KIET Group of Institutions, Delhi-NCR, Ghaziabad

(An ISO – 9001: 2008 Certified & 'A' Grade accredited Institution by NAAC)

Department of Information Technology

IEEE Standard for Floating Point Numbers

-Compiled by Prof. Minakshi

Topics Covered:

- Techniques to represent the real numbers
- Fixed Point and Floating Point Representation
- IEEE-754 Standard Format (Single Precision and Double Precision)
- Special Values Representation

After this lecture, a student will be able to

1. Describe the techniques to represent the real numbers.
2. Describe the IEEE 754 standard for real/fractional number representation.
3. Solve the numerical problems given on number representation according to the IEEE 754 Format.

Fractional/Real Numbers:

A number may have a binary (or decimal) point. The position of the binary point is needed to represent fractions, integers, or mixed integer-fraction numbers (for example 0.bbbb, bbbb.0, bbbb.bbb etc, where b is a binary digit).

The representation of the binary point is characterized by a position in the register.

There are two ways of specifying the position of the binary point in register:

- **By giving it a fixed position (Fixed point representation)**
- **By employing a floating-point representation (Floating Point Representation)**

Fixed point representation:

The fixed-point method assumes that the binary point is always fixed in one position.

The two positions most widely used are:

- **A binary point in the extreme left of the register to make the stored number a fraction.**
- **A binary point in the extreme right of the register to make the stored number an integer**

In either case the, binary point is not actually present, but its presence is assumed from the fact that the number stored in register is treated as a fraction or as an integer.

The fixed point representation method is not flexible. It cannot represent very small and very large numbers.

Floating Point Representation:

Floating point numbers can be used to represent very large and very small numbers. In this case, the binary point is said to float, and the numbers are called floating-point numbers.

Most processors implement the IEEE 754 (Institute of Electrical and Electronics Engineers) standard for floating-point representation and floating-point arithmetic.

In an actual typical computer, a real number is stored as per the IEEE-754 floating-point arithmetic format.

Floating Point Representation

A binary floating-point number can be represented by:

- **A sign for the number (Plus or Minus)**
- **Significand or Mantissa (S or M)**
- **A signed scale factor exponent for an implied base of 2 (E)**

We can represent a floating point number in the form given below:

$$\boxed{\pm S \times B^{\pm E}}$$

This number can be stored in a binary word with three fields:

• Sign: plus or minus

• Significand S or mantissa M

• Exponent E

Where B= Base (Base is 2 for binary number). The base B is implicit and need not be stored because it is the same for all numbers.

Any floating-point number can be expressed in many ways. The following are equivalent, where the significand is expressed in binary form:

$$\begin{array}{c} 0.110 \times 2^5 \\ 110 \times 2^2 \\ 0.0110 \times 2^6 \end{array}$$

To simplify operations on floating-point numbers, it is typically required that they be normalized. The normalized form is shown in the following figure:

$$\boxed{\pm 1.bbb\ldots b \times 2^{\pm E}}$$

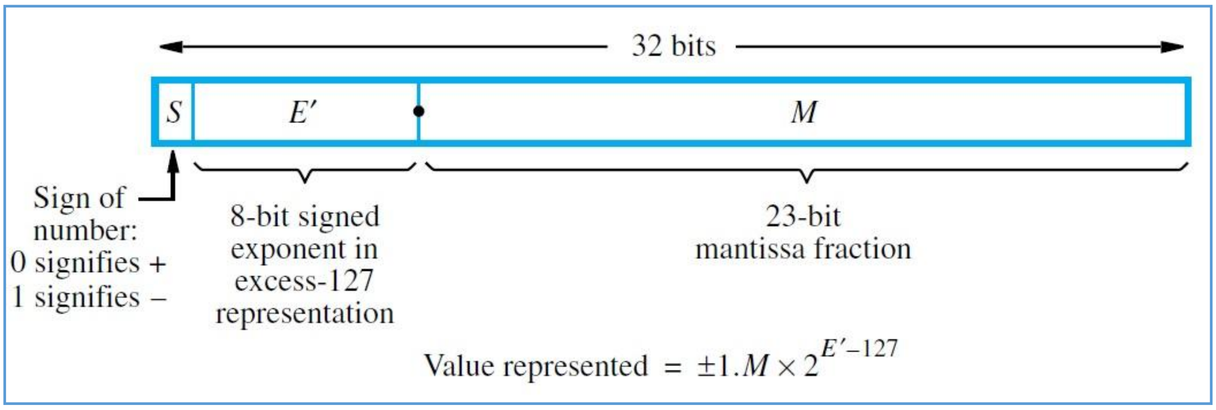
Where, b is a binary digit (0 or 1). Because the most significant bit is always one, it is unnecessary to store this bit; rather, it is implicit.

