Toppersnotes
Unleash the topper in you

ENGINEERS ACADEMY

RPSC - A.En.

Assistant Engineering

ELECTRICAL

Rajasthan Public Service Commission (RPSC)

Volume - 11

Digital Electronics

SIERRA INNOVATIONS

NUMBER SYSTEMS & BINARY ARITHMETIC

THEORY

1.1 INTRODUCTION

A decimal number system is said to be base 10 because it uses 10 digits (0 to 9).

A binary number system has base 2 because it uses 2 digits (0 and 1).

Any number to the base 'r' uses r digits from 0 to (r – 1). If the value of base r ≤ 10, r digits are taken from decimal number system. If the value of base r > 10 the letters of alphabets are used in addition to 10 decimal digits (0 to 9).

Ex.: Hexadecimal number system have base 16 and it requires 16 digits. Out of sixteen digits, 10 digits are taken from decimal number system while remaining 6 digits (10 to 15) are represented by letters (A to F) respectively.

1.2 TYPE OF NUMBER SYSTEMS

- Binary number system (Base-2)
- Octal number system (Base-8)
- Decimal number system (Base-10)
- Hexadecimal number system (Base-16)

1.2.1 Binary Number System

This number system uses 0 and 1 as digits. Its radix or base is 2. Binary number system consists of only two digits i.e. 0 and 1. In binary number system, each digit is called a bit, the group of four bits is called a nibble and group of 8 bits is called a byte. The highest decimal number represented by n digit number is $2^n - 1$. For example, with 4 bit binary number, highest decimal number represented is

$$2^4 - 1 = 15$$

(i) Decimal to Binary Conversion : A decimal number can be converted into binary number dividing the decimal number by two progressively until quotient zero is obtained. Binary number is obtained by taking remainder of each division in reverse order with first remainder as LSB and last remainder as MSB.

If decimal number is a fraction, its binary equivalent is obtained by multiplying the number progressively by 2 with carry generated as binary digit. First carry is taken as MSB and last carry as LSB.

Example 1 : Convert the $(117)_{10}$ into binary.

Solution :

2	117 Quotient	Remainder	
2	58	1	LSB
2	29	0	↑
2	14	1	
2	7	0	
2	3	1	
2	1	1	
	0	1	MSB

$$(117)_{10} = (1110101)_2$$

Example 2 : Convert the $(0.61)_{10}$ into binary.

Solution :

Conversion of 0.61 in to Binary

$0.61 \times 2 = 1.22$ (MSB) 1 ↓
$0.22 \times 2 = 0.44$ 0
$0.44 \times 2 = 0.88$ 0
$0.88 \times 2 = 1.76$ 1

only upto 4 or 5 digits

$$\Rightarrow \quad (0.61)_{10} = (0.1001)_2$$

(ii) Binary to Decimal Conversion : $(.... X_4 X_3 X_2 X_1 X_0.X_{-1} X_{-2} X_{-3}.....)$

Decimal equation : $[(.... X_3 r^3 + X_2 r^2 + X_1 r^1 + X_0 r^0).(X_{-1} r^{-1} + X_{-2} r^{-2} + X_{-3} r^{-3} + ...)]$

For binary number system, $r = 2$

Ex.: $(101101.001)_2 = (X)_{10}$

$(1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0).(0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3})$

$$[1 + 0 + 4 + 8 + 0 + 32].\left[\frac{1}{8}\right] = (45.125)_{10}$$

1.2.2 Octal Number System

This number system uses 0, 1, 2, 3, 4, 5, 6 and 7 as digits. Its radix or base is 8.

(i) Conversion from Decimal to Octal : Absolute (Integral value) : An absolute decimal number is converted into octal number by dividing the number by 8 progressively as in binary system.

Fractional number : A fractional number be converted to octal number by multiplying the number by 8 progressively as in case of binary system.

(ii) Conversion from Octal to Decimal : A number with radix or base 'r' can be converted to its decimal equivalent by multiplying each digit by its respective weight and adding the products.

