

①

Unit 1 Number System

① Number System \rightarrow 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 $\rightarrow (987.25)_{10}$

(II) Binary number system (Base-2)

(0, 1)

Example $\rightarrow (10101.011)_2$

(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)

(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$
10 11 12 13 14 15

Example $\rightarrow (7F2.C4)_{16}$

(2)

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

$$\text{Sol:} \rightarrow 11010.11$$

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

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

$$= (26.75)_{10}$$

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

$$\text{Sol:} \rightarrow 630.4 = 6 \times 8^2 + 3 \times 8^1 + 0 \times 8^0 + 4 \times 8^{-1}$$

$$= (408.5)_{10}$$

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

$$\text{Sol:} \rightarrow 63.FD7$$

$$= 6 \times 16^1 + 3 \times 16^0 + 15 \times 16^{-1} + 13 \times 16^{-2} + 7 \times 16^{-3}$$

$$= (99.99)_{10}$$

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

Sol: \rightarrow

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: \rightarrow

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: \rightarrow

$0.6875 \times 2 =$	1	+	0.3750
$0.3750 \times 2 =$	1	+	0.7500
$0.75 \times 2 =$	1	+	0.50
$0.50 \times 2 =$	1	.	00

Ans $\rightarrow (0.6875)_{10} = (0.1011)_2$

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

Sol: \rightarrow

$0.513 \times 8 =$	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.

(5)

Sol: \rightarrow

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



$$\begin{aligned} 0.25 \times 8 &= 2.00 \\ 0.00 \times 8 &= 0.00 \end{aligned}$$

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

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

Sol: \rightarrow

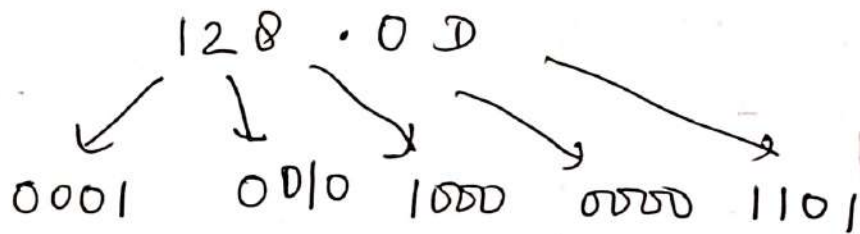
$$\begin{array}{ccccccc} & 3 & 7 & 2 & . & 6 & 7 \\ & \swarrow & \downarrow & \searrow & & \swarrow & \searrow \\ 0 & 1 & 1 & & & 1 & 1 & 0 & 1 & 0 & . & 1 & 1 & 0 & & 1 & 1 & 1 \\ & \underbrace{} & & \underbrace{} & & & & & & & & & & & & & & & \\ & \text{FA} & . & \text{DC} & & & & & & & & & & & & & & \end{array}$$

$$\begin{aligned} (372.67)_8 \\ = (\text{FA.DC})_{16} \end{aligned}$$

⑥

(10) Convert Hexadecimal number $(128.0D)_{16}$ into octal number.

$$\text{Sol: } \rightarrow (128.0D)_{16} = (450.032)_8$$



0001 0010 1000 0000 1101

450.032

(B) Complements (V)
Complements → These 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 performed 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: → $\begin{array}{r} 999999 \\ 546700 \\ \hline 453299 \end{array}$

(14) The 10's Complements of 012398 ⁽⁸⁾

Sol: \rightarrow

$$\begin{array}{r} 9's \text{ of } 012398 = 999999 \\ \phantom{9's \text{ of } 012398} 012398 \\ \hline 987601 \\ \hline \end{array}$$

$$\begin{array}{r} 10's \text{ of } 012398 = 987601 \\ \phantom{10's \text{ of } 012398} + 1 \\ \hline 987602 \end{array}$$

10's and 9's are called 9's Complement.

1's and 9's are called (9-1)'s Complement.

(C) Binary addition and Subtraction \rightarrow

(i) If $X = 101101$, $Y = 100111$

Determine $X+Y$ and $X-Y$.

