K. J. Somaiya College of Engineering, Mumbai-77

Batch : B2	**Roll No. : 16010122151**
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 :

CO 2-Detail working of the arithmetic logic unit and its sub modules

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 :

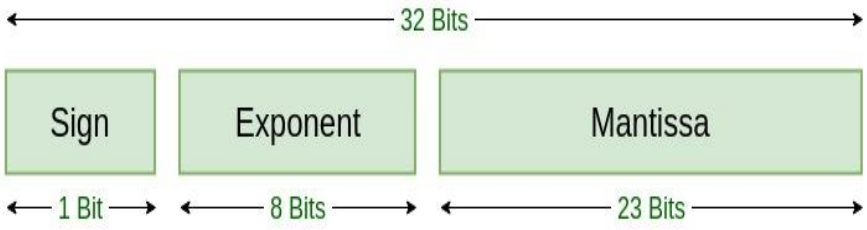
The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE). The standard addressed many problems found in the diverse floating point implementations that made them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



Single Precision
IEEE 754 Floating-Point Standard

Example:

85.125

85 = 1010101

0.125 = 001

85.125 = 1010101.001

=1.010101001 x 2^6

sign = 0

1. Single precision :

biased exponent 127+6=133

133 = 10000101

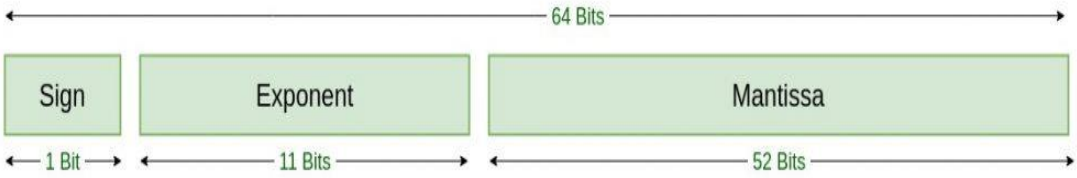
Normalised mantisa = 010101001

we will add 0's to complete the 23 bits

The IEEE 754 Single precision is:

= 0 10000101 01010100100000000000000

Example (Double Precision- 64 bit representation)



Double Precision
IEEE 754 Floating-Point Standard

85.125

85 = 1010101

0.125 = 001

85.125 = 1010101.001

=1.010101001 x 2^6

sign = 0

Double precision:

biased exponent 1023+6=1029

1029 = 10000000101

Normalised mantisa = 010101001

we will add 0's to complete the 52 bits

The IEEE 754 Double precision is:

= 0 10000000101 01010100100000000000000000000000000000000000000000000000

Implementation details :

```
#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--;
  }
}

int main(void)
 {
   float num,x;
   int n;
  printf("Enter the no.:\n");
  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.:
25.44

IEEE Representation:

Single precision:
Biased exponent:131

Double precision:
Biased exponent:1027

1.10010111000010 x 2^4

Single bit precision:

Sign bit     Exponent                Mantissa
0           010000011           10010111000010000000000

Double bit precision:

Sign bit     Exponent                Mantissa
0           0100000000011           1001011100001000000000000000000000000000000000000000000
```

Conclusion :

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

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

Post Lab Descriptive Questions :

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

Ans.

- 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.