Batch: A2	**Roll No.: 16010122041**
Experiment / assignment / tutorial No. 5	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

TITLE: Implementation of IEEE-754 floating point representation

AIM: To demonstrate the single and double precision formats to represent floating point numbers.

Expected OUTCOME of Experiment: (Mention CO attained here)

Books/ Journals/ Websites referred:

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, "Computer Organization", Fifth Edition, TataMcGraw-Hill.

2. William Stallings, "Computer Organization and Architecture: Designing for Performance", Eighth Edition, Pearson.

Pre Lab/ Prior Concepts:

The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE). The standard addressed many problems found in the diverse floating point implementations that made them difficult to use reliably and portably. Many hardware floating point units now use the IEEE 754 standard.

The standard defines:

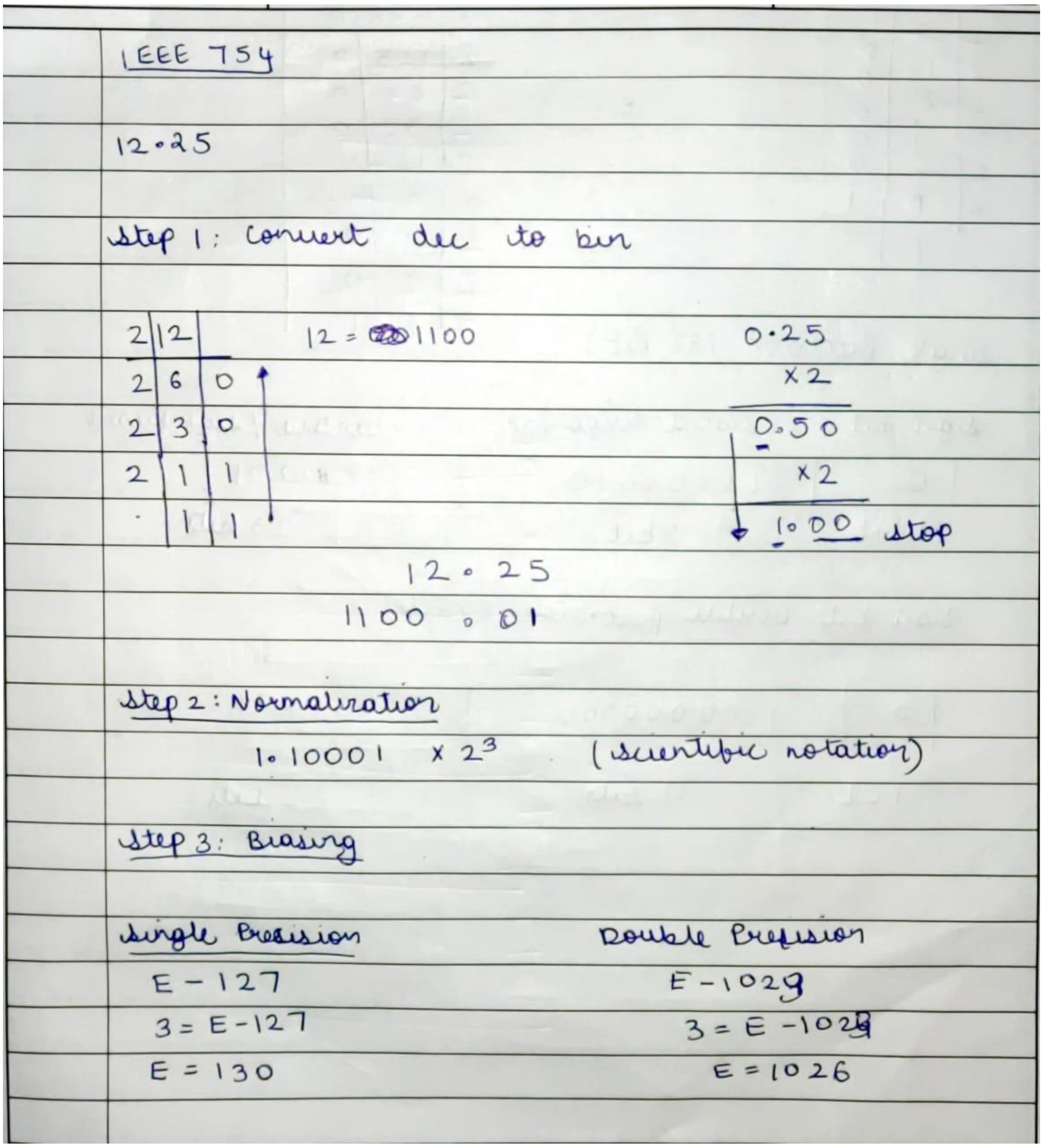
- *arithmetic formats:* sets of binary and decimal floating-point data, which consist of finite numbers (including signed zeros and subnormal numbers), infinities, and special "not a number" values (NaNs)

- *interchange formats:* encodings (bit strings) that may be used to exchange floating-point data in an efficient and compact form

- *rounding rules:* properties to be satisfied when rounding numbers during arithmetic and conversions

- *operations:* arithmetic and other operations (such as trigonometric functions) on arithmetic formats
- *exception handling:* indications of exceptional conditions (such as division by zero, overflow, *etc*