Sol: \rightarrow

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

Rules for addition

- (i) $1 + 1 = 0$, with Carry 1
- (ii) $0 + 0 = 0$
- (iii) $1 + 1 + 1 = 1$ with Carry 1

$$X = 101101 \quad (45)^{(9)}$$

$$Y = 100111 \quad (39)$$

$$\begin{array}{r} X - Y = 000110 \quad (6) \end{array}$$

Rules for binary

Subtraction →

$$0 - 0 = 0$$

$$1 - 1 = 0$$

$$1 - 0 = 1$$

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

Sol: → $X = 11001 \quad (25)$

$$\begin{array}{r} Y = 10011 \quad (19) \\ X + Y = \hline 101100 \quad (44) \end{array}$$

$$X = 11001 \quad (25)$$

$$Y = 10011 \quad (19)$$

$$\begin{array}{r} X - Y = \hline 10110 \quad (6) \end{array}$$

(iii) Given the two numbers $X = 10101000$ and $Y = 1000011$, Perform the subtraction (a) $X - Y$ (b) $Y - X$ using 2's Complements.

(a) $X = 10101000$

2's Complement of $Y = +0111101$

Sum $\begin{array}{r} 10101000 \\ +0111101 \\ \hline \end{array}$

Discard Carry $= 0010001$ (Ans)

$X - Y = 0010001$

(b) $Y - X$

$Y = 1000011$

2's of $X = +0101100$

~~Sum~~ Sum $= \begin{array}{r} 1000011 \\ +0101100 \\ \hline 1101111 \end{array}$

There is no carry

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

$= -(0010001)$

(11)

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

Sol: \rightarrow (a) $(X - Y) \rightarrow$

$$X = 1010100$$

$$\begin{array}{r} \text{1's of } Y \\ \hline = 0111100 \\ \hline \text{Sum} \quad 10010000 \end{array}$$

Add Carry

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

$$X - Y = 0010001 \quad (\text{Ans})$$

(b) $Y - X \rightarrow$

$$\cancel{X = 1010100}$$

$$\begin{array}{r} \cancel{\text{1's of } Y = 0111100} \\ \hline \cancel{010000} \end{array}$$

$$Y = 1000011$$

$$\begin{array}{r} \text{1's of } X = 0101011 \\ \hline \text{Sum} \quad 1101110 \end{array}$$

There is no Carry

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

$$= - (0010001)$$

Note \rightarrow This process is applicable in 9's also.

(12)
(V) Using 10's Complement, Subtract
~~72532~~ $72532 - 3250$ • (X-Y)
 Determine

Sol: $\rightarrow X = 72532, Y = 3250$

10's of
 $X = 72532$
 $Y = 96750$
 Sum ①69282

Discard Carry

$X - Y = 69282$ (Ans)

(VI) Using 10's Complement, Subtract
 $3250 - 72532$ • (Y-X)
 Determine

Sol: $\rightarrow Y = 3250, X = 72532$

~~3250~~ $= 03250$
 10's of ~~3250~~ $= 27468$
30718

Sum

There is no Carry

Ans $- (10's \text{ of } 30718)$

$= -(69282)$

(13)

(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 Subtraction

Using 2's Complement find the following

(i)	+ 6	00000110	
	+ 13	00001101	(Sum)
	<hr/>		
	+ 19	00010011	(Sum)

(ii)	- 6 (2's of 6)	11111010
	+ 13	Sum 00001101
	<hr/>	
	+ 7	Sum 10000011
		Ans → 00000111
		Discard Carry

(iii)	+ 6	00000110	00000110
	- 13	(2's of 13)	11110011
	<hr/>		
	- 7	(Sum)	<hr/> 11110011

(14)

$$(W) \quad -6 \quad (2's \text{ of } 6) \quad 11111010 \\ -13 \quad 11110011$$

$$\begin{array}{r} -19 \\ \hline \end{array} \quad (Sum) \quad \textcircled{1}1110101$$

Discard Carry

$$Ans \rightarrow 11101101$$

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.

