Machine numbers: how floating point numbers are stored?

Floating-point number representation

What do we need to store when representing floating point numbers in a computer?

$$x = \pm 1.\boldsymbol{f} \times 2^{\boldsymbol{m}}$$

Initially, different floating-point representations were used in computers, generating inconsistent program behavior across different machines.

Around 1980s, computer manufacturers started adopting a standard representation for floating-point number: IEEE (Institute of Electrical and Electronics Engineers) 754 Standard.

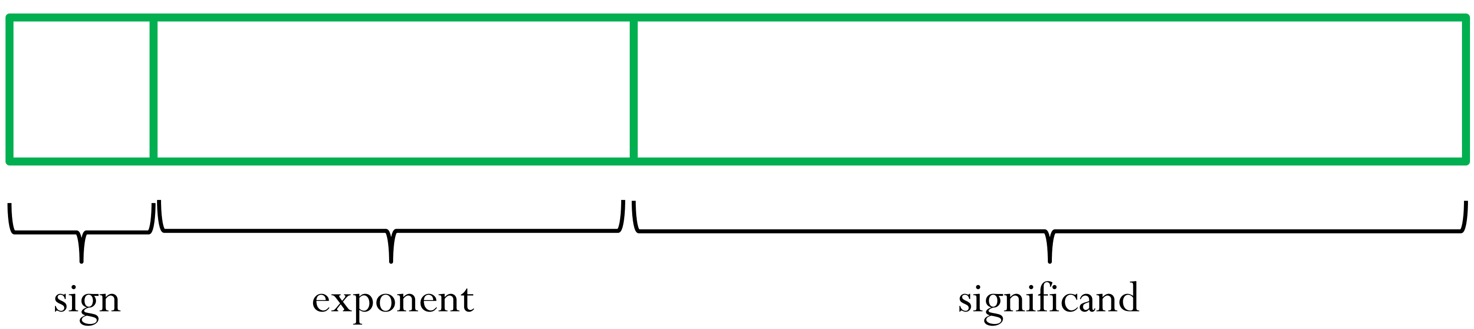
Floating-point number representation

Numerical form:

$$x = \pm 1.\boldsymbol{f} \times 2^{\boldsymbol{m}}$$

Representation in memory:

$x =$



Precisions:

Finite representation: not all numbers can be represented exactly!

$x = \pm 1.\boldsymbol{f} \times 2^{\boldsymbol{c-shift}}$

IEEE-754 Single precision (32 bits):

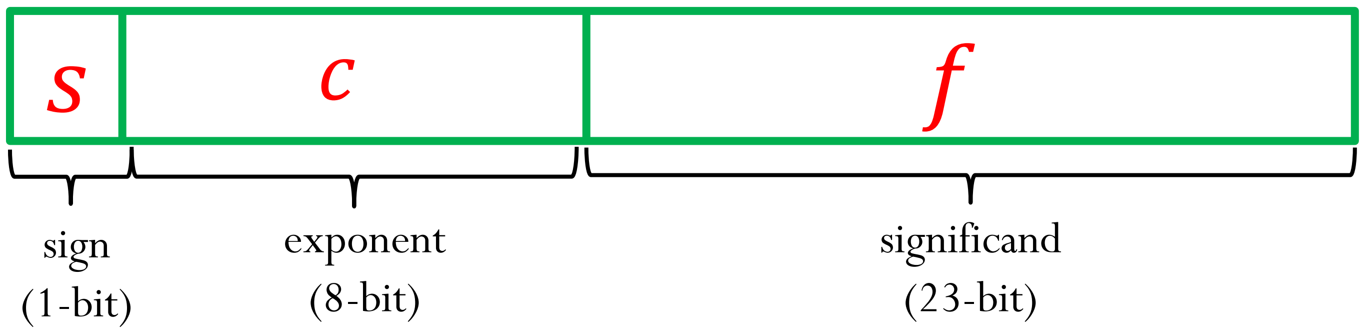
$x =$ 

IEEE-754 Double precision (64 bits):

