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?

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

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

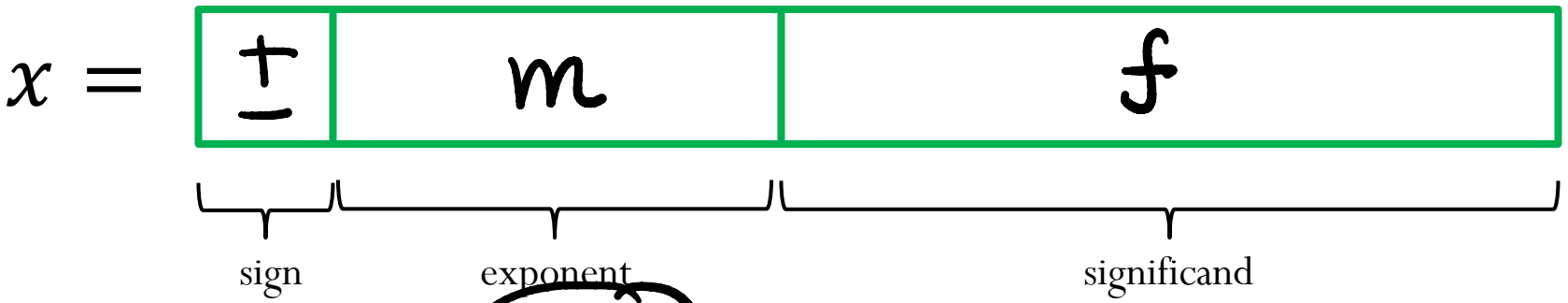
Numerical form:

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

Representation in memory:

$m \in [L, U]$

$m \in [-4, 4]$



$c = m + \text{shift}$

c: unsigned int

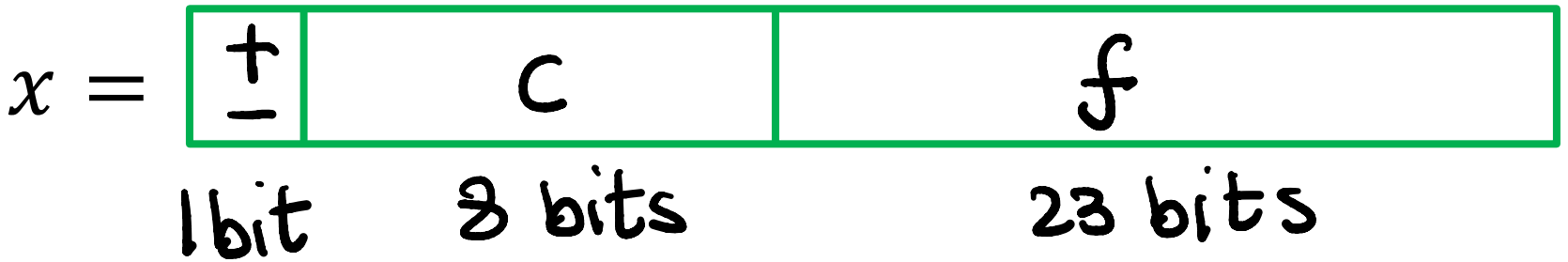
shift: signed int

Precisions:

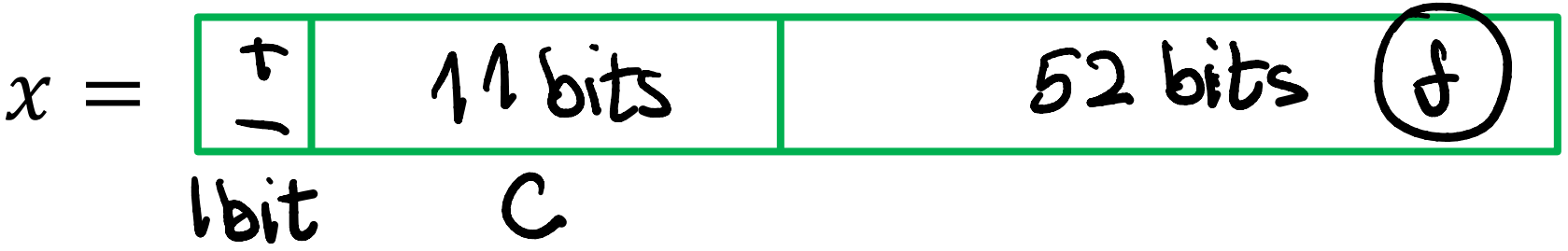
Finite representation: not all numbers can be represented exactly!

