

10 Floating-Point Numbers

Base 10:

$$(12.34)_{10} = 12 + \frac{34}{100} \quad \text{or} \quad (12.34)_{10} = 12 + \frac{3}{10} + \frac{4}{100} = 1 \cdot 10^1 + 2 \cdot 10^0 + 3 \cdot 10^{-1} + 4 \cdot 10^{-2}$$

Base 2:

$$(101.11)_2 = 5 + \frac{3}{4} \quad \text{or} \quad (101.11)_2 = 5 + \frac{1}{2} + \frac{1}{4}, \quad (0.1111)_2 = \frac{15}{16}, \quad (0.1)_2 = \frac{1}{2}$$

To store real numbers (too small / big numbers) in the memory the exponential notation (scientific notation) is used.

10.1 Scientific notation, exponential notation:

$\pm F \times B^{\pm E}$

F: Fraction (mantissa, significand)

E: Exponent

B: Base

$\pm F$ and $\pm E$ are stored in the memory.

The base **B** is implicit and does not need to be stored. It is the same (in computers $B=2$) for all numbers.

Examples (Base 10):

a)

$$976,000,000,000,000 = +0.976 \times 10^{15}$$

+976 and +15 are stored in the memory.

b)

$$0.000\,000\,000\,000\,976 = +0.976 \times 10^{-12}$$

+976 and -12 are stored in the memory.

Normalized Number:

A number can be written in exponential form in many ways.

For example $3.14 = 314 \times 10^{-2} = 3.14 \times 10^0 = 0.314 \times 10^1$

In normalized notation the location of the radix point is predetermined.

For example, the exponent (E) can be chosen so that the point is always to the left of the most significant digit.

Example:

3.14 normalization $\rightarrow 0.314 \times 10^1$

This number can be stored as $\pm F \pm E \rightarrow +314 +01$

It is not necessary to store the base (now it is 10) and the location of the point.

There are also other methods to normalize the numbers.

For example the point can be placed to the right of the most significant digit.

Biased Exponent:

Exponent values are signed and they can be negative (small numbers) and positive.

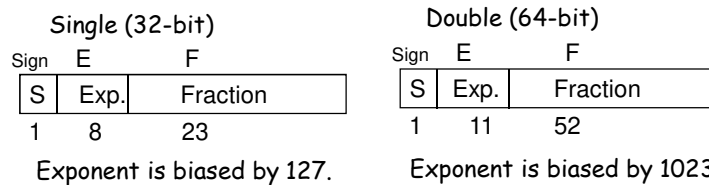
It is difficult to compare signed numbers (2's complement).

To make the comparison easier the exponent is biased (added with a fixed bias) before being stored, by adjusting its value to make it unsigned suitable for comparison. (Now we don't need to store the sign of the exponent)

10.2 IEEE 754 Standard (1985, updated in 2008)

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).

Many hardware floating point units and compilers use the IEEE 754 standard.



If E consists of k bits the exponent value is biased by adding $(2^{k-1} - 1)$.

In the current standard there are also half (16-bit) and quadruple (128-bit) numbers.

Normalized Number (IEEE 754):

The radix point is to the right of the most significant digit that is not zero.

As we are working with base-2 the digit different than "0" is "1".

Example:

$$(10110.101)_2 \xrightarrow{\text{(normalization)}} 1.0110101 \times 2^4$$

As there is always a "1" on the left side of the point, this "1" is not stored in the memory either.

This "1" is called **hidden one**.

Example:How to store $(+22.625)_{10}$ in IEEE single format?

$$(22)_{10} = (10110)_2$$

Converting $(.625)_{10}$ to base 2:

$$2 \times 0.625 = 1 + 0.25$$

$$2 \times 0.25 = 0 + 0.5$$

$$2 \times 0.5 = 1 + 0$$

Most significant bit

$$\Rightarrow (0.625)_{10} = (0.101)_2$$

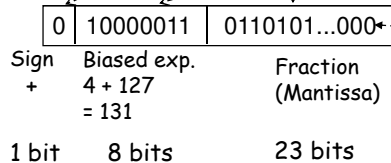
5/8

Hidden 1 is not stored.

Base 2, implicit, not stored.

$$(+22.625)_{10} = (+10110.101)_2 = +1.0110101 \times 2^4 \text{ (Normalized)}$$

IEEE Single:



The least significant bits of the fraction are filled with "0", because the point is on the left.

The value of the number in Base 10 (N):

Single: $N = (-1)^S (1.F) 2^{E-127}$

Double: $N = (-1)^S (1.F) 2^{E-1023}$

$$N = (-1)^S \left(1 + \frac{F}{2^{23}}\right) 2^{E-127}$$

$$N = (-1)^S \left(1 + \frac{F}{2^{52}}\right) 2^{E-1023}$$

To represent **zero** a special notation is used: $E=0$ and $F=0 \rightarrow N=0$ **Examples (Single):**

S	E	F	Value (Meaning):
0	10010011	101000100000000000000000	$= +1.1010001 \times 2^{10100} = +1.6328125 \times 2^{20}$
1	10010011	101000100000000000000000	$= -1.1010001 \times 2^{10100} = -1.6328125 \times 2^{20}$
0	01101011	101000100000000000000000	$= +1.1010001 \times 2^{-10100} = +1.6328125 \times 2^{-20}$
1	01101011	101000100000000000000000	$= -1.1010001 \times 2^{-10100} = -1.6328125 \times 2^{-20}$

Special Cases:1. $E = 0$ and $F = 0 \rightarrow N = 0$ There are two zeros (+,-).2. $E = 255$, $F = 0 \rightarrow N = \pm\infty$ 3. $E = 255$, $F \neq 0 \rightarrow \text{NaN (Not a Number)}$ 0/0, ∞/∞ 4. $E = 0$, $F \neq 0 \rightarrow$ Denormalized number

The last case is used for numbers with too small magnitudes (absolute values) which cannot be represented using the normalized notation.

Denormalized Numbers:

With the normalized notation numbers with too small magnitudes cannot be expressed.

The smallest normalized number: $S = 0$, $E = 0000\ 0001$, $F = 000\dots 0$

$$N = +(1+0)2^{1-127} = 2^{-126}$$

Numbers between 0 and 2^{-126} cannot be expressed as normalized numbers.

The value of denormalized numbers ($E = 0$, $F \neq 0$) is calculated as follows:

$$\text{Single: } N = (-1)^S \left(\frac{F}{2^{23}}\right) 2^{-126} \quad \text{Double: } N = (-1)^S \left(\frac{F}{2^{52}}\right) 2^{-1022}$$

Denormalized smallest number: $S = 0$, $E = 0000\ 0000$, $F = 000\dots 01$

$$N = +(1/2^{23})2^{-126} = 2^{-149}$$

Numbers between 0 and 2^{-149} cannot be expressed. (*Underflow*).

Limit values:

The number with the smallest magnitude:

Single: 2^{-149} (Denormalized) , Double: 2^{-1074} (Denormalized)

The number with the largest magnitude :

Single: $0\ 11111110\ 111111111111111111111111 = (2-2^{-23}) \times 2^{127} \approx 10^{38.53}$ (Normalized)

Double: $(2-2^{-52}) \times 2^{1023} \approx 10^{308.3}$ (Normalized)

Example: Addition and subtraction of floating point numbers
An arithmetic pipeline

$$X = A \cdot 2^a$$

$$Y = B \cdot 2^b$$

