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:

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

Representation in memory:

$m \in [L, U]$

$m \in [-4, 4]$

$x =$

sign	exponent	significand
$\pm$	m	f

$c = m + \text{shift}$

unsigned int

signed int

Precisions:

Finite representation: not all numbers can be represented exactly!

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

IEEE-754 Single precision (32 bits):

$c = m + \underline{\underline{shift}}$

$x =$

±	c	f
1bit	8 bits	23 bits

IEEE-754 Double precision (64 bits):

$x =$

±	11 bits	52 bits f
1bit	c	

IEEE-754 Single Precision (32-bit)

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

1

s	c (8)	f (23)
sign (1-bit)	exponent (8-bit)	significand (23-bit)

$n = 23$

$\boxed{p = 24}$

$(00000000)_2 = (0)_{10}$

$(11111111)_2 = (255)_{10}$

$0 \leq c \leq 255$

reserve 0 , 255 $\longrightarrow$ special cases

$1 \leq c \leq 254 \longrightarrow 1 \leq m + shift \leq 254$

set $\boxed{shift = 127}$ $\longrightarrow$ $\boxed{-126 \leq m \leq 127}$ $\quad m \in [-126, 127]$

IEEE-754 Single Precision (32-bit)

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

S → 0 → $(-1)^0$ ⇒ Positive

S → 1 → $(-1)^1$ ⇒ Negative

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

$$(67.125)_{10} = \underline{(1000011.001)_2} = \underline{(1.\boxed{000011001})_2 \times 2^{6}}$$

$m = 6$

$c = m + 127$

$c = (133)_{10}$

$S = 0$

$f = \underbrace{0000110010000\ldots0}_{23 \text{ bits}}$

$c = (\underbrace{10000101}_{8 \text{ bits}})_2$

$\underbrace{0}_{1 \text{ bit}}\ \underbrace{10000101}_{8 \text{ bits}}\ \underbrace{0000110010\ldots0}_{23 \text{ bits}}$

IEEE-754 Single Precision (32-bit)

$p = 24$

$$x = (-1)^s \, 1.f \times 2^m = \begin{array}{|c|c|c|}\hline s & \overset{8}{c} & \overset{23}{f} \\ \hline\end{array} \qquad c = m + \underline{\underline{127}}$$

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

$$(1)_{10} = 1.\underbrace{000 \cdots 00}_{23 \text{ bits}} \times 2^0 \qquad 0.000 \cdots 01 \times 2^0$$

$$= 1.\underbrace{000 \cdots 01}_{23 \text{ bits}} \times 2^0 \qquad \boxed{\epsilon_m = 2^{-23}} \qquad 2^{-23}$$

$$\epsilon_m = 2^{-n}$$

- **Smallest positive normalized FP number:**

$$UFL = 2^L = 2^{-126} \approx 10^{-38}$$

- **Largest positive normalized FP number:**

$$OFL = 2^{U+1}(1 - 2^{-p}) = 2^{128}(1 - 2^{-24}) \cong 10^{38}$$

IEEE-754 Double Precision (64-bit)

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

$p = 53$ $(n+1)$

s	$c = m + 1023$	f
sign (1-bit)	exponent (11-bit)	significand (52-bit)

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

Reserved exponent number for special cases:

$$\begin{cases} c = (00000000000)_2 = 0 \\ c = (11111111111)_2 = 2047 \end{cases}$$

$c = m + \text{shift}$

Therefore $1 \leq c \leq 2046$ → $1 \leq m + \text{shift} \leq 2046$

shift = 1023

$-1022 \leq m \leq 1023$ → $\boxed{m \in [-1022, 1023]}$

IEEE-754 Double Precision (64-bit)

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

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

$\epsilon_m = 2^{-n}$

$n = 52$

$(1)_{10} =$	0	0111 … 111	000000000000 … 000000000
$(1)_{10} + \epsilon_m =$	0	0111 … 111	000000000000 … 000000001