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

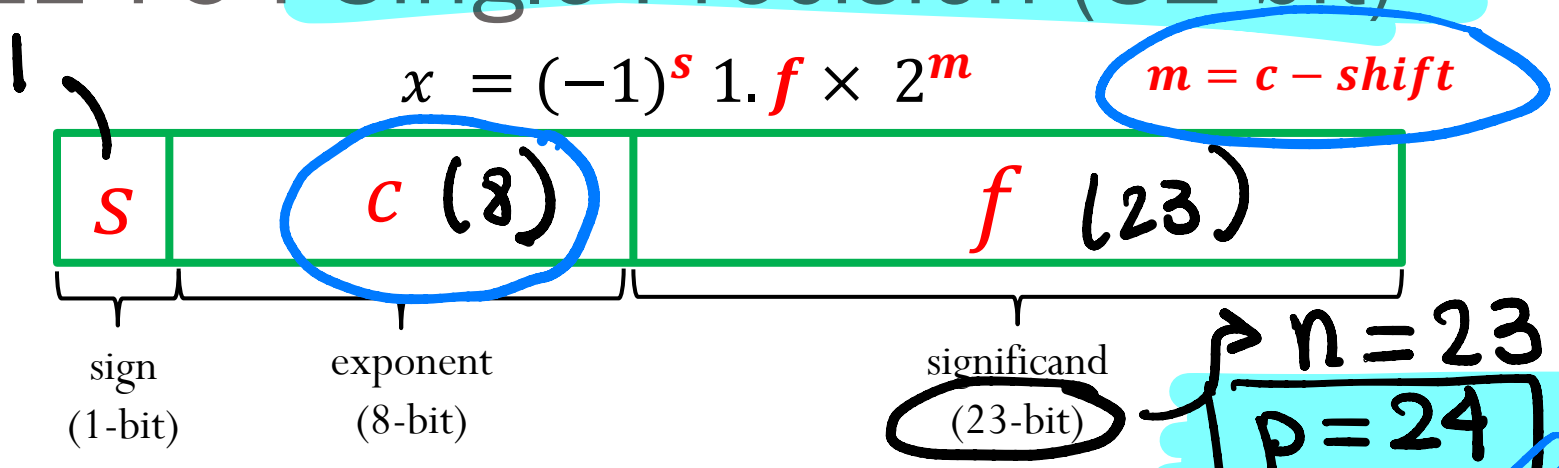
IEEE-754 Single precision (32 bits): $c = m + shift$



IEEE-754 Double precision (64 bits):



IEEE-754 Single Precision (32-bit)



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

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

$0 < c < 255$

reserve 0 , 255 ⟶ special cases

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

set shift = 127 ⟶ $-126 \leq m \leq 127$

$m \in [-126, 127]$

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

$(67.125)_{10} = (1000011.001)_2 = (1.000011001)_2 \times 2^6$

m = 6

C = m + 127

$C = (133)_{10}$

S = 0

f = 00001100100000...0 (23 bits)

$C = (10000101)_2$ (8 bits)

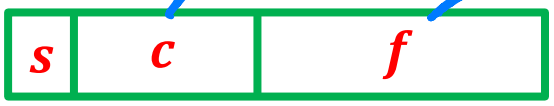
0 | 10000101 | 00011010...0

1 bit | 8 bits | 23 bits

IEEE-754 Single Precision (32-bit)

p = 24

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



$c = m + 127$

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

$(1)_{10} = 1.\underbrace{000\ldots00}_{23\text{ bits}} \times 2^0 \ominus 0.000\ldots01 \times 2^0$