Ex.: $(467.32)_8 = (X)_{10}$

For octal number system, $r = 8$

$$X = (7 \times 8^0 + 6 \times 8^1 + 4 \times 8^2).(3 \times 8^{-1} + 2 \times 8^{-2})$$

$$= (7 + 48 + 256).\left(\frac{3}{8} + \frac{2}{64}\right)$$

$$= 311.(0.375 + 0.03125)$$

$$= (311.406)_{10}$$

1.2.3 Decimal Number System

Decimal number system consists of 10 digits

i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

The base or radix of this number system is 10.

Digits = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and base = 10.

1.2.4 Hexadecimal Number System

Hexadecimal number system consists of 16 digits

i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F.

The base or radix of this number system is 16.

Digits = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F and base = 16.

(i) Conversion from Decimal to Hexadecimal

(a) Integral Part : An absolute decimal number is converted to hexadecimal number by dividing the number by 16 progressively in similar manner as in binary number system.

(b) Decimal Part / Fractional Part : A fractional decimal number is converted to hexadecimal number by multiplying the number by 16 progressively in similar manner as in binary number system.

(ii) Conversion from Hexadecimal to Decimal : A hexadecimal number can be converted to its decimal equivalent by multiplying each digit by its respective weight and adding all the products.

Example 3 : $(B7AF.3E)_{16} = (X)_{10}$ find the value of X.

Solution :

$$X = (11 \times 16^3 + 7 \times 16^2 + 10 \times 16^1 + 15 \times 16^0).\left(\frac{3}{16}+\frac{14}{16^2}\right)$$

$$= (11 \times 4096 + 7 \times 256 + 10 \times 16 + 15 \times 1).\left(\frac{3}{16}+\frac{14}{256}\right)$$

$$= (45056 + 1792 + 160 + 15).(0.1875 + 0.0546)$$

$$= (47023.242)_{10}$$

1.3 Conversion of Number System

1.3.1 Binary to Hexadecimal

Binary to Hexadecimal conversion is obtained by making group of four bits to left and right of decimal point. Then each group is replaced by its hexadecimal equivalent number.

1.3.2 Hexadecimal to Binary

Hexadecimal to binary conversion is obtained by replacing each digit by its binary equivalent of four bits.

Ex.: $(1A3.BA)_{16} = \left(\underbrace{0001}_{1}\underbrace{1010}_{A}\underbrace{0011}_{3}.\underbrace{1011}_{B}\underbrace{1010}_{A}\right)_2$

1.3.3 Binary to Octal Conversion

A binary number can be converted to its octal equivalent by making group of three bits to left and right of decimal point. Then each group is replaced by its octal equivalent number.

Ex.: $(1\,0\,1\,1\,1\,0\,0\,1\,0\,.\,0\,0\,1\,1)_2 = (X)_8$

5 6 2 1 4

$$X = 562.14$$

1.3.4 Octal to Binary Conversion

Octal to binary conversion is obtained by replacing each digit by its binary equivalent of three bits.

Ex.: $(462.71)_8 = (X)_2$

$$X = 100\ 110\ 010\ .\ 111\ \ 001$$

1.3.5 Hexadecimal to Octal Conversion

Hexadecimal to octal conversion is obtained by converting hexadecimal to binary and then from binary to octal number.

1.3.6 Octal to Hexadecimal Conversion

Octal to hexadecimal conversion is obtained by converting octal to binary and then from binary to hexadecimal number.

Table of Different Number System

Decimal	Binary	Octal	Hexadecimal
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	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

1.3.7 Arithmetic Operation

Binary addition, subtraction, multiplication :

(i) Binary Addition

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Ex.:
$$\begin{array}{r} 101011 \\ +101100 \\ \hline 1010111 \end{array}$$

(ii) Binary Subtraction

A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Ex.:
$$\begin{array}{r} \overset{2}{\curvearrowleft} \\ 11011 \\ -10110 \\ \hline 00101 \end{array}$$

(iii) Binary multiplication

Ex.:
$$\begin{array}{r} 1011 \\ 101 \\ \hline 1011 \\ 0000 \\ 1011 \\ \hline 110111 \end{array}$$

Octal addition, subtraction :

$0 + 0 = 0$

$0 + 3 = 3$

$1 + 4 = 5$

$1 + 7 = 10$ $\quad [7 + 1 = (8)_{10} = (10)_8]$

$7 + 3 = 12$

Ex.: Octal sum

$$\begin{array}{r} 243 \\ +322 \\ \hline 565 \end{array} \qquad \begin{array}{r} 564 \\ +323 \\ \hline 1107 \end{array}$$