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

- **Smallest positive normalized FP number:**

$m \in [-1022, 1023]$

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

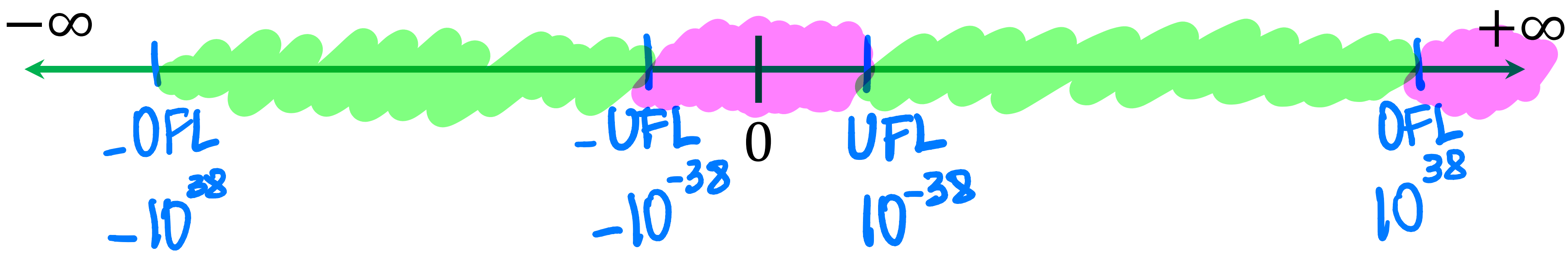
$p = 52$

$U = 1023$

- **Largest positive normalized FP number:**

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

Normalized floating point number scale (single precision)



∞ , zero

Special Values:

$c = (0000 \ldots 0)$
$c = (111 \ldots 11)$

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

1) **Zero**:

$$x = \boxed{s}\underbrace{\boxed{000\ldots000}}_{8,\,11}\underbrace{\boxed{0000\ldots\ldots0000}}_{23,\,52}$$

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

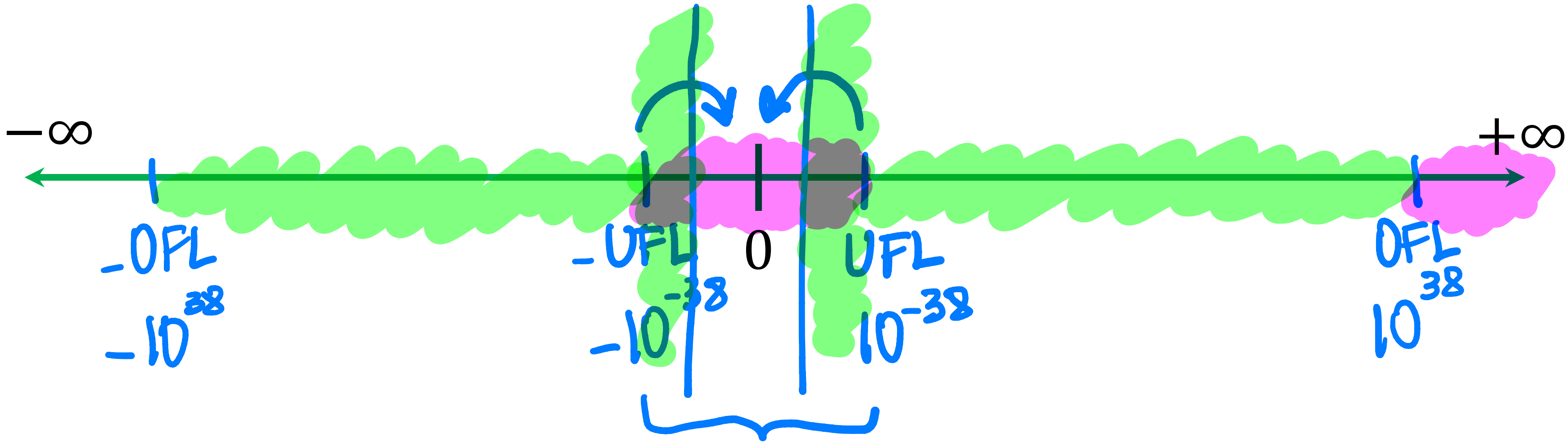
$$x = \boxed{s}\underbrace{\boxed{111\ldots111}}_{8,\,11}\underbrace{\boxed{0000\ldots\ldots0000}}_{23,\,52}$$

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

$$x = \boxed{s}\underbrace{\boxed{111\ldots111}}_{8,\,11}\underbrace{\boxed{anything \neq 00\ldots00}}_{(100\ldots010)}$$

4) $c = (000 \ldots 0)$ $f = (\text{anything}) \rightarrow$ subnormal

Normalized floating point number scale (single precision)



$$1.f \times 2^{m} \longrightarrow 1.f \times 2^{L}$$

relax

$$\boxed{0.f \times 2^{L}}$$

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 ($m = L$)