$= 1.\underbrace{000\ldots01}_{23\text{ bits}} \times 2^0$

2^{-23}

$\epsilon_m = 2^{-23}$

$\epsilon_m = 2^{-p}$

- **Smallest positive normalized FP number:**

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

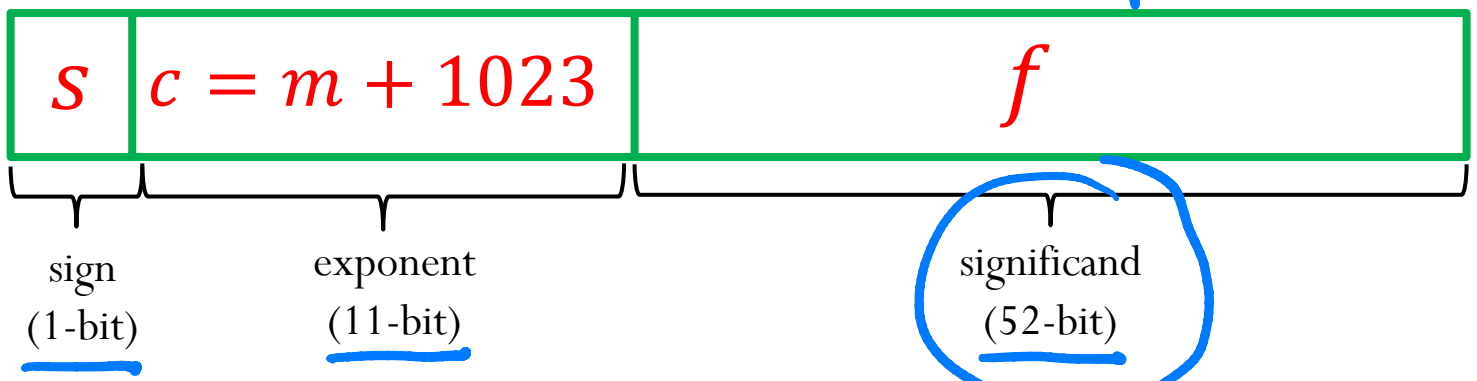
- **Largest positive normalized FP number:**

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

IEEE-754 Double Precision (64-bit)

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

p = 53 (n+1)



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

Reserved exponent number for special cases:

$c = (00000000000)_2 = 0$

$c = (11111111111)_2 = 2047$

c = m + shift

Therefore $1 \leq c \leq 2046$ → $1 \leqslant m + shift \leqslant 2046$

shift = 1023

$-1022 \leqslant m \leqslant 1023$ → $m \in [-1022, 1023]$

IEEE-754 Double Precision (64-bit)

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

- **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 |

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

- **Smallest positive normalized FP number:**

$m \in [-1022, 1023]$

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

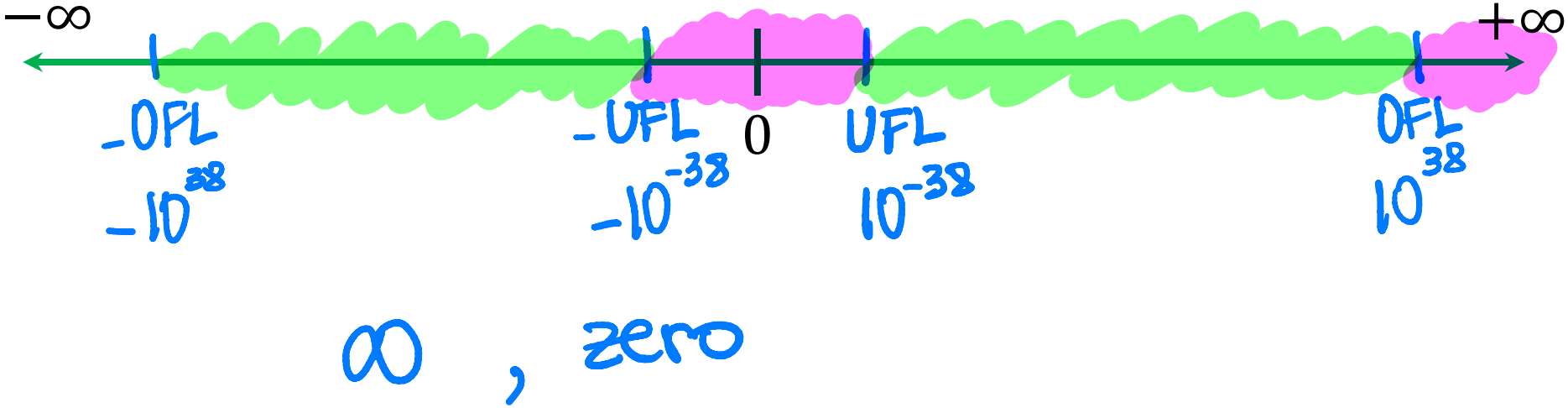
$p = 52$

$U = 1023$

- **Largest positive normalized FP number:**

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

Normalized floating point number scale (single precision)



