Floating point representation

$$[\quad]_{n\times n} \longrightarrow \boxed{\begin{array}{c}\text{Method}\\ O(n^\alpha)\end{array}} \longrightarrow \text{output quantity}$$

$$n \longrightarrow t$$

$$10 \longrightarrow 2s$$

$$10^3 \longrightarrow 40s$$

$$t = c\,n^\alpha$$

$$\boxed{\begin{array}{l} 2 = c\,10^\alpha \\ 40 = c(10)^{3\alpha} \end{array}}$$

(Unsigned) Fixed-point representation

The numbers are stored with a fixed number of bits for the integer part and a fixed number of bits for the fractional part.

Suppose we have 8 bits to store a real number, where 5 bits store the integer part and 3 bits store the fractional part:

$$\left(\underset{2^4}{1}\ \underset{2^3}{0}\ \underset{2^2}{1}\ \underset{2^1}{1}\ \underset{2^0}{1}.\underset{2^{-1}}{0}\ \underset{2^{-2}}{1}\ \underset{2^{-3}}{1}\right)_2$$

Smallest number: $(00000.001)_2 = (0.125)_{10}$

Largest number: $(11111.111)_2 = (31.875)_{10}$

(Unsigned) Fixed-point representation

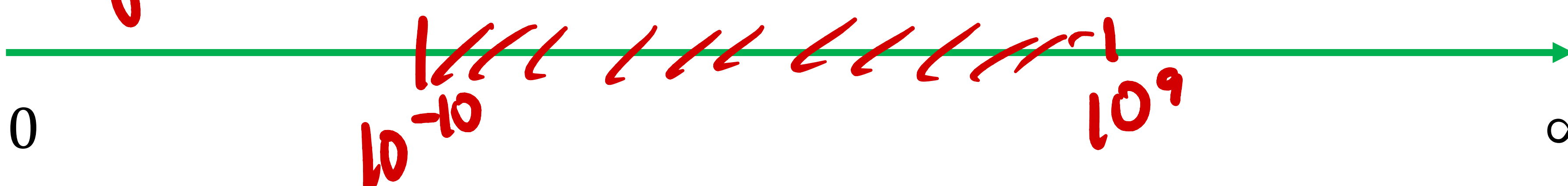
Suppose we have 64 bits to store a real number, where 32 bits store the integer part and 32 bits store the fractional part:

$$(a_{31} \ldots a_2 a_1 a_0 . b_1 b_2 b_3 \ldots b_{32})_2 = \sum_{k=0}^{31} a_k \, 2^k + \sum_{k=1}^{32} b_k \, 2^{-k}$$

$$= a_{31} \times 2^{31} + a_{30} \times 2^{30} + \cdots + a_0 \times 2^0 + b_1 \times 2^{-1} + b_2 \times 2^{2} + \cdots + b_{32} \times 2^{-32}$$

smallest : $00 \ldots 00.\underbrace{0}_{2^{-1}}\underbrace{0}_{2^{-2}} \ldots 01 \sim 2^{-32} \approx 10^{-10}$

largest : $111 \ldots 11.111 \ldots 11 \approx 10^9$



(Unsigned) Fixed-point representation

Range: difference between the largest and smallest numbers possible.

More bits for the integer part $\longrightarrow$ increase range

Precision: smallest possible difference between any two numbers

More bits for the fractional part $\longrightarrow$ increase precision

$(a_2 a_1 a_0 . b_1 b_2 b_3)_2$ OR $(a_1 a_0 . b_1 b_2 b_3 b_4)_2$

Wherever we put the binary point, there is a trade-off between the amount of range and precision. **It can be hard to decide how much you need of each!**

Scientific Notation

In **scientific notation**, a number can be expressed in the form

$$x = \pm r \times 10^m$$

where r is a coefficient in the range $1 \leq r < 10$ and m is the exponent.