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

★ $m = c - \text{shift}$

$m = L$

$c = (0000\ldots) \longrightarrow$ subnormal

Subnormal (or denormalized) numbers

IEEE-754 Single precision (32 bits):

$c = (00000000)_2 = 0$

Exponent set to $m = -126$ $\rightarrow$ $0.f \times 2^{-126}$

Smallest positive subnormal FP number:

$0.0000\ldots01 \times 2^{-126} = 2^{-23} \times 2^{-126} \simeq 1.4 \times 10^{-45}$

IEEE-754 Double precision (64 bits):

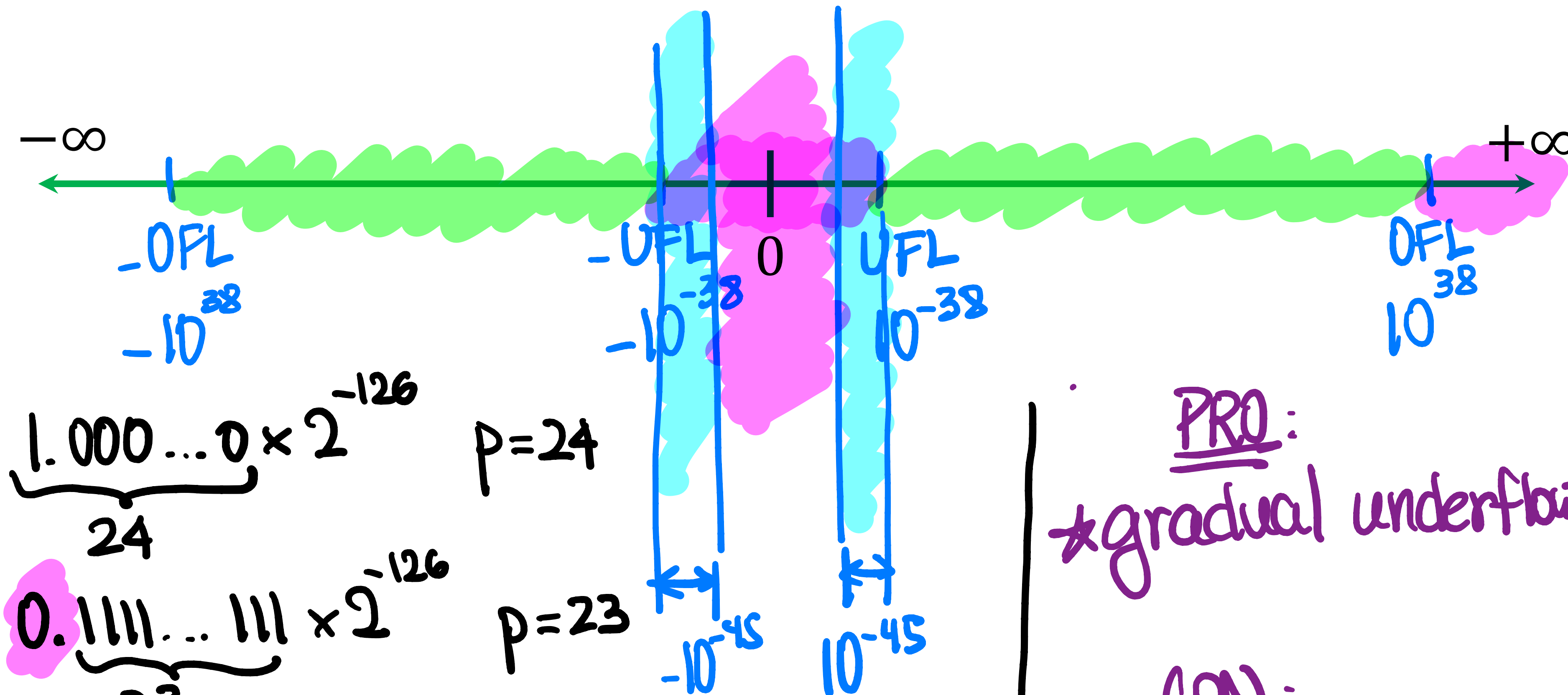
$c = (00000000000)_2 = 0$

Exponent set to $\boxed{m = -1022}$ $\rightarrow$ $0.f \times 2^{-1022}$

Smallest positive subnormal FP number:

$0.\underbrace{000\ldots001}_{52} \times 2^{-1022} = 2^{-52} \times 2^{-1022} \simeq 10^{-324}$

Normalized floating point number scale (single precision)



$\underbrace{1.000\ldots0}_{24} \times 2^{-126}$ $p=24$

$0.\underbrace{1111\ldots111}_{23} \times 2^{-126}$ $p=23$

$0.\underline{\underline{00}}\underbrace{1111\ldots11} \times 2^{-126}$ $p=21$

$0.000\ldots00\underbrace{1010} \times 2^{-126}$ $p=4$

PRO:

* gradual underflow

CON:

* Loss of precision

Subnormal (or denormalized) numbers

Another special case:

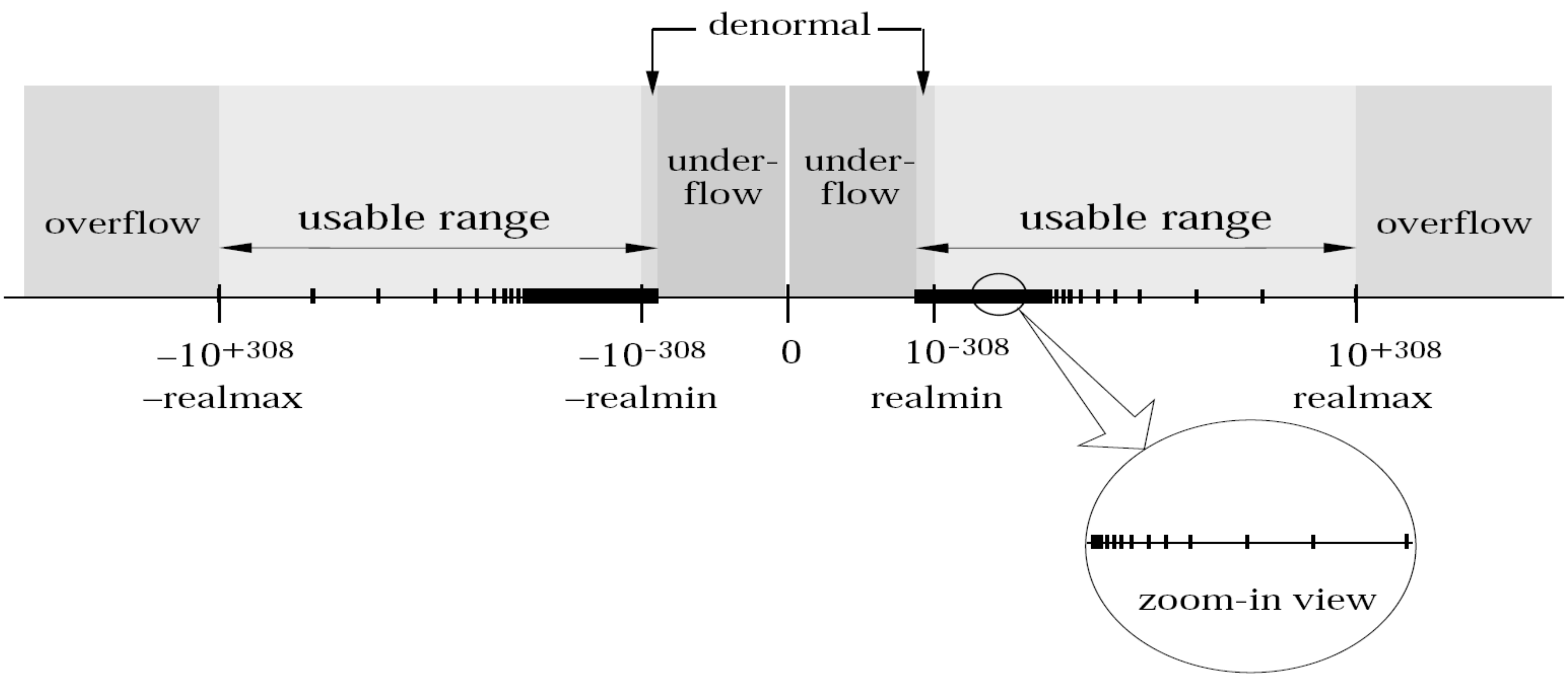
$$x = \boxed{s} \boxed{c = 000 \ldots 000} \boxed{\quad f \quad}$$

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

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

IEEE-754 Double Precision



Summary for Single Precision

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

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