∞ , zero

Special Values:

c = (000 ... 0)
c = (111 ... 11)

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

s	c	f

1) **Zero**:

$$x =$$

s	000 ... 000	0000 0000

8, 11 | 23, 52

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

$$x =$$

s	111 ... 111	0000 0000

8, 11 | 23, 52

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

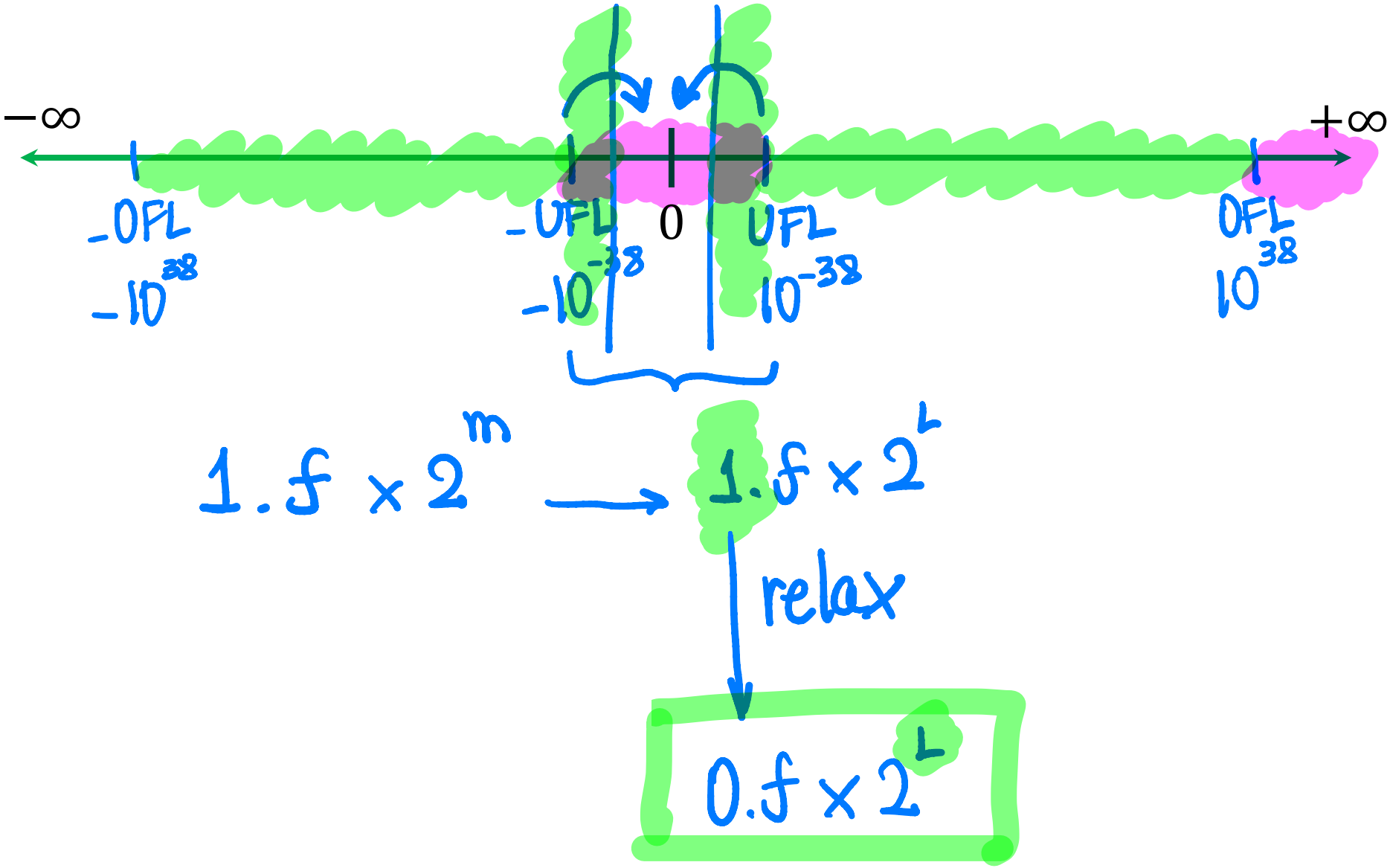
$$x =$$

s	111 ... 111	$anything \neq 00 \ldots 00$

8, 11 | (100 ... 010)