$1165.7 =$

$0.0004728 =$

Floating-point numbers

A floating-point number can represent numbers of different order of magnitude (very large and very small) with the same number of fixed bits.

In general, in the binary system, a floating number can be expressed as

$$x = \pm\, q \times 2^m$$

q is the significand, normally a fractional value in the range $[1.0, 2.0)$

m is the exponent

Floating-point numbers

Numerical Form:

$$x = \pm q \times 2^m = \pm \overbrace{b_0 . \underbrace{b_1 b_2 b_3 \ldots b_n}_{\text{Fractional part of significand } (n \text{ bits})}}^{\text{significand}} \times 2^m$$

leading bit

$b_i \in \{0, 1\}$

$m \in [L, U]$

Precision : $p = n + 1$

Normalized floating-point numbers

Normalized floating point numbers are expressed as

$$x = \pm 1.b_1 b_2 b_3 \ldots b_n \times 2^m = \pm 1.f \times 2^m$$

where f is the fractional part of the significand, m is the exponent and $b_i \in \{0,1\}$.

store 5 bits

$b_0 . b_1 b_2 b_3 b_4 \rightarrow p = 5$ bits

$1 . b_1 b_2 b_3 b_4 b_5 \rightarrow p = 6$

hidden bit representation → "gain" 1 bit of precision

Converting floating points

Convert $(39.6875)_{10} = (100111.1011)_2$ into floating point representation

Iclicker question

meet.ps/cs357

Determine the normalized floating point representation $1.f \times 2^{m}$ of the decimal number $x = 47.125$ (f in binary representation and m in decimal)

A) $(1.01110001)_2 \times 2^{5}$
B) $(1.01110001)_2 \times 2^{4}$
C) $(1.01111001)_2 \times 2^{5}$
D) $(1.01111001)_2 \times 2^{4}$

$(47.125)_{10} = (101111.011)_2$

1.01111011×2^{5}

Normalized floating-point numbers

$$x = \pm q \times 2^m = \pm 1.b_1 b_2 b_3 \ldots b_n \times 2^m = \pm \boxed{1}\, f \times 2^m$$

- **Exponent range**: $m \in [L, U]$
- **Precision**: $p = n+1$ n: # bits in f
- **Smallest positive normalized FP number:**

$$1.\underbrace{000 \ldots 00}_{n} \times 2^L = 2^L$$

- **Largest positive normalized FP number:**

$$1.\underbrace{111 \ldots 11}_{n} \times 2^U = 2^{U+1}(1-2^{-p})$$

Floating-point numbers: Simple example

A "toy" number system can be represented as $x = \pm 1.b_1 b_2 \times 2^m$ for $m \in [-4,4]$ and $b_i \in \{0,1\}$.

$1.00 \times 2^0 = 1$

$1.01 \times 2^0 = 1.25$

$1.10 \times 2^0 = 1.5$

$1.11 \times 2^0 = 1.75$

$m=1$

1.00×2^1

1.01×2^1

1.10×2^1

1.11×2^1

$m=2$

$m=3$

$m=4$

$m=-1$

$m=-2$

$m=-3$

$m=-4$

Floating-point numbers: Simple example

A "toy" number system can be represented as $x = \pm 1.b_1 b_2 \times 2^m$ for $m \in [-4,4]$ and $b_i \in \{0,1\}$.

$2^{U+1}(1-2^{-p}) = 2^5(1-2^{-3})$

$(1.00)_2 \times 2^0 = 1$
$(1.01)_2 \times 2^0 = 1.25$
$(1.10)_2 \times 2^0 = 1.5$
$(1.11)_2 \times 2^0 = 1.75$

$(1.00)_2 \times 2^1 = 2$
$(1.01)_2 \times 2^1 = 2.5$
$(1.10)_2 \times 2^1 = 3.0$
$(1.11)_2 \times 2^1 = 3.5$

$(1.00)_2 \times 2^2 = 4.0$
$(1.01)_2 \times 2^2 = 5.0$
$(1.10)_2 \times 2^2 = 6.0$
$(1.11)_2 \times 2^2 = 7.0$