Octal subtract

$$\begin{array}{r} {}^{\overset{8}{\curvearrowright}}734 \\ 362 \\ \hline 352 \end{array}$$

Hexadecimal (addition, subtraction)

$1 + 1 = 2$

$1 + 7 = 8$

$2 + 9 = B$

$A + A = 14$ $\quad [A + A = (20)_{10} = (14)_{16}$

Ex.: Hexadecimal sum

$$\begin{array}{r} 8689 \\ 3032 \\ \hline B6BB \end{array} \qquad \begin{array}{r} DAD \\ ADD \\ \hline 188A \end{array}$$

Hexadecimal subtract

$$\begin{array}{r} {}^{\overset{16}{\curvearrowright}}\,{}^{\overset{16}{\curvearrowright}} \\ 956C \\ -274D \\ \hline 6E1F \end{array} \qquad \begin{array}{r} \\ 6287 \\ -2145 \\ \hline 4142 \end{array}$$

1.4 Codes

Codes are symbolic representation of discrete information, which may be present in the form of number, letters or physical quantities. These symbols are used to communicate information to computers.

Note :

(i) Maximum number of distinct quantities using n bits = 2^n.

(ii) Each digit of decimal number system is represented by a four digit code in weighted and non weighted codes. Except Gray code.

1.4.1 Weighted Binary Codes

These are the codes in which each symbol position is assigned a weight. In weighted binary code, each bit is multiplied by its weight indicated in code to get the decimal equivalent of the code.

e.g. BCD (8421), 7421, 5421, 5211, 4221, 3321, 2421, $84\bar{2}\bar{1}$, $74\bar{2}\bar{1}$.

Example 4 : Convert $(8)_{10}$ in 7421, 5421, 5211, 4221, 3321, 2421, 8421, $74\bar{2}\bar{1}$ codes.

Solution :

(a) $(8)_{10} = 7 + 1 = (1001)_{7421}$

(b) $(8)_{10} = 5 + 2 + 1 = (1011)_{5421}$

(c) $(8)_{10} = 5 + 2 + 1 = (1101)_{5211} = (1110)_{5211}$

(d) $(8)_{10} = 4 + 2 + 2 = (1110)_{4221}$

(e) $(8)_{10} = 3 + 3 + 2 = (1110)_{3321}$

(f) $(8)_{10} = 2 + 4 + 2 = (1110)_{2421}$

(g) $(8)_{10} = 8 + 0 + 0 + 0 = (1000)_{8421}$

(h) $(8)_{10} = 7 + 4 - 2 - 1 = (1111)_{74\bar{2}\bar{1}}$

Note : Self complement : A code is called self complementing code if is complement of the coded number yield 9's complement of number itself i.e., '0' is complement of '9', 1 is complement of '8', 2 is of 7, 3 is of 6, 4 is of 5.

Example of self complement codes is 5211, 4221, 3321, 2421, $84\bar{2}\bar{1}$.

Ex.: Representation of $(8)_{10}$ in 5211 code

- $(1110)_{5211} = (8)_{10}$
- $(1101)_{5211} = (8)_{10}$

  Both are correct.
- For self complementary, '1' should be complement of code of '8'.

1.4.2 Non Weighted Codes

Non-weighted codes are codes which are not positionally weighted, excess-3 code and gray codes are example of non-weighted codes.

(i) Excess-3 : Excess-3 code is obtained by adding 0011 to each BCD code of decimal numbers.

e.g.

decimal	BCD	Excess-3
0	0000	0011
1	0001	0100
2	0010	0101
:	:	:
9	1001	1100

Note : Excess-3 code is self complementing code.

(ii) Gray code : In gray code, every new code differs from the previous code only by a single bit. That is why, it is also called Reflected code. It can be used for measurement of shaft speed.

Note : It is also called unit distance code or minimum change code.

1.4.3 Alpha Numeric Codes

(i) **ASCII** (American Standard code for Information Interchange) : It is a 7 bit code. It is most commonly used in microcomputers. This code represents a character with seven bits, which can be stored as one byte with one bit used.

(ii) **ISCII** (Indian Standard code for Information Interchange).