4) c = (000 ... 0) f = (anything) → subnormal

Normalized floating point number scale (single precision)



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 ($m = L$)

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

★

$m = c - \text{shift}$

$m = L$

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

Subnormal (or denormalized) numbers

IEEE-754 Single precision (32 bits):

$c = (00000000)_2 = 0$

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

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):

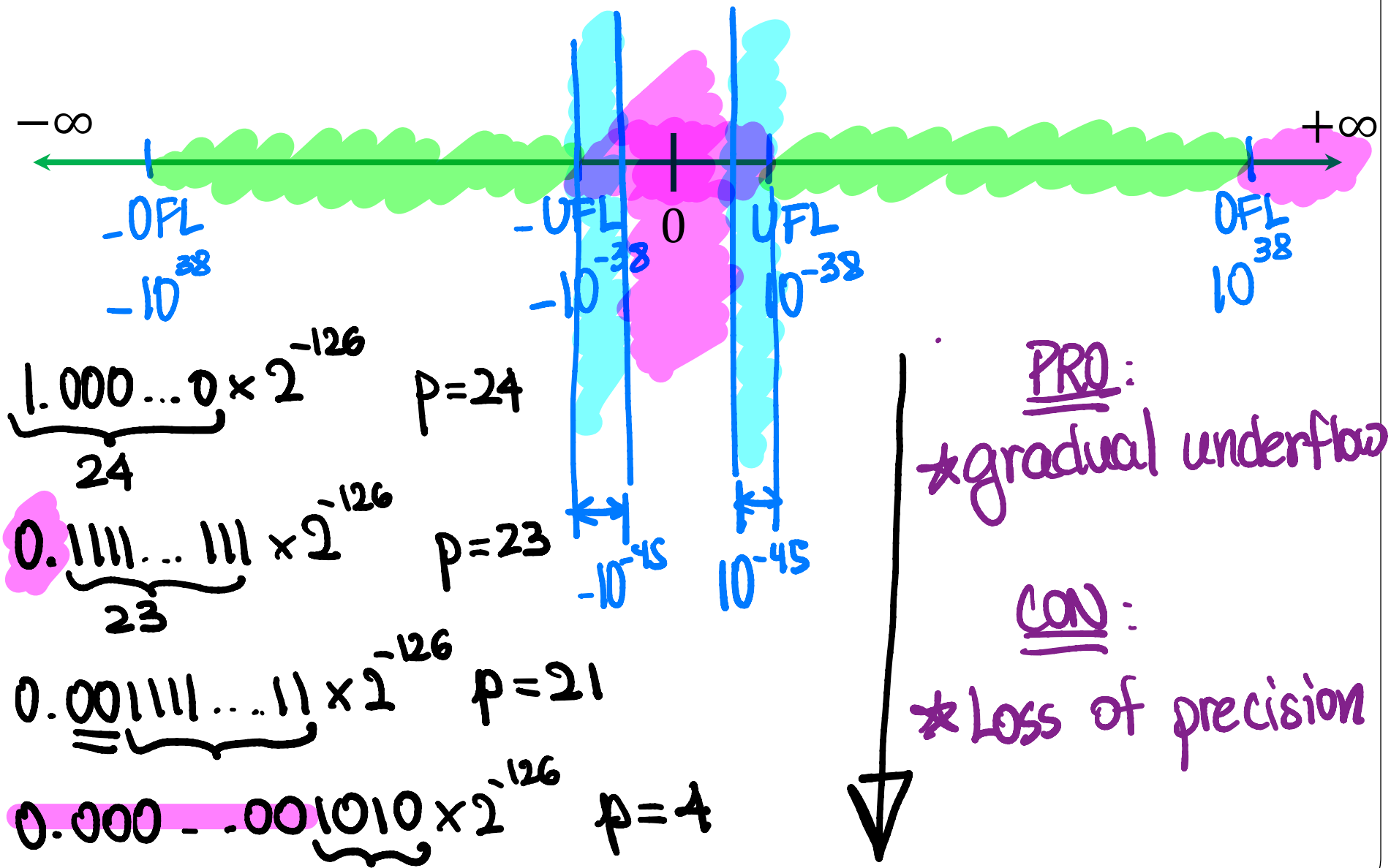
$c = (00000000000)_2 = 0$

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

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)