IEEE 754 Formats:

IEEE 754 defines both a 32-bit and a 64-bit format.

- **Single precision (32-bit)**
- **Double Precision (64-bit)**

Single precision (32-bit):


32 bits
S
E′
M
Sign of number: 0 signifies + 1 signifies −
8-bit signed exponent in excess-127 representation
23-bit mantissa fraction
Value represented = ±1.M × 2^{E′−127}

The leftmost bit stores the sign of the number (0= positive, 1=Negative). The exponent value is stored in the next 8 bits. The exponent representation used is known as a biased representation. In single precision representation, bias is 127.

Typically, the bias equals (2^{k-1}-1) where k is the number of bits in the binary exponent.

Biased Exponent (E') = True Exponent (E) + Bias

Biased Exponent (E') = True Exponent (E) + 127

In this case, the 8-bit field yields the numbers 0 through 255. With a bias of 127 (2^7-1), the true exponent values are in the range (-127 to +128).

The base is assumed to be 2. The remaining 23 bits, M, are used for mantissa/significand.

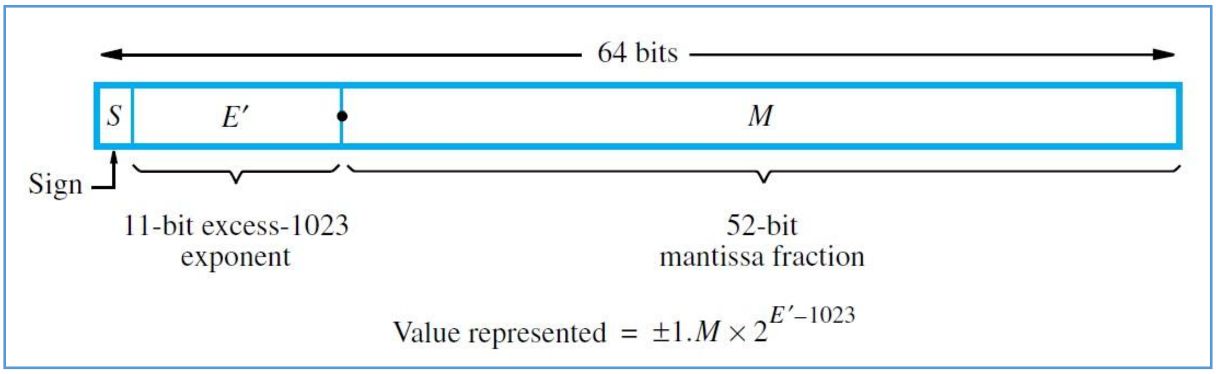
A fixed value, called the bias, is subtracted from the field of biased exponent to get the true exponent value.

Double Precision (64-bit):

In double precision IEEE-754 format, a real number is stored in 64 bits. The first bit is used for the sign (0= positive, 1=Negative), the next 11 bits are used for the biased exponent, and the rest of the bits, that is 52, are used for mantissa. In double precision representation, bias is 1023.

Biased Exponent (E') = True Exponent (E) + Bias

Biased Exponent (E') = True Exponent (E) + 1023


64 bits
S
E′
M
Sign
11-bit excess-1023 exponent
52-bit mantissa fraction
Value represented = ±1.M × 2^(E′−1023)

The following table summarized single precision and double precision methods for floating point representation.

	Total bits	Sign bit	Biased Exponent bits	Bias	Fraction bits
Single Precision	32	1	8	127	23
Double Precision	63	1	11	1023	52

Normalization and Biasing:

Two functions required to represent floating point numbers:

- Normalization
- Biasing

Normalization:

A normalized number is one in which the most significant digit of the significand/mantissa is nonzero.

For base-2 representation, in a normalized number, the most significant bit of the significand is one. The typical convention is that there is one bit to the left of the radix point. Thus, a normalized nonzero number is one in the form.

As the most significant bit is always one, so it is unnecessary to store that.

Given a number that is not normalized, the number may be normalized by shifting the radix point to the right of the leftmost bit 1 and adjusting the exponent accordingly. (Shifting right by 1 bit means divide by 2 and shifting left by 1 bit means multiply by 2.

Biasing:

Exponent is stored in biased representation.

In single precision case, 8-bit field yields the numbers 0 through 255. With the bias of 127 (2^7-1), the true exponent values are in range (-127) to (+128)

In double Precision case, 11-bit field yields the numbers 0 through 2047. With a bias of 1023 (2^{10}-1), the true exponent values are in range (-1023) to (+1024). The exponent is calculated according to the following equations.

Biased Exponent (E') = True Exponent (E) + Bias

True Exponent (E) = Biased Exponent (E') - Bias

Numerical Problem

Question 1: Represent (-1234.125)10 into single precision and double precision methods.