Example (Single Precision- 32 bit representation)
Example (Double Precision- 64 bit representation)



IEEE 754

12.25

Step 1: convert dec to bin

2	12	
2	6	0
2	3	0
2	1	1
.	1	1

12 = 1100

0.25
× 2
0.50
× 2
1.00 stop

12.25
1100.01

Step 2: Normalization

1.10001×2^3 (scientific notation)

Step 3: Biasing

Single Precision	Double Precision
E − 127	E − 1023
3 = E − 127	3 = E − 1023
E = 130	E = 1026

	130	
2	65	0
2	32	1
2	16	0
2	8	0
2	4	0
2	2	0
2	0	1
	1	

	1026	
2	512	0
2	256	0
2	128	0
2	64	0
2	32	0
2	16	0
2	8	0
2	4	0
2	2	0
2	1	1

Single Precision (32 bit)

Sign bit	Biased exponent	Mantisa / significant
0	10000010	10001
1 bit	8 bit	23 bits

~~sign bit~~ Double Precision (64 bit)

0	10000000000	10001
1 bit	11 bits	52 bits.

Implementation:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int bi[11], f[23], sign[1], expo[8], frac[23];
int expo1[11], fract[52];
int m = 0, fl = 0, i;
// to convert decimal to binary
void binary(int n)
{
    while (n > 0)
    {
        bi[m] = n % 2;
        n = n / 2;
        m++;
    }
}
// to convert floating decimal to binary
void floating(float x)
{
    for (i = 0; i < 23; i++)
    {
        x = x * 2;
        f[i] = (int)x;
        x = x - f[i];
    }
}
// for finding single and double precision
void precision(int num)
{
    int e, ee, ee1, k = 0, j = 0, l, r = 0;

    while (m != 0)
    {
        if (bi[m] == 1)
```

```
        {
            e = m;
            ee = m + 127;
            ee1 = m + 1023;
            printf("\nSingle precision:\nBiased exponent:%d\n", ee);
            printf("\nDouble precision:\nBiased exponent:%d\n",
ee1);

            while (ee1 > 0)
            {
                expo1[r] = ee1 % 2;
                ee1 = ee1 / 2;
                r++;
            }

            printf("\n");
            printf("%d.", bi[m]);
            m--;
            for (i = m; i >= 0; i--)
            {
                frac[k] = bi[i];
                fract[k] = bi[i];
                printf("%d", frac[k]);
                k++;
            }
            for (i = 0; i < 10; i++)
            {
                frac[k] = f[i];
                fract[k] = f[i];
                printf("%d", frac[k]);
                k++;
            }
            printf(" x 2^%d", e);
            printf("\n");
            if (num > 0)
                sign[0] = 0;
            else
```

```
                sign[0] = 1;
            while (ee > 0)
            {
                expo[j] = ee % 2;
                ee = ee / 2;
                j++;
            }
            // Display
            printf("\nSingle bit precision:\n");
            printf("\nSign bit     Exponent\t \t
\t          Mantissa\n");
            printf("%d", sign[0]);
            printf("\t\t\t");
            for (i = j; i >= 0; i--)
                printf("%d", expo[i]);
            printf("\t\t\t");
            for (i = 0; i < 23; i++)
                printf("%d", frac[i]);
            printf("\n");
            // Display
            printf("\nDouble bit precision:\n");
            printf("\nSign bit     Exponent\t \t
\t          Mantissa\n");
            printf("%d", sign[0]);
            printf("\t\t\t");
            for (i = r; i >= 0; i--)
                printf("%d", expo1[i]);
            printf("\t\t\t");
            for (i = 0; i < 52; i++)
                printf("%d", fract[i]);
            break;
        }
        else
            m--;
    }
}
```

```c
int main(void)
{
    float num, x;
    int n;
    printf("Enter the no.: ");
    scanf("%f", &num);
    n = (int)fabs(num);
    x = fabs(num) - n;
    binary(n);
    floating(x);
    printf("\nIEEE Representation:\n");
    precision(num);
    return 0;
}
```

Output:

```
Enter the no.: 12.25

IEEE Representation:

Single precision:
Biased exponent:130

Double precision:
Biased exponent:1026

1.1000100000000 x 2^3

Single bit precision:

Sign bit     Exponent                          Mantissa
0                      110000010                        10001000000000000000000

Double bit precision:

Sign bit     Exponent                          Mantissa
0                      110000000010                     1000100000000000000000000000000000000000000000000000%
```

Post Lab Descriptive Questions

1. **Give the importance of IEEE-754 representation for floating point numbers?**

- The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation which was established in 1985 by the **Institute of Electrical and Electronics Engineers (IEEE)**.
- The standard addressed many problems found in the diverse floating point

implementations that made them difficult to use reliably and reduced their portability. IEEE Standard 754 floating point is the most common representation today for real numbers on computers, including Intel-based PC's, Macs, and most Unix platforms.

- There are several ways to represent floating point number but IEEE 754 is the most efficient in most cases.

Conclusion : The code for single and double precision formats to represent floating point numbers was executed successfully.

Date: _____________ **Signature of faculty in-charge**