(iii) **EBCDIC (ebb. see-dic) :** It uses eight bits for character and ninth bit for parity.

(iv) **Hollerith Code :** It is used in punch cards used for storage of information.

(v) **Morse Code :** It is used in Telegraph.

(vi) **Tele Type Writter** (TTW).

1.4.4 Error Detecting and Correcting Codes

(i) **Parity Check Code :** This code detects single bit error which changes parity of information. Errors with more than one bit cannot be detected with this code.

(ii) **Hamming Code :** This code detects as well as corrects the error in an information. This code transmits additional parity bits in addition to information bits. These parity bits are used to detect and correct the error.

1.4.5 Error Detecting Code

(i) Even Parity Method : In this method, the value of parity is added so that total number of 1's in code group (including parity bit) is even.

$$\text{Parity} = 0 \text{ or } 1$$

(ii) Odd Parity Method : In this method, the value of parity is added so that total number of 1's in code group (including parity bit) is odd.

Ex.: $(1011) \rightarrow$ odd

$(11000) \rightarrow$ even

1.5 Code Conversions

(i) **Decimal to BCD conversion :** Decimal to BCD conversion is obtained by replacing each digit of decimal number by its binary equivalent.

(ii) **Binary to BCD conversion :** Binary to BCD code conversion is possible by converting binary to decimal and then from decimal to its BCD equivalent.

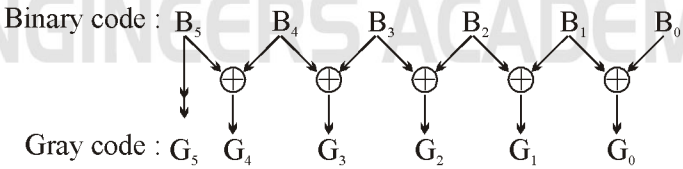
(iii) **BCD to Binary conversion :** BCD to binary conversion is obtained by converting the BCD to decimal and then decimal to its binary equivalent.

(iv) **BCD to Excess-3 conversion :** BCD to excess-3 code conversion is obtained by adding 0011 to each BCD coded digit.

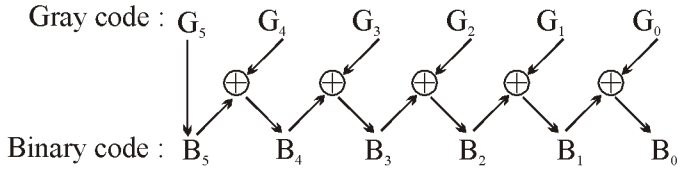
(v) **Excess-3 to BCD conversion :** Each (Excess-3 code) – 0011 = BCD

Excess-3 to BCD code conversion is obtained by substracting 0011 from excess-3 code of each digit.

(vi) **Binary to gray code conversion :** Binary to gray code conversion is obtained by using rules as shown under.



(vii) **Gray to binary code conversion :** Gray to binary code conversion is obtained by using rules as shown under.



1.6 BCD and Excess-3 Arithmetic

1.6.1 BCD

(a) BCD Addition :

Rules :

(i) If result of addition contains invalid BCD, add 0110 to that BCD.

(ii) If carry is passed from lower BCD to next BCD add 0110 to that BCD.

Example 5 : $(80)_{BCD} + (80)_{BCD} = ?$

Solution :

$$\begin{array}{r} 1000\ \ 0000 \\ 1000\ \ 0000 \\ \hline \boxed{1}\ 0000\ \ 0000 \\ \quad\ 0110 \\ \underbrace{1}\ \underbrace{0110}\ \ \underbrace{0000} \\ 1\ \ \ 6\ \ \ \ 0 \end{array}$$

$(80)_{BCD} + (80)_{BCD} = (160)_{BCD}$

Example 6 : $(88)_{BCD} + (88)_{BCD} = ?$

Solution :

$$\begin{array}{r} 1000\ \ 1000 \\ 1000\ \ 1000 \\ \quad\ \boxed{1} \\ \hline \boxed{1}\ 0001\ \ 0000 \\ \quad\ 0110\ \ 0110 \\ \hline \underbrace{1}\ \underbrace{0111}\ \ \underbrace{0110} \\ 1\ \ \ 7\ \ \ \ 6 \end{array}$$