$(1.00)_2 \times 2^3 = 8.0$
$(1.01)_2 \times 2^3 = 10.0$
$(1.10)_2 \times 2^3 = 12.0$
$(1.11)_2 \times 2^3 = 14.0$

$(1.00)_2 \times 2^4 = 16.0$
$(1.01)_2 \times 2^4 = 20.0$
$(1.10)_2 \times 2^4 = 24.0$
$\boxed{(1.11)_2 \times 2^4 = 28.0}$

$(1.00)_2 \times 2^{-1} = 0.5$
$(1.01)_2 \times 2^{-1} = 0.625$
$(1.10)_2 \times 2^{-1} = 0.75$
$(1.11)_2 \times 2^{-1} = 0.875$

2^L

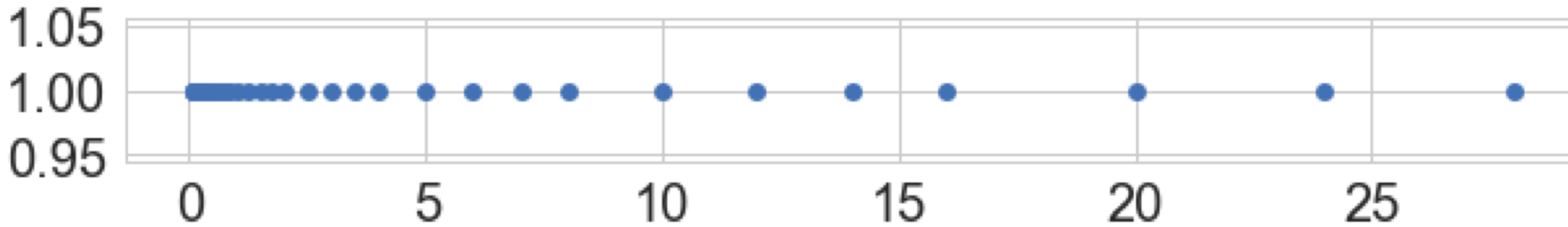
$(1.00)_2 \times 2^{-2} = 0.25$
$(1.01)_2 \times 2^{-2} = 0.3125$
$(1.10)_2 \times 2^{-2} = 0.375$
$(1.11)_2 \times 2^{-2} = 0.4375$

$(1.00)_2 \times 2^{-3} = 0.125$
$(1.01)_2 \times 2^{-3} = 0.15625$
$(1.10)_2 \times 2^{-3} = 0.1875$
$(1.11)_2 \times 2^{-3} = 0.21875$

$\boxed{(1.00)_2 \times 2^{-4} = 0.0625}$
$(1.01)_2 \times 2^{-4} = 0.078125$
$(1.10)_2 \times 2^{-4} = 0.09375$
$(1.11)_2 \times 2^{-4} = 0.109375$

Same steps are performed to obtain the negative numbers. For simplicity, we will show only the positive numbers in this example.

$$x = \pm 1.b_1 b_2 \times 2^m \text{ for } m \in [-4,4] \text{ and } b_i \in \{0,1\}$$



- Smallest normalized positive number:

$$2^L = 2^{-4} = 0.0625$$

- Largest normalized positive number:

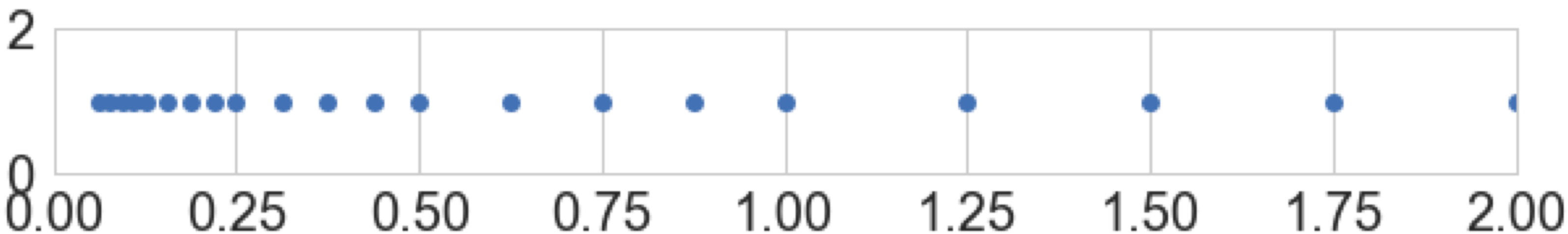
$$2^{U+1}(1-2^{-p}) = 28$$

Machine epsilon

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

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

$$x = \pm 1.b_1 b_2 \times 2^m \quad \text{for } m \in [-4,4] \text{ and } b_i \in \{0,1\}$$



$x \Rightarrow 1.\underbrace{0000\ldots00}_{n} \times 2^0$

$1.000\ldots01 \times 2^0$

$0.000\ldots0\boxed{1} \times 2^{0} = 2^{-n}$

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?

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

$$x = \underbrace{\boxed{\pm}}_{\text{sign}}\underbrace{\boxed{\quad m \quad}}_{\text{exponent}}\underbrace{\boxed{\qquad f \qquad}}_{\text{significand}}$$

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:

signed

$$x = \pm 1.f \times 2^{m} \qquad m \in [L, U]$$

Representation in memory:

$$x = \boxed{S} \boxed{C = m + \text{shift}} \boxed{\quad f \quad}$$

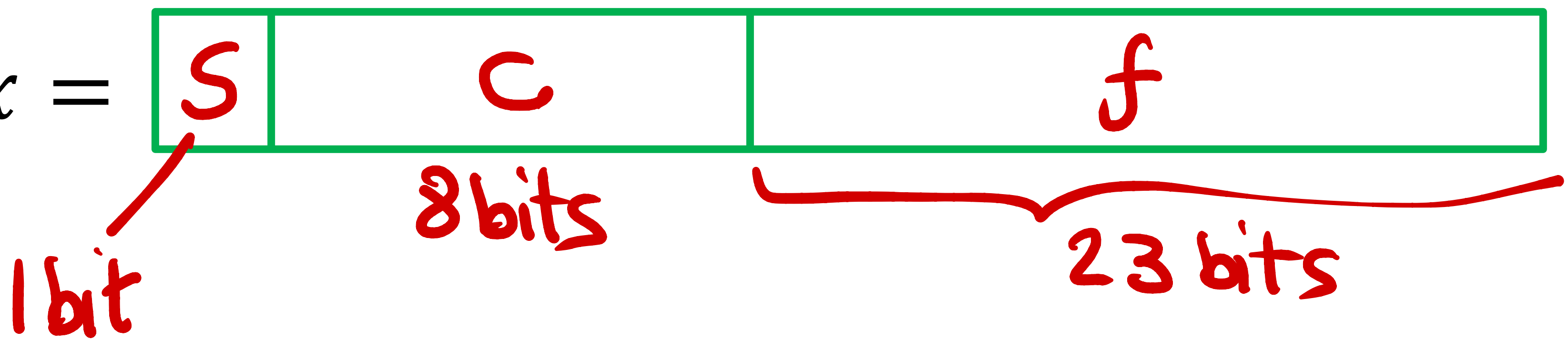
unsigned

signed

Precisions:

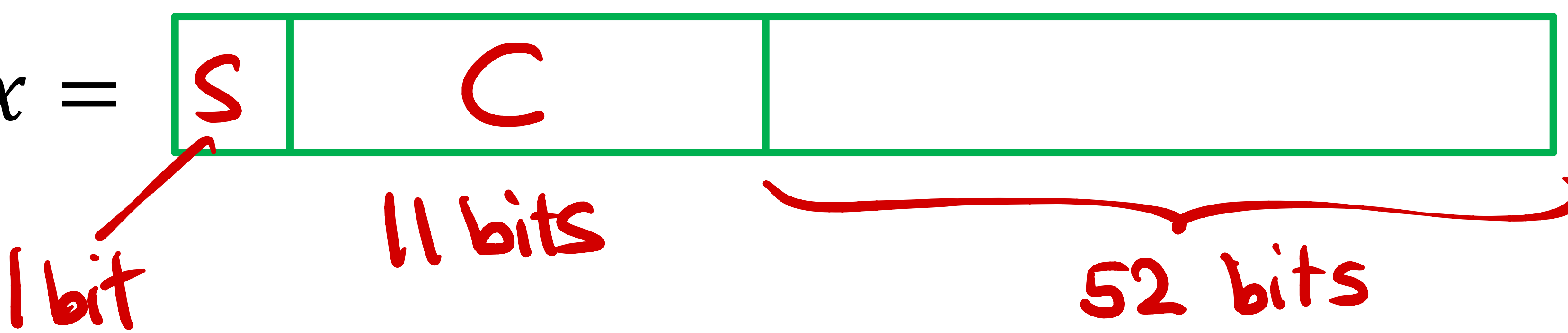
IEEE-754 Single precision (32 bits):

$x =$



IEEE-754 Double precision (64 bits):

$x =$



IEEE-754 Single Precision (32-bit)

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

L = −126
U = 127
p = 24

s	$c = m + shift$	f
sign (1-bit)	exponent (8-bit)	significand (23-bit)

$c = (00000000)_2 = (0)_{10}$ $\qquad 0 \leq c \leq 255$

$c = (11111111)_2 = 255$ $\qquad$ set aside $c = 0$, $c = 255$

$$\boxed{\begin{gathered} 1 \leq c \leq 254 \\ (00000001) \leq c \leq (11111110) \end{gathered}}$$

