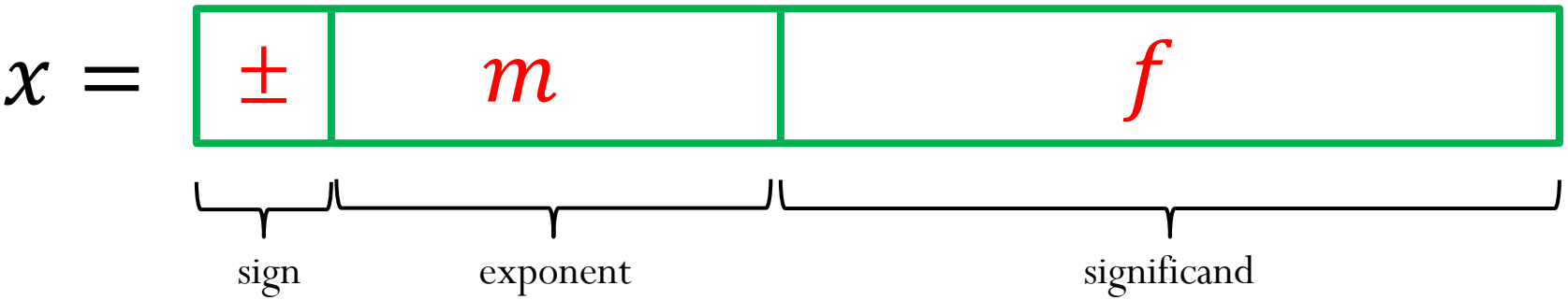
Machine numbers: how floating point numbers are stored?

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:

$x =$ 

Precisions:

IEEE-754 Single precision (32 bits):

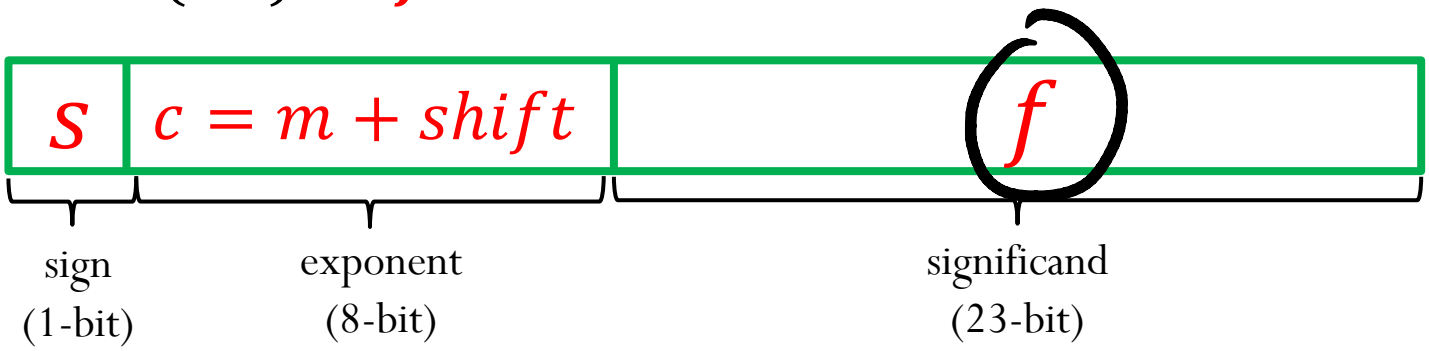
$x =$ 

IEEE-754 Double precision (64 bits):

$x =$ 

IEEE-754 Single Precision (32-bit)

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



000...0 ≤ C ≤ 111...1

0 255

not use 0 and 255

$m \in [L, U]$

$m \in [-126, 127]$

$1 \le C \le 254$

$1 \le m + shift \le 254$

shift = 127 (choice) → $-126 \le m \le 127$

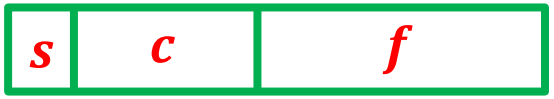
IEEE-754 Single Precision (32-bit)

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

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)

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

$c = m + 127$

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

- **Smallest positive normalized FP number:**

- **Largest positive normalized FP number:**

Special Values:

$0 < c < 255$

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

s	c	f

1) Zero:

$x =$

	c: 000...00	f: 00 00 ... 000

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

$x =$

	111...111	0000 ... 0000

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

$x =$

	111...111	anything but zeros

hold

Subnormal

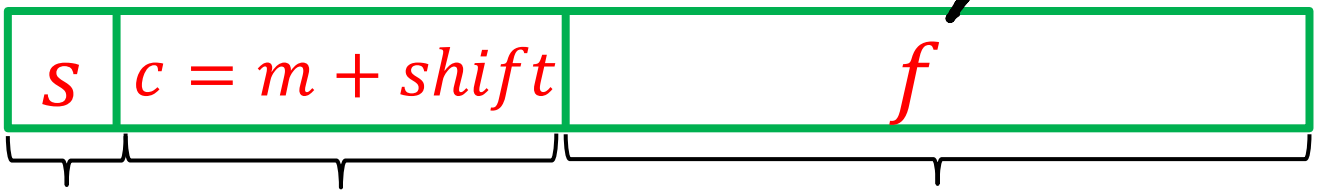
	c: 000 0...00	anything but zeros

IEEE-754 Double Precision (64-bit)