$(88)_{BCD} + (88)_{BCD} = (176)_{BCD}$

Example 7 : $(23)_{BCD} + (88)_{BCD} = ?$

Solution :

$$\begin{array}{r} 0010\ \ 0011 \\ 1000\ \ 1000 \\ \hline \underbrace{1010}_{\text{invalid BCD}}\ \ \underbrace{1011}_{\text{invalid BCD}} \\ \text{Step–I} \quad\ \boxed{1}\ 0110 \\ \hline 1011\ \ 0001 \\ \text{Step–II} \quad 0110 \\ \hline \underbrace{1}\ \underbrace{0001}\ \ \underbrace{0001} \\ 1\ \ \ 1\ \ \ \ 1 \end{array}$$

$(23)_{BCD} + (88)_{BCD} = (111)_{BCD}$

Example 8 : $(76)_{BCD} + (86)_{BCD} = ?$

Solution :

$$\begin{array}{lll} & 0111 & 0110 \\ & 1000 & 0110 \\ \hline & \underbrace{1111}_{\text{invalid BCD}} & \underbrace{1100}_{\text{invalid BCD}} \\ \text{Step–I} & \boxed{1} & 0110 \\ \hline & 0000 & 0001 \\ \text{Step–II} & 0110 & \\ \hline & \underbrace{1}_{1} \quad \underbrace{0110}_{6} & \underbrace{0010}_{2} \end{array}$$

$$(76)_{BCD} + (86)_{BCD} = (162)_{BCD}$$

(b) BCD Subtraction : Subtraction BCD is performed either by 9′s complement or by 10′s complement. Rules are similar to binary subtraction with 1′s and 2′s complement.

1.6.2 Excess-3 Arithmetic

(a) Excess-3 addition :

Rules :

(i) Convert the given number to Excess-3 format.

(ii) Now add Excess-3 numbers.

(iii) If carry is generated by Excess-3 code of 2 digits, add $(0011)_2$ to the sum of digits.

(iv) If carry is not generated, subtract 0011 from sum of two digits.

(v) Subtracting $(0011)_2$ is equivalent to adding $(1101)_2$ or $(13)_{10}$.

Example 9 : $(38)_{10} + (44)_{10} = ?$

Solution :

$$\begin{array}{lll} & 0110 & 1011 \\ & 0111 & 0111 \\ & \boxed{0} \quad \boxed{1} & \\ \hline & 1110 & 0010 \\ & -0011 & +0011 \\ \hline \text{ignored} & \boxed{1} \ \underbrace{1011}_{(8} & \underbrace{0101}_{2)_{10}} \end{array}$$

(b) Excess-3 subtraction : Excess-3 subtraction is done with 9′s complement or 10′s complements.

1.7 Number Representation in Binary Number System

1.7.1 Sign Magnitude Representation

Left most bit represents sign.

For example $\underbrace{B_{n-1}}_{\text{Sign}} \underbrace{B_{n-2}B_{n-3}.....B_2B_1B_0}_{\text{Magnitude}}$

Note : Range = $-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$

Example 10 : Find sign Magnitude representation of $(-\ 20)_{10}$ and $(+\ 20)_{10}$.

Solution :

$$-\ 20 \Rightarrow \underset{-}{1}\ \underbrace{1\ 0\ 1\ 0\ 0}_{20}$$

$$+\ 20 \Rightarrow \underset{+}{0}\ \underbrace{1\ 0\ 1\ 0\ 0}_{20}$$

Example 11 : Find sign magnitude representation of $(-\ 0)_{10}$ and $(+\ 0)_{10}$.

Solution :

$$-\ 0 \Rightarrow 1000$$

$$+\ 0 \Rightarrow 0000$$

Note : Zero has two distinct representation in sign magnitude representation.

1.7.2 One's Complement Representation

When each bit of number is complemented, resulting number is one's complement of original number. One's complement of a number gives its negative number.

Note : Range = $-\ (2^{n-1} - 1)$ to $+\ (2^{n-1} - 1)$.

Example 12 : Find the one's complement of $(-20)_{10}$.

Solution :

$(+20)_{10} \Rightarrow (00010100)_2$ original number

$(-20)_{10} \Rightarrow (11101011)_2$ one's complement

Example 13 : Find the one's complement of $(-0)_{10}$ and $(+0)_{10}$.