$x =$ 

IEEE-754 Single Precision (32-bit)

$$x = (-1)^{s}\, 1.f \times 2^{m} \qquad m = c - shift$$



IEEE-754 Single Precision (32-bit)

$$x = (-1)^{\boldsymbol{s}}\, 1.\boldsymbol{f} \times 2^{\boldsymbol{m}}$$

Example: Represent the number $x = -67.125$ using IEEE Single-Precision Standard

$$67.125 = (1000011.001)_2 = (1.000011001)_2 \times 2^6$$

IEEE-754 Single Precision (32-bit)

$$x = (-1)^{\boldsymbol{s}}\, 1.\boldsymbol{f} \times 2^{\boldsymbol{m}} = \boxed{\;\boldsymbol{s}\;}\boxed{\qquad \boldsymbol{c} \qquad}\boxed{\qquad\quad \boldsymbol{f} \qquad\quad} \qquad c = m + 127$$

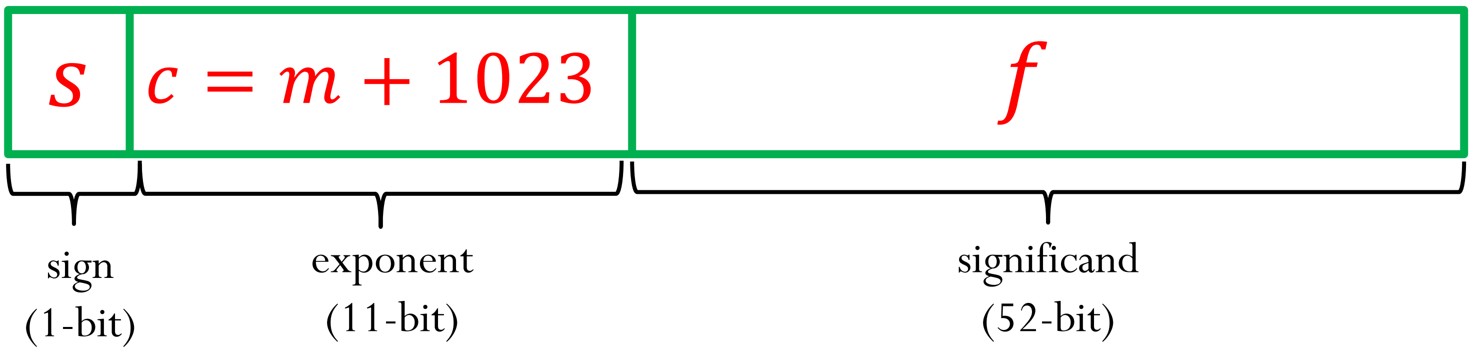
- **Machine epsilon** (ϵ_m): is defined as the distance (gap) between 1 and the next larger floating point number.

- **Smallest positive normalized FP number:**

- **Largest positive normalized FP number:**

IEEE-754 Double Precision (64-bit)

$$x = (-1)^{s}\, 1.f \times 2^{m}$$



$s = 0$: positive sign, $s = 1$: negative sign

Reserved exponent number for special cases:

$c = (00000000000)_2 = 0$

$c = (11111111111)_2 = 2047$

Therefore $1 \leq c \leq 2046$

IEEE-754 Double Precision (64-bit)

$$x = (-1)^{s}\, 1.f \times 2^{m} = \boxed{s}\boxed{c}\boxed{f} \qquad c = m + 1023$$

- **Machine epsilon** (ϵ_m): is defined as the distance (gap) between 1 and the next larger floating point number.