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

52

$p = 53$



sign (1-bit) | exponent (11-bit) | significand (52-bit)

$000\ldots00 \leq c \leq 111\ldots11$

11 11

$0 \leq c \leq 2047$

save 2047, 0 → special cases

$1 \leq c \leq 2046$

$1 \leq m + shift \leq 2046 \Longrightarrow -1022 \leq m \leq 1023$

shift = 1023 (choice!)

$m \in [-1022, 1023]$

IEEE-754 Double Precision (64-bit)

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

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

$\epsilon_m = 0.000\ldots01 \times 2^{0} = 2^{-n} = 2^{-52} \approx 10^{-16}$

(n bits; 2^{-n})

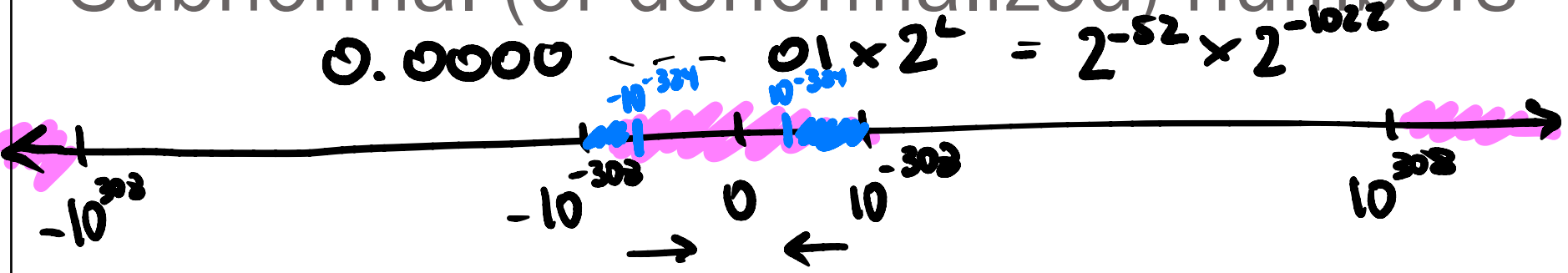
- **Smallest positive normalized FP number:**

$UFL = 1.000\ldots00 \times 2^{L} = 2^{L} = 2^{-1022} \approx 10^{-308}$

- **Largest positive normalized FP number:**

$OFL = 1.111\ldots11 \times 2^{U} = 2^{U+1}(1-2^{-p}) = 2^{1024}(1-2^{-53}) \approx 10^{308}$

Subnormal (or denormalized) numbers



$$x = 1.0000\ldots0 \times 2^L$$

$$x = 0.b_1 b_2 \ldots b_n \times 2^L$$

$$x = 1.\overbrace{000101\ldots01}^{52} \times 2^L \longrightarrow p = 53$$

$$x = 0.111111\ldots11 \times 2^L \longrightarrow p = 52$$

$$x = 0.1100\ldots100 \times 2^L \longrightarrow p = 52$$

$$x = 0.0001001\ldots01 \times 2^L \longrightarrow p = 49$$

Subnormal (or denormalized) numbers

IEEE-754 Single precision (32 bits):

$c = (00000000)_2 = 0$

smallest # $0.000\ldots01 \times 2^{-126} = 2^{-23} \times 2^{-126} \approx 10^{-45}$

IEEE-754 Double precision (64 bits):

$c = (00000000000)_2 = 0$

smallest # $0.000\ldots01 \times 2^{-1022} = 2^{-52} \times 2^{-1022} \approx 10^{-324}$