Solution :

$(+0)_{10} \Rightarrow (0000)_2$

$(-0)_{10} \Rightarrow (1000)_2$

Two distinct representation of zero.

Note :

- 1's complement representation of zero has two distinct representations.
- 1's complement of a number gives original number.

1.7.3 Two's Complement Representation

2's complement of a number = 1's complement of the number + 1

Note : 2's complement has unique representation of zero.

Ex.:

$$\begin{array}{r} (0)_{10} = (0\ 0\ 0\ 0)_2 \\ 1\ 1\ 1\ 1 \\ +\ 1 \\ \hline \end{array}$$

Final carry ignored $\rightarrow \boxed{1}0000$ 2's complement

Note : Range $\rightarrow -\ 2^{n-1}$ to $(2^{n-1} - 1)$.

Example 14 : Find 2's complement representation of $(-37)_{10}$.

Solution :

$$(+37)_{10} = (00100101)_2$$

2's complement representation of $(-37)_{10}$ = 1's complement of $(00100101)_2 + 1$

$$= (11011010)_2 + 1$$

$$= (11011011)_2$$

Table of number representation

Binary	Sign magnitude	1's	2's
0000	+0	+0	+0
0001	+1	+1	+1
0010	+2	+2	+2
0011	+3	+3	+3
0100	+4	+4	+4
0101	+5	+5	+5
0110	+6	+6	+6
0111	+7	+7	+7
1000	–0	–7	–8
1001	–1	–6	–7
1010	–2	–5	–6
1011	–3	–4	–5
1100	–4	–3	–4
1101	–5	–2	–3
1110	–6	–1	–2
1111	–7	–0	–1

1.8 r's Complement & (r-1)'s Complement

(i) r's complement : r's complement of a positive number 'N' of n digits in integral part = $r^n - N$

Where, r = radix or base

n = number of digits in integral part of the number.

Ex.:

(1) 10's complement of $(52520)_{10}$ is

$10^5 - 52520 = 47480$ here n = 5 and r = 10

(2) 10's complement of $(0.3267)_{10}$ is

$(10^0 - 0.3267) = 0.6733$ here n = 0 and r = 10

(3) 2's complement of $(101100)_2$ is

$(2^6)_{10} - (101100)_2 = (1000000)_2 - (101100)_2 = 010100$

(4) 2's complement of $(0.0110)_2$ is

$2^0 - 0.0110 = 1 - 0.0110 = (0.1010)_2$

(ii) (r – 1)'s complement : (r^n)'s complement of a number N with 'n' digits in integral part and m digits in fractional part is $(r^n - r^{-m} - N)$.

Ex.: 9's complement of $(52520)_{10}$ is $(10^5 - 10^{-0} - 52520) = 47479$ [n = 5, m = 0]

Note : Complement of complement of a number restore the original number.

(iv) Subtraction using (r – 1)'s complement : Let M – N is found with (r – 1)'s complement.

(a) Add minuend M to (r – 1)'s complement of subtrahend N.

(b) Inspect for an end carry

- If an end carry is generated, add it to the sum and result is positive and in its true form.
- If an end carry is not generated, result is negative and in its (r – 1)'s complement form.

Binary subtraction using 1's complement

Case-I : (A – B) :

A → Minuend

B → Subtrahend

- 1′s complement of B is added to 'A'.
- If end around carry is generated, it is added to the result and result is a positive number and in its true form.

Example 15 : Solve $(6)_{10} - (4)_{10}$ By using 1′s complement.

Solution :

$(4)_{10} = 0100$

1′s complement of 0100 = 1011

$$(6)_{10} + (-4)_{10} = \begin{array}{r} 0110 \\ +1011 \\ \hline \boxed{1}0001 \\ \hline 0010 \end{array} \quad \text{(1 → end around carry)}$$

If end around carry is not generated, the result is negative and takes ones complement of result to find true result.

Example 16 : Solve $(4)_{10} - (6)_{10}$ By using 1′s complement ?

Solution :

$(6)_{10} = 0110$

1′s complement of 0110 = 1001

$\therefore$ $$(4)_{10} - (6)_{10} = \begin{array}{r} 0100 \\ +1001 \\ \hline 1101 \end{array}$$

Case-II : (– A – B) :

- One's complement of A and B are added.
- If carry is generated that should be added to the result and result is in its 1′s complement form.