$$(V) \quad -4 \quad (11111100) \\ +11 \quad (00001011) \\ \hline +7 \quad (00000111)$$

(15)

Binary Multiplication →

Rules for Binary Multiplication

$$= 0 \times 0 = 0$$

$$1 \times 1 = 1$$

$$1 \times 0 = 0$$

$$0 \times 1 = 0$$

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

Sol: →

$$\begin{array}{r}
 1101 \\
 + 110 \\
 \hline
 0000 \\
 1101 \\
 1101 \\
 \hline
 1001110 \quad (\text{Ans} \rightarrow)
 \end{array}$$

(11) Multiply 1011.101 by 101.01

Sol: →

$$\begin{array}{r}
 1011.101 \\
 101.01 \\
 \hline
 1011.101 \\
 00000.00 \\
 1011.101 \\
 00000.00 \\
 1011.101 \\
 \hline
 \cancel{11100000.001} \\
 111.1010001 \\
 \hline
 \end{array}$$

Binary Division → (16)

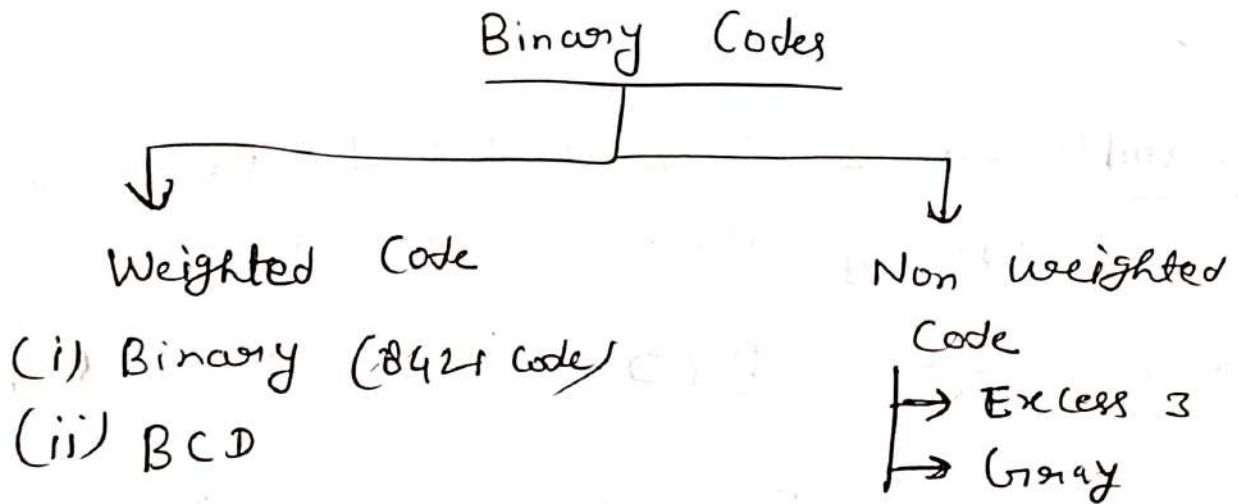
(1) Divide $(110101.11)_2$ by $(101)_2$

$$\begin{array}{r} 101 \overline{) 110101.11} \\ \underline{101} \\ 00110 \\ \underline{101} \\ 00111 \\ \underline{101} \\ 101 \\ \underline{101} \\ 0 \end{array} \quad (1010.11)$$

(D) Binary Code \rightarrow 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 contains 1's and 0's.



Weighted Codes \rightarrow In weighted codes, each digit is assigned a specific weight according to its position.

Non weighted Code \rightarrow In non-weighted codes, are not additionally weighted.

(18)

BCD Code \rightarrow 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 \rightarrow Consider decimal 185 and its corresponding value in BCD and binary.

$(185)_{10} \rightarrow \text{BCD} = (0001\ 1000\ 10101)$

Binary $\rightarrow (10111001)_2$ BCD

(19)
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 bits 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~~ 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⁽²¹⁾ Representation \rightarrow

The floating-point representation of a number has two parts. The first 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:-

<u>Fraction</u>	<u>Exponent</u>
$+0.6132789$	$+04$

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 $m \times 10^e$

$m \rightarrow$ mantissa, $10 \rightarrow$ radix

$e \rightarrow$ exponent

(22)

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 non-zero.