Step 1: first convert the number $(124.125)_{10}$ into binary number

$(124.125)10 = (10011010010.001)_2$

Step 2: Normalize this number. Move the binary point right to the leftmost 1.

$1.0011010010001x\ 2^{+10}$

Single precision: 32 bit Representation

Calculate biased exponent.

Biased exponent E’	= E + bias =+10 +127
	=+137
	$=(10001001)_2$

Sign = Negative (sign bit=1)

Sign	Exponent (E’)	Mantissa
1	10001001	00110100100010000000000

Double precision: 64 bit Representation

Calculate biased exponent.

Biased exponent E’	= E + bias =+10 +1023
	=+1033
	$=(10000001001)_2$

Sign	Exponent (E’)	Mantissa
1	10000001001	0011010010001000000000000000000000000000000000000000

Note: Add zeros to the right of mantissa to make mantissa bits 23 in single precision or 64 bits in double precision.

Question 2:

Represent 0001001101001.001101 using single precision and double precision methods.

Normalize this number. Move the binary point right to the leftmost 1.

1. 001101001001101x 2^{+9}

Single precision: 32 bit Representation

Calculate biased exponent.

Biased exponent E’	= E + bias =+9 +127
	=+136
	=$(10001000)_2$

Sign = positive (sign bit=0)

Sign	Exponent (E’)	Mantissa
0	10001000	00110100100110100000000

Double precision: 64 bit Representation

Calculate biased exponent.

Biased exponent E’	= E + bias =+9 +1023
	=+1032
	=$(10000001000)_2$

Sign	Exponent (E’)	Mantissa
0	10001000	0011010010011010000000000000000000000000000000000000

Question 3: Represent (-0.00000100011001101) using single precision and double precision methods.

Normalize this number. Move the binary point right to the leftmost 1.

$1.00011001101 \times 2^{-6}$

Single precision: 32 bit Representation

Calculate biased exponent.

Biased exponent E'	**= E + bias** **=-6 +127**
	=+121
	$=(01111001)_2$

Sign = negative (sign bit=1)

Sign	Exponent (E')	Mantissa
1	01111001	00011001101000000000000

Double precision: 64 bit Representation

Calculate biased exponent.

Biased exponent E'	**= E + bias** **=-6 +1023**
	=+1017
	$=(01111111001)_2$

Sign	Exponent (E')	Mantissa
1	01111111001	0001100110100000000000000000000000000000000000000000

Question 4: Find the value represented by the following floating point representation.

Sign	Exponent (E’)	Mantissa
0	10001000	00110100100110100000000

Sign bit is 0➔	It denotes the number is positive.
Biased exponent (E’)➔	$(10001000)_2$
	$= (136)_{10}$
True Exponent E	= E’- Bias
	= 136-127
	= +9
Mantissa	= 001101001001101
Normalized Number	$= +1.\ 001101001001101 \times 2^{+9}$
	= +1001101001.001101
	$= +(1x2^{9}+1x2^{6}+1x2^{5}+1x2^{3}+1x2^{0}+1x2^{-3}+1x2^{-4}+1x2^{-6}$
	= + (1x512+1x64+1x32+1x8+1x1+1x0.125+1x0.0625+1x0.015625)
	= + (512+64+32+8+1+0.125+0.0625+0.015625)
	= +(617+0.203125)
Decimal value	= +617.203125

Special Values Representation

When E' =00………00000

M = 0000………….0000	**Represents the number zero**
M ≠ 00000…………0000	**Represents the number very close to zero**

When E' =11..……11111

M = 0000…………….0000	**Represents the value infinity**
M ≠ 0000.…………….0000	**Represents NaN (Not a Number) or invalid number**

Important Note:

For exponent values in the range of 1 through 254 for single format and 1 through 2046 for double format, normalized nonzero floating-point numbers are represented.

The exponent is biased, so that the range of exponents is -126 through -127 for single format and -1022 through +1023. A normalized number requires a 1 bit to the left of the binary point; this bit is implied, giving an effective 24-bit or 53-bit significand (called fraction in the standard).

An exponent of zero together with a fraction of zero represents positive or negative zero, depending on the sign bit. As was mentioned, it is useful to have an exact value of 0 represented.

An exponent of all ones together with a fraction of zero represents positive or negative infinity, depending on the sign bit. It is also useful to have a representation of infinity. This leaves it up to the user to decide whether to treat overflow as an error condition or to carry the value infinity and proceed with whatever program is being executed.

An exponent of zero together with a nonzero fraction represents a denormalized number. In this case, the bit to the left of the binary point is zero and the true exponent is -126 or -1022. The number is positive or negative depending on the sign bit.

An exponent of all ones together with a nonzero fraction is given the value NaN, which means Not a Number, and is used to signal various exception conditions.