Video 1: Intro to Floating point

(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:

$$(1\,0\,1\,1\,1.0\,1\,1)_2$$

$$2^4\quad 2^3\quad 2^2\quad 2^1\quad 2^0\quad 2^{-1}\quad 2^{-2}\quad 2^{-3}$$

$$2^4 + \cancel{0} + 2^2 + 2^1 + 2^0 + 0 + 2^{-2} + 2^{-3} = (23.375)_{10}$$

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

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

(Unsigned) Fixed-point representation

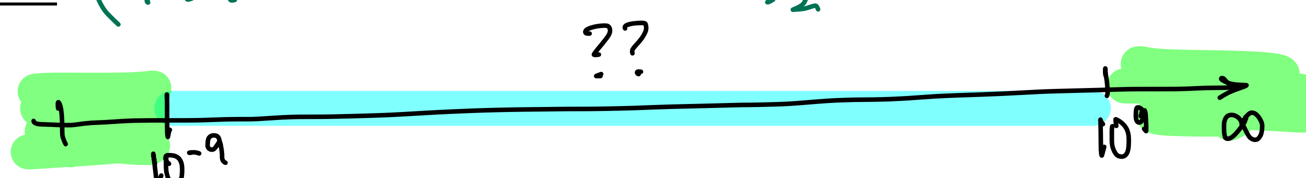
Suppose we have 64 bits to store a real number, where 32 bits store the integer part and 32 bits store the fractional part:

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

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

Smallest number: $\underbrace{0000 \dots 0}_{32} . \underbrace{000 \dots 0\overset{2^{-32}}{1}}_{32} = 2^{-32} \cong 10^{-9}$

Largest number: $(111 \dots 1 . 11 \dots 1)_2 \cong 10^9$



Fixed-point representation

How can we decide where to locate the binary point?

More bits on the integer part?

More bits on the fractional part?

5 bits $\rightarrow a_0.b_1b_2b_3b_4$

1

$(0.0001)_2 = 0.0625$

$(0.0010)_2 = 0.125$

$\vdots$

$1.1110 = 1.875$

$(1.1111) = 1.9375$

0.0625

0.0625

5 bits $\rightarrow a_2a_1a_0.b_1b_2$

2

$000.01 = 0.25$

$000.10 = 0.5$

$\vdots$

$(111.10) = 7.5$

$(111.11)_2 = (7.75)_{10}$

0.25

0.25

(Unsigned) Fixed-point representation

Range: difference between the largest and smallest numbers possible.

More bits for the integer part ⟶ increase range

Precision: smallest possible difference between any two numbers

More bits for the fractional part ⟶ increase precision

$$(a_2 a_1 a_0 . b_1 b_2 b_3)_2 \qquad \text{OR} \qquad (a_1 a_0 . b_1 b_2 b_3 b_4)_2$$

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!**

Fix: Let the binary point "float"

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.

$1165.7 = \underline{1.1657} \times 10^{3}$

$0.0004728 = \underline{4.728} \times 10^{-4}$

Note how the decimal point "floats"!

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 digits.

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 $\longrightarrow$ $m \in [L, U]$ $m \in [-4, 4]$

Floating-point numbers

leading bit

fractional

Numerical Form:

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

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

Exponent range: $m \in [L, U]$

Precision: $\boxed{p = n + 1}$

Video 2: Normalized floating point representation

Converting floating points

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

1.001111011×2^5

0.1001111011×2^6

Normalized floating-point numbers

leading bit

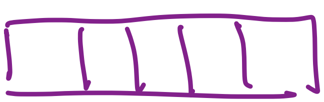
$m \in [L, U]$

Normalized floating point numbers are expressed as

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

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

5 bits



$a_0 . \underbrace{b_1 b_2 b_3 b_4} \longrightarrow p = 5$

$1 . \underbrace{b_1 b_2 b_3 b_4} b_5 \longrightarrow \boxed{p = 6}$

"gain" 1 bit of precision

"hidden" bit

Normalized floating-point numbers

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

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

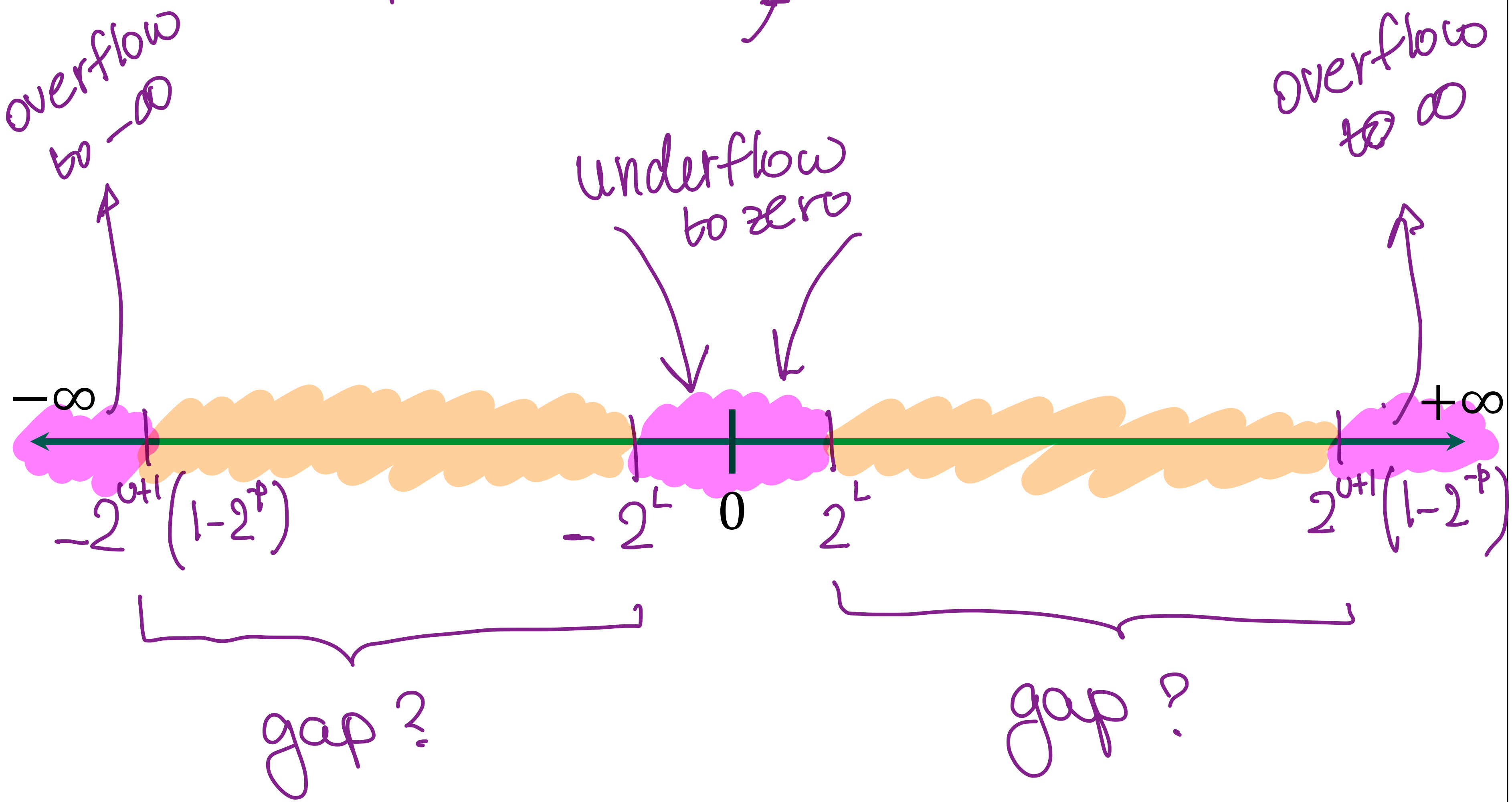
$$1.\underbrace{00.\,\ldots\,00}_{n} \times 2^L = \boxed{2^L} \rightarrow \text{exponent}$$