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.

Arithmetic operations with floating point numbers are more complicated than arithmetic operations with fixed point numbers and their execution takes longer and requires more complex hardware.

(F) Floating Point Arithmetic Operations: →

Consider the sum of following floating point numbers

$$\begin{aligned} & \cdot 5372400 \times 10^2 \\ & + \cdot 1580000 \times 10^7 \end{aligned}$$

It is necessary that two exponents be equal before the mantissas can

be added. (23) 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 numbers are subtracted

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

(24)

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 until a non zero digit appears in the first position.

In the example above, it is necessary to shift left twice to obtain 0.35000×10^3 . 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 division do not require an alignment of the mantissas.

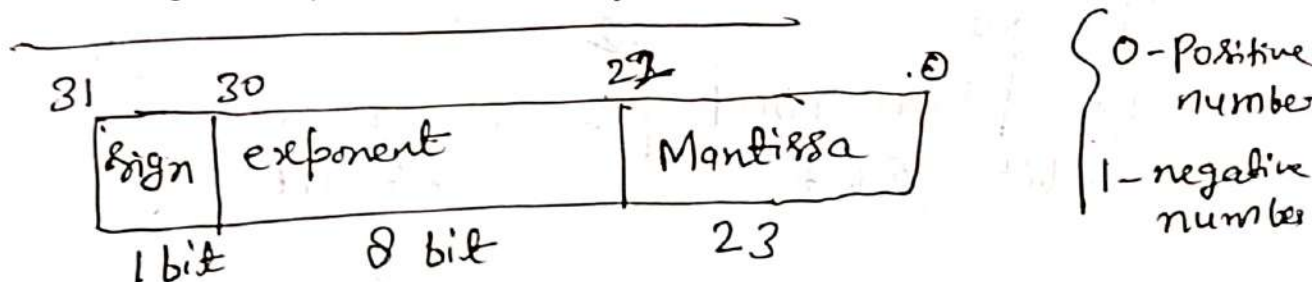
Addition and Subtraction \rightarrow The algorithm can be divided into four consecutive parts:-

- (i) Check for zeros
- (ii) Align the mantissas

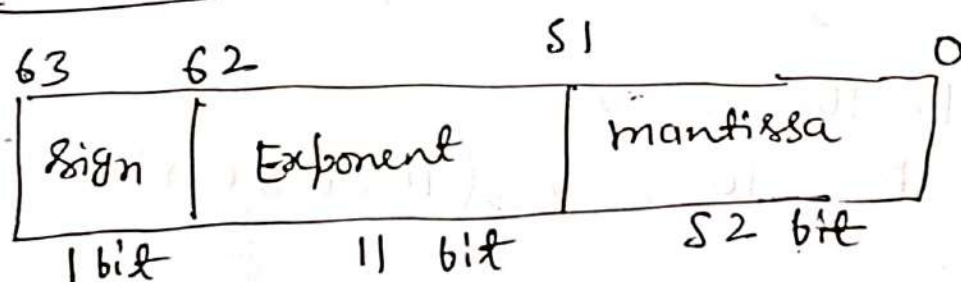
- (iii) Add or Subtract the mantissas
- (iv) Normalize the result

IEEE Standard for Floating Point →

(i) Single Precision format → 32 bit



(ii) Double Precision format → 64 (bit)