$$(1)_{10} = \boxed{0}\boxed{0111\ldots111}\boxed{000000000000\ldots000000000}$$

$$(1)_{10} + \epsilon_m = \boxed{0}\boxed{0111\ldots111}\boxed{000000000000\ldots000000001}$$

$$\boldsymbol{\epsilon_m = 2^{-52}} \approx 2.2 \times 10^{-16}$$

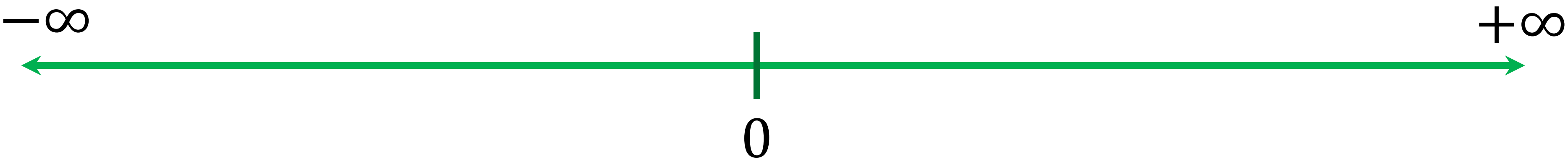
- **Smallest positive normalized FP number:**

$$\mathbf{UFL} = 2^{L} = 2^{-1022} \approx 2.2 \times 10^{-308}$$

- **Largest positive normalized FP number:**

$$\mathbf{OFL} = 2^{U+1}(1 - 2^{-p}) = 2^{1024}(1 - 2^{-53}) \approx 1.8 \times 10^{308}$$

Normalized floating point number scale (single precision)



Special Values:

$$x = (-1)^{s}\, 1.f \times 2^{m} = \boxed{s \mid c \mid f}$$

1) **Zero**:

$$x = \boxed{s \mid 000 \ldots 000 \mid 0000 \ldots \ldots 0000}$$

2) **Infinity**: $+\infty$ $(s = 0)$ and $-\infty$ $(s = 1)$

$$x = \boxed{s \mid 111 \ldots 111 \mid 0000 \ldots \ldots 0000}$$

3) **NaN**: (results from operations with undefined results)

$$x = \boxed{s \mid 111 \ldots 111 \mid anything \neq 00 \ldots 00}$$

Subnormal (or denormalized) numbers

- Noticeable gap around zero, present in any floating system, due to normalization
  - ✓ The smallest possible significand is 1.00
  - ✓ The smallest possible exponent is L
- Relax the requirement of normalization, and allow the leading digit to be zero, only when the exponent is at its minimum ($m = L$)

$$x = (-1)^{\boldsymbol{s}}\, 0.\boldsymbol{f} \times 2^{\boldsymbol{L}}$$

Subnormal (or denormalized) numbers

Another special case:

$$x = \boxed{s} \boxed{c = 000 \ldots 000} \boxed{\quad f \quad}$$

$$x = (-1)^{s} \, 0.f \times 2^{L}$$

Note that this is a special case, and the exponent $\boldsymbol{m}$ is **not** evaluated as $\boldsymbol{m = c - shift = -shift}$.
Instead, the exponent is set to the lower bound, $\boldsymbol{m} = \mathbf{L}$

- PROS: More gradual underflow to zero
- CONS: - Computations with subnormal numbers are often slow;
  - Loss of precision

Subnormal (or denormalized) numbers

IEEE-754 Single precision (32 bits):

$c = (00000000)_2 = 0$

Exponent set to $m = -126$

Smallest positive subnormal FP number:

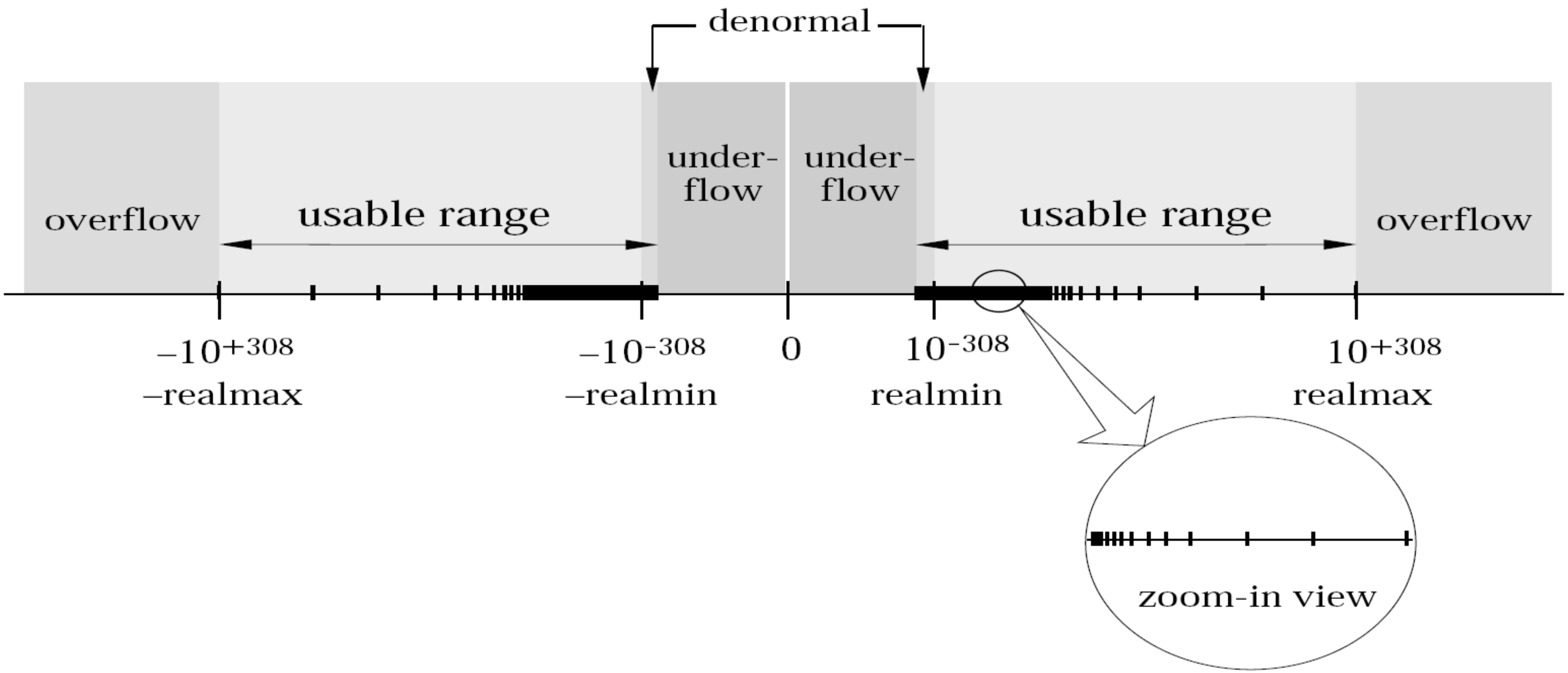
IEEE-754 Double precision (64 bits):

$c = (00000000000)_2 = 0$

Exponent set to $m = -1022$

Smallest positive subnormal FP number:

IEEE-754 Double Precision



Summary for Single Precision

$x = (-1)^{s}\, 1.\boldsymbol{f} \times 2^{\boldsymbol{m}} = \boxed{s \mid c \mid f}$ $\quad m = c - 127$

Stored binary exponent (c)	Significand fraction (f)	value
00000000	0000…0000	zero
00000000	$any\ f \neq 0$	$(-1)^{s}\, 0.\boldsymbol{f} \times 2^{-126}$
00000001	$any\ f$	$(-1)^{s}\, 1.\boldsymbol{f} \times 2^{-126}$
⋮	⋮	⋮
11111110	$any\ f$	$(-1)^{s}\, 1.\boldsymbol{f} \times 2^{127}$
11111111	$any\ f \neq 0$	NaN
11111111	0000…0000	infinity