other bit remain same. When comparing this with binary numbers, the change from 7 to 8 will be from 0111 to 1000, which causes all four bits to change values.

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

The gray code used in applications where ~~binary~~ the normal sequence of binary numbers may produce an error during transition from one number to the next. The gray code eliminate this problem since only one bit changes in value during any transition between two numbers.

(E) Floating Point Representation :→

The floating - point representation of a number has two parts. The first ~~point~~ part represents a signed, fixed point number called the mantissa. The second part designates the position of the decimal (or binary) point and is called the exponent. The fixed point mantissa may be a fraction or an integer. For example, the decimal number +6132.789 is represented in floating point with a fraction and an exponent as follows :-

Fraction	Exponent
+0.6132789	+04

The value of the exponent indicates that the actual position of the decimal point is four positions to the right of the indicated decimal point in the fraction. This representation is equivalent to the scientific notation $+0.6132789 \times 10^{+4}$

i.e $m \times r^{e}$

$m \rightarrow$ mantissa , $r \rightarrow$ radix

$e \rightarrow$ exponent

Binary number 1001.11

$$m \times 2^{e} = +(0.100111)_2 \times 2^{+4}$$

A floating point number is said to be normalized if most significant digit of the mantissa is nonzero. For example, the decimal number 350 is normalized but 0035 is not.

The 8-bit binary number 00011010 is not normalized because of three leading 0's.

Airthmatic operations with floating point numbers are more complicated that airthmatic operations with fixed point numbers and their execution takes longer and requires more complex hardware.

(F) Floating Point Arithmetic operations :→

Consider the sum of folling ~~floating~~ floating point numbers

$$.5372400 \times 10^{2}$$

$$+ .1580000 \times 10^{-1}$$

It is necessary that two exponents be equal before the ~~main~~ mantissas can be added. We can shift either the first number three positions to the left, or shift the second number three positions to the right. When the mantissas are stored in registers, shifting to the left causes a loss of most significant digits and shifting to the right causes a loss of least significant digits. The second method is preferable because it only reduces the accuracy, while the first method cause an error.

After this is done, the mantissas can be added :-

$$
\begin{array}{r}
.5372400 \times 10^2 \\
.0001580 \times 10^2 \\
\hline
.5373980 \times 10^2
\end{array}
$$

When two normalized mantissas are added, the sum may contain an overflow digit. An overflow can be corrected easily by shifting the sum once to the right and incrementing the exponent. When two number are subtracted

$$
\begin{array}{r}
0.56780 \times 10^5 \\
- \quad .56430 \times 10^5 \\
\hline
.00350 \times 10^5
\end{array}
$$

A floating point number that has a 0 in the most significant position of the mantissa is said to have an underflow. To normalize a number that contains an underflow, it is necessary to shift the mantissa to left and decrement the exponent untill a nonzero digit appears in the first position.

In the example above, it is necessary to shift left twice to obtain $.35000 \times 10^3$. In most computers, a normalization procedure is performed after each operation to ensure that all results are in a normalized form.

Floating point multiplication and ~~divisio~~ division do not require an alignment of the mantissas.

<u>Addition and ~~Subs~~ Subtraction</u> → The algorithm can be divided into four consecutive parts:-

(i) Check for zeros

(ii) Align the mantissas

(III) Add or subtract the mantissas

(IV) Normalize the result

IEEE Standard for Floating Point →

(I) Single Precision format → 32 bit

31	30 ... 23	22 ... 0
sign	exponent	Mantissa
1 bit	8 bit	23

{ 0 - Positive number
1 - negative number

(II) Double Precision format → 64 (bit)

63	62 ... 52	51 ... 0
sign	Exponent	mantissa
1 bit	11 bit	52 bit

Example 1 → Represent $(1259.125)_{10}$ in ~~Signal~~ Single and double precision format.

Sol:→
Step 1 → $(1259.125)_{10} = (10011100011.001)_2$

Step 2 → Normalig the member

Single Precision : $(1.N)\,2^{E-127}$

double Precision : $(1.N)\,2^{E-1027}$

$(10011100011.001)_2 = \cancel{1.0011100011 \times 2^{10}}$
$1.0011100011001 \times 2^{10}$

Step 3 → Single Precision format

$$[1.N]\,2^{E-127} = 1.0011100011001 \times 2^{10}$$

$$E - 127 = 10$$
$$E = 137$$

$$E = 137 = 10001001$$

31	30	22 … 0
0	10001001	0011100011001 - - -
1 bit	8 bit	23 bit

Step 4 → Double Precision format

$$[1.N]\,2^{E-1023} = 1.0011100011001 \times 2^{10}$$

$$E - 1023 = 10$$
$$E = 1033 \Rightarrow (10000001001)_2$$

63	62	51 … 0
0	10000001001	0011100011001
1 bit	11 bit	52 bit

BCD Addition → when binary sum is equal to or less than 1001 (without carry), the corresponding BCD is correct. However, when the binary sum is greater than or equal to 1010, the result is an invalid BCD digit. The addition of $6 = (0110)_2$ to the binary sum converts it to the correct digit and also produce a carry as required.

Consider the following BCD addition

(i)
$$\begin{array}{rr} 4 & 0100 \\ +5 & 0101 \\ \hline 9 & 1001 \end{array}$$

(ii)
$$\begin{array}{rr} 4 & 0100 \\ +8 & 1000 \\ \hline 12 & 1100 \\ & 0110 \\ \hline & 10010 \end{array}$$

(iii)
$$\begin{array}{rr} 8 & 1000 \\ +9 & 1001 \\ \hline 17 & 10001 \\ & 0110 \\ \hline & 10111 \end{array}$$

In first example sum is equal to 9 and is correct BCD sum. In second example, binary sum produces an invalid BCD digit (1100). The addition of 0110 produces the correct BCD sum 0010 (number 2) and a carry. In third example, Adding 0110, we obtain BCD sum 0111 (number 7) and a carry.

(1) Add 184 + 576 in BCD.

	1	1		
	0001	1000	0100	184
	0101	0111	0110	+ 576
Binary Sum	0111	(1)0000	1010	
		0110	0110	
BCD Sum	0111	0110	(1)0000	760

(2) Add 679.6 + 536.8

679.6
+536.8
1296.4 Ans → (0001 0100 0000 0110 0100)

0110	0111	1001	. 0110
0101	0011	0110	. 1000
1011	1010	1111	1110 (All are illegal Code
0110	0110	0110	0110
1	1	1	
1110	0001	0110	0100
0110			
1, 0100			

(3) Subtract 38 − 15

38	0011	1000
− 15	0001	0101
23	0010	0011

(4) 206.7
− 147.8
58.9

	0010	0000	0110	. 0111
−	0001	0100	0111	. 1000
(Borrows are Present)	0000	1011	1110	. 1111
(Subtract 0110) −		0110	0110	0110
(Correct difference)		0101	1000	1001