Example 17 : Solve $(-4)_{10} - (6)_{10}$ by using 1's complement.

Solution :

Number bits required = 5

$(+4)_{10} = 00100$

1's complement of $(+4)_{10} = (-4)_{10} = 11011$

$(+6)_{10} = 00110$

1's complement of $(+6)_{10} = (-6)_{10} = 11001$

$\therefore \quad (-4)_{10} + (-6)_{10} =$

$$\begin{array}{r} 11011 \\ +11001 \\ \hline \boxed{1}\,10100 \\ \quad\quad \rightarrow 1 \\ \hline 10101 \end{array}$$

(iii) Subtraction using r's complement : Let M – N is subtraction to be found. Both are of base r:

Steps :

(a) Add the minuend M to the r's complement of the subtrahend N.

(b) Inspect for an end carry

- If an end carry occurs discard it and result is positive.
- If an end carry does not occur, result is negative and is r's complement of true result.

Binary subtraction using 2's complement

Case-I : (A – B) : Minuend is added to 2's complement of subtrahend.

- If carry is generated result is positive and in its true form, carry is discarded.
- If carry is not generated result is negative and it is 2's complement of true value.

Case-II : (– A – B) : Two's complement of both A and B are added.

- Generated carry is ignored.
- Result is always negative and in 2's complement form.

Ex.:

(–6) + (+9)	(+6) + (–9)	(–6) + (–9)
1111010	0000110	1111010
+0001001	+1110111	+1110110
(+3) $\boxed{1}$ 0000011	(–3) 1111101	(–15) $\boxed{1}$ 1101111

1.10 Floating-Point Numbers

Floating-point number notation can be used conveniently to represent both large and small fractional or mixed numbers. It makes the process of arithmetic operations on these numbers relatively much easier. It greatly increases the range of numbers from the smallest to the largest, that can be represented using a given number of digits. Floating-point numbers are in general expressed in the form

$$N = m \times b^e \qquad ...(1)$$

Where m is the fractional part called the significant or mantissa, e is the integer part called the exponent, and b is the base of the number system or numeration. Fractional part m is a p-digit number of the form (±d.dddd .... dd), with each digit d being an integer between 0 and b – 1 inclusive. If the leading digit of m is non-zero, then the number is said to be normalized.

Decimal system, $N = m \times 10^e$ ...(2)

Hexadecimal system, $N = m \times 16^e$ ...(3)

Binary system, $N = m \times 2^e$ ...(4)

Floating-point numbers consist of two parts :

Mantissa-the part of floating-point number that represents the magnitude of the number.

Exponent-the part of a floating-point number that represents the number of places that the decimal point (or binary) is to be moved.

S	Exponent (E)	Mantissa (fraction, F)
1 bit	8 bits	23 bits

Formula to be used to calculate the floating-point numbers is as follows :

$$\text{Number} = (-1)^s (1 + F) (2^{E-127})$$

For example, the decimal numbers 0.0003754 and 3754 are represented in floating-point notation as 3.754×10^{-4} and 3.754×10^3, respectively. Similarly, a hex number 257. ABF will be represented as $2.57ABF \times 16^2$.

In case of normalized binary numbers, the leading digit, which is the MSB, is always '1' and thus does not need to be stored explicitly. While expressing a given mixed binary number as a floating-point number, the radix point is shifted in such a manner so as to have the MSB immediately to the right of the radix point as a '1'. Both the mantissa and the exponent can have a positive or a negative value.

The mixed binary number $(110.1011)_2 = 0.1101011 \times 2^3 = 0.1101011e + 0011$.

Here, 0.1101011 is the mantissa and e + 0011 implies that the exponent is +3.

For example, $(0.000111)_2$ will be written as 0.111e – 0011, with 0.111 being the mantissa and e – 0011 implying an exponent of –3. Also, $(-0.00000101)_2$ may be written as $-0.101 \times 2^5 = -0.101e - 0101$, where – 0.101 is the mantissa and e^{0101} indicates an exponent of –5.

If we want to represent the mantissas using eight bits, then 0.1101011 and 0.111 would be represented as 0.11010110 and 0.11100000, respectively.

❍❍❍