Subnormal (or denormalized) numbers

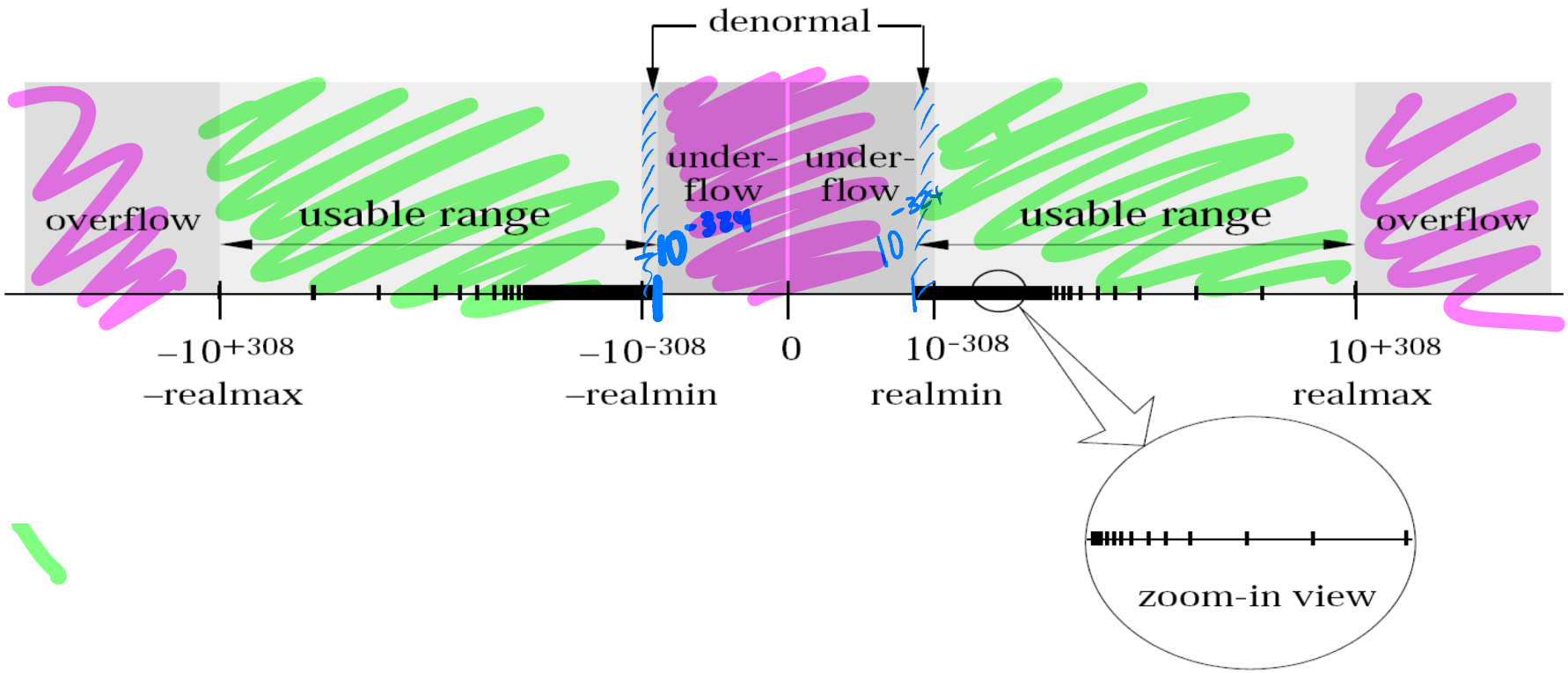
- Noticeable gap around zero, present in any floating system, due to normalization
- Relax the requirement of normalization, and allow the leading digit to be zero, only when the exponent is at its minimum ($m = L$)
- Computations with subnormal numbers are often slow.

Representation in memory (another special case):

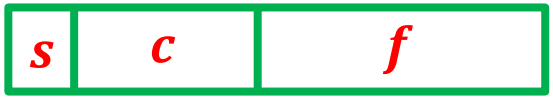
Numerical value:

$$x = (\pm)\, 0.b_1 b_2 b_3 \ldots \times 2^{L}$$

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

Iclicker question

A number system can be represented as $x = \pm 1.b_1 b_2 b_3 \times 2^m$ for $m \in [-5,5]$ and $b_i \in \{0,1\}$.

1) What is the smallest positive normalized FP number:

a) 0.0625 b) 0.09375 c) 0.03125 d) 0.046875 e) 0.125

2) What is the largest positive normalized FP number:

a) 28 b) 60 c) 56 d) 32

3) How many additional numbers (positive and negative) can be represented when using subnormal representation?

a) 7 b) 14 c) 3 d) 6 e) 16

4) What is the smallest positive subnormal number?

a) 0.00390625 b) 0.00195313 c) 0.03125 d) 0.0136719

5) Determine machine epsilon

a) 0.0625 b) 0.00390625 c) 0.0117188 d) 0.125

A number system can be represented as $x = \pm 1.b_1 b_2 b_3 b_4 \times 2^m$ for $m \in [-6,6]$ and $b_i \in \{0,1\}$.

$n = 4 \rightarrow p = 5$

1) Let's say you want to represent the decimal number 19.625 using the binary number system above. Can you represent this number exactly?

1.0000×2^p ← last integer

2) What is the range of integer numbers that you can represent exactly using this binary system?

$1.0000 \times 2^{\boxed{5}} = 32$

$1.0001 \times 2^5 = 34$

2^5

$(1)_{10} = (1)_2 = 1.0000 \times 2^0$

$(2)_{10} = (10)_2 = 1.0000 \times 2^1$

$(3)_{10} = (11)_2 = 1.1000 \times 2^1$

$(11111)_2 = 1.1111 \times 2^4 = (31)_{10}$