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

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

Step 1 →

Step 2 → Normalizing the number

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

Double Precision: $(1.N) \times 2^{E-1023}$

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

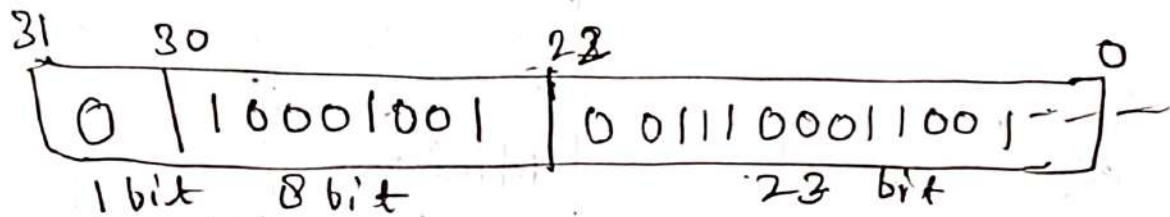
Step 3 → Single Precision format ⁽²⁶⁾

$$[1N]_2 2^{E-127} = 1.001110001001 \times 2^{10}$$

$$E-127 = 10$$

$$E = 137$$

$$E = 137 = 10001001$$

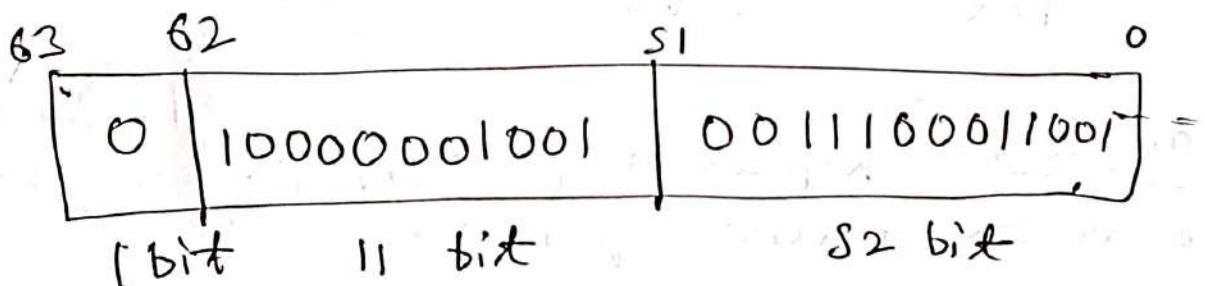


Step 4 → Double Precision format

$$[1N]_2 2^{E-1023} = 1.001110001001 \times 2^{10}$$

$$E-1023 = 10$$

$$E = 1033 \Rightarrow (10000001001)_2$$



BCD Addition \rightarrow 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 produces a carry as required.

Consider the following BCD addition

(i)	4	0100	(ii)	4	0100	(iii)	8	1000
	+ 5	0101		+ 8	1000		+ 9	1001
	<hr/>			<hr/>			<hr/>	
	9	1001		12	1100		17	10001
					0110			0110
					<hr/>			<hr/>
					10010			10111

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)

(58)

Add 184 + 576 in BCD.

	1	1		
	0001	1000	0100	184
	0101	0111	0110	+ 576
Binary Sum	0110	①0000	1010	
		0110	0110	
BCD Sum	0110	0110	①0000	760

(2)

Add 679.6

+ 536.8

Ans →

1296.4 (0001 01000001 0110 0100)

0110 0111 1001 . 0110

0101 0011 0110 . 1000

1011 1010 1111 1110 (All are
0110 0110 0110 0110 illegal Code)

1 1 1 1

1110 0001 0110 0100

0110

1, 6100

(3) Subtract $38 - 15$

(29)

$$\begin{array}{r}
 38 \\
 - 15 \\
 \hline
 23
 \end{array}$$

$$\begin{array}{r}
 0011 \\
 0001 \\
 \hline
 0010
 \end{array}$$

$$\begin{array}{r}
 1000 \\
 0101 \\
 \hline
 0011
 \end{array}$$

(4) 206.7

$- 147.8$

58.9

$$\begin{array}{r}
 0010 \quad 0000 \quad 0110 \quad . 0111
 \end{array}$$

$$\begin{array}{r}
 - 0001 \quad 0100 \quad 0111 \quad . 1000
 \end{array}$$

(Borrows are Present) $0000 \quad 1011 \quad 1110 \quad . 1111$

(Subtract 0110) $- 0110 \quad 0110 \quad 0110$

(Correct difference) $0101 \quad 1000 \quad 1001$