$1.000\ldots0 \times 2^{-126}$ (24) $p=24$

$0.1111\ldots111 \times 2^{-126}$ (23) $p=23$

$0.001111\ldots11 \times 2^{-126}$ $p=21$

$0.000\ldots.001010 \times 2^{-126}$ $p=4$

PRO:

* gradual underflow

CON:

* Loss of precision

Subnormal (or denormalized) numbers

Another special case:

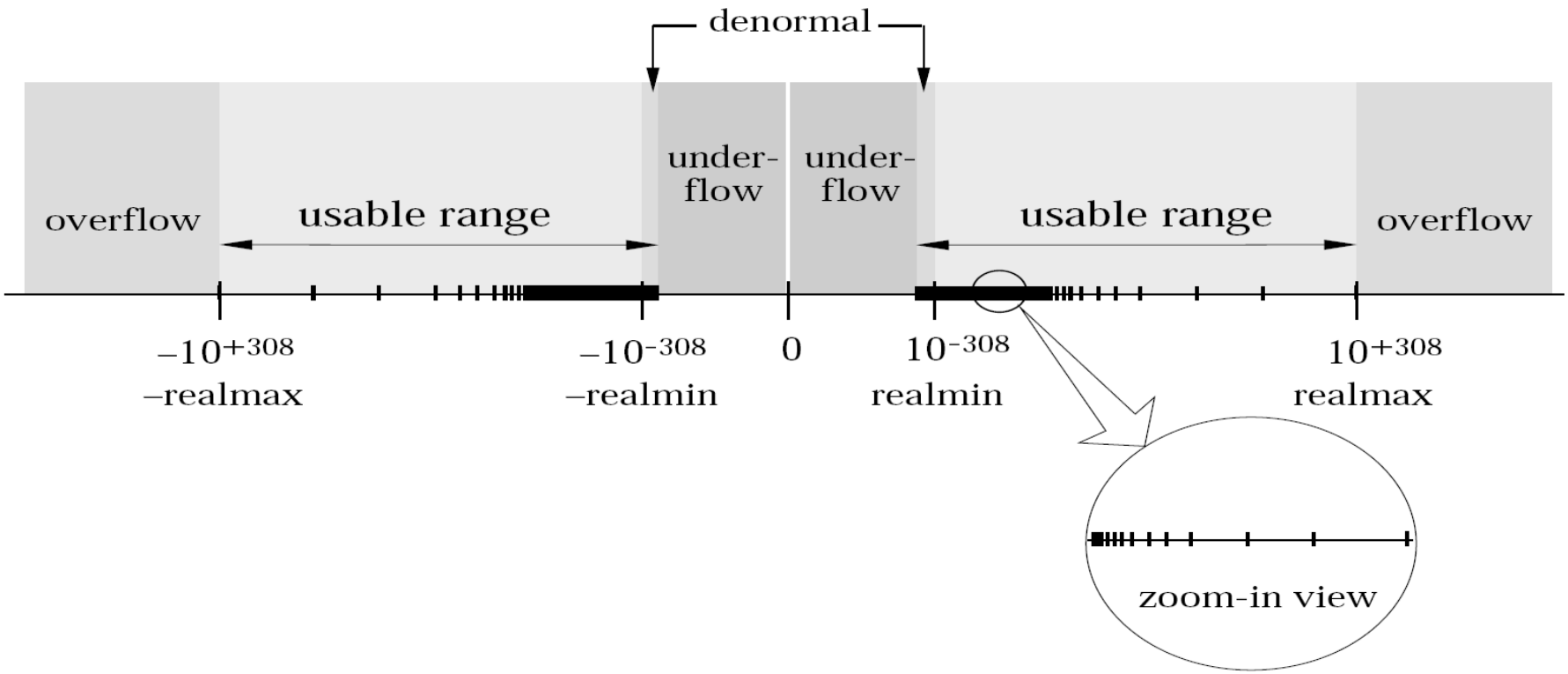
$$x = \boxed{s \mid c = 000 \ldots 000 \mid f}$$

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

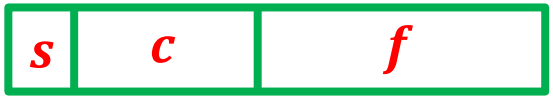
Note that this is a special case, and the exponent m is **not** evaluated as $m = c - shift = -shift$.
Instead, the exponent is set to the lower bound, $m = \mathbf{L}$

- 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

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

$m = c - 127$

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