$1 \leq m + shift \leq 254 \longrightarrow$ choice of shift as 127

$$\boxed{-126 \leq m \leq 127}$$

m

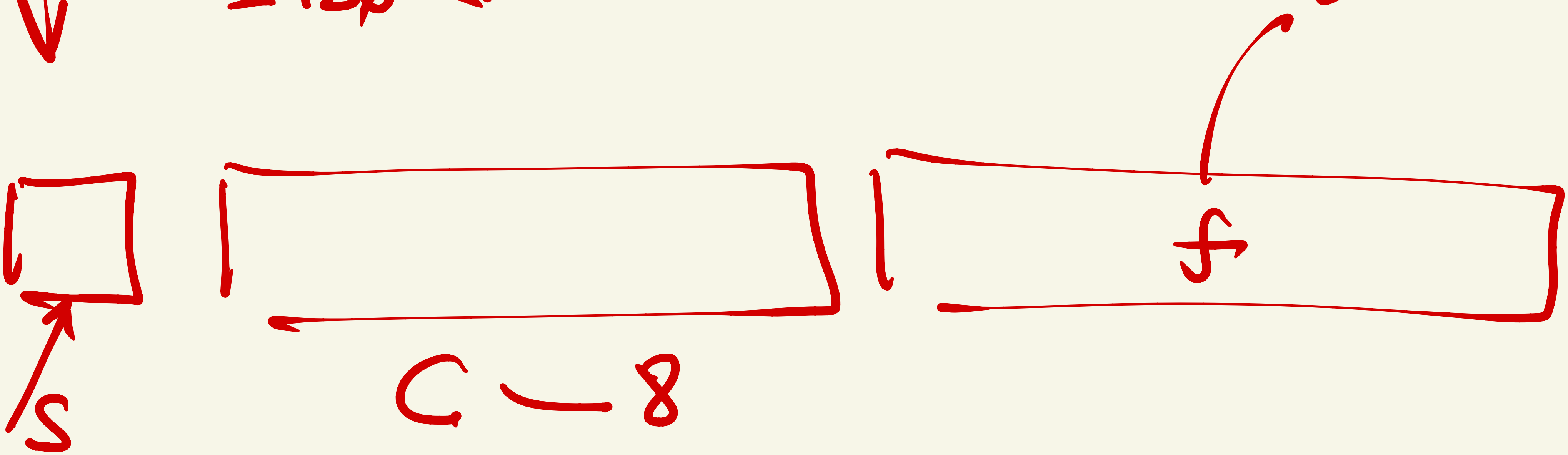
$$1 \leq c \leq 254$$

$$1 \leq m + \text{shift} \leq 254$$

$$\boxed{\text{shift} = 127}$$ choice!

$$1 - 127 \leq m \leq 254 - 127$$

$$-126 \leq m \leq 127$$



$$c = (01110010)$$

$$c = m + shift$$

$$m = c - shift$$

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)

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

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

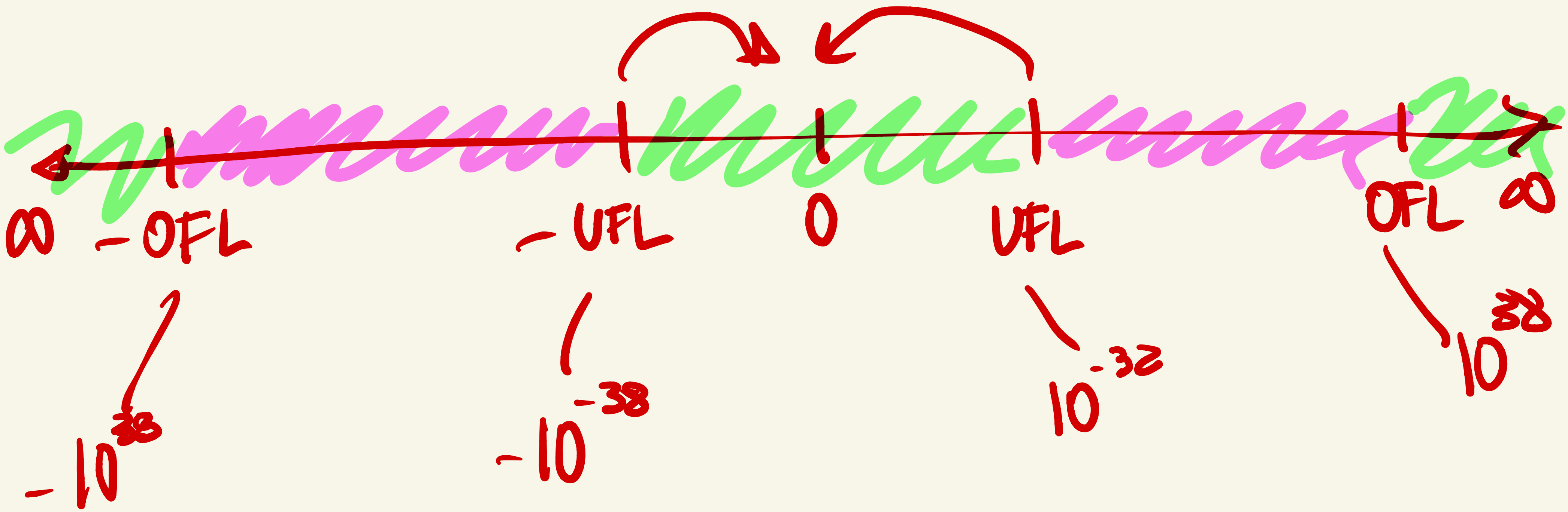
$$\epsilon_m = 2^{-n} = 2^{-23} \approx 1.2 \times 10^{-7}$$

- **Smallest positive normalized FP number:**

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

- **Largest positive normalized FP number:**

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



∞
-OFL
-UFL
0
UFL
OFL
∞
-10^{38}
-10^{-38}
10^{-32}
10^{38}