- **Largest positive normalized FP number:**

$$1.\underbrace{111\ldots1}_{n} \times 2^U = \boxed{2^{U+1}(1-2^{-p})} \rightarrow \text{exponent range + precision}$$

Normalized floating point number scale

$p = n+1$ $\quad 1.f \times 2^m$ $\quad m \in [L, U]$



Floating-point numbers: Simple example

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

$m = 0$

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

$m = 1$

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

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

0.25

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

0.5

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

1.0

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

2.0

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

4.0

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

0.125

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

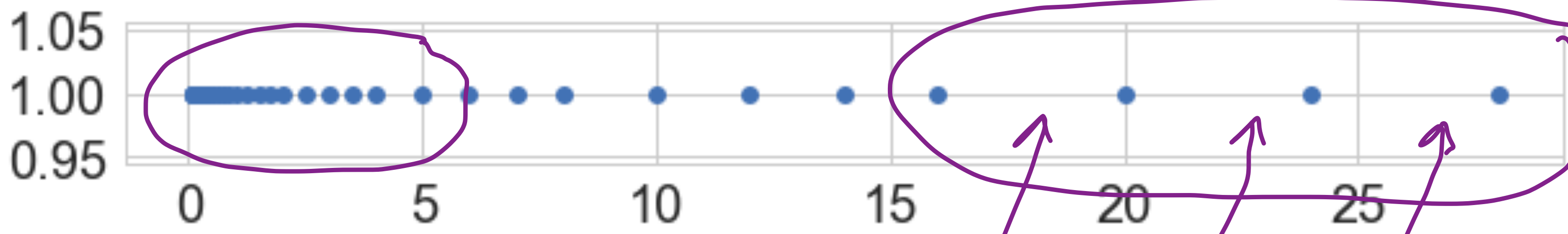
$(1.00)_2 \times 2^{-3} = 0.125$
$(1.01)_2 \times 2^{-3} = 0.15625$
$(1.10)_2 \times 2^{-3} = 0.1875$
$(1.11)_2 \times 2^{-3} = 0.21875$

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

0.015625

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

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



- Smallest normalized positive number:

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

- Largest normalized positive number:

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

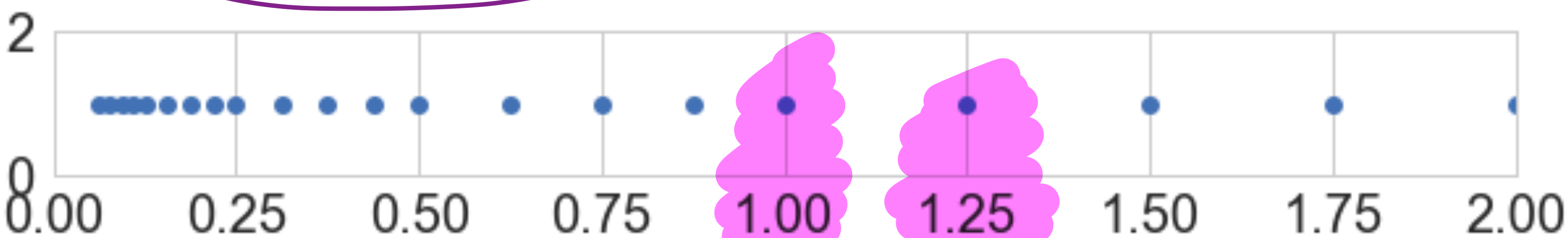
$U = 4$

$p = n+1 = 3$

Machine epsilon

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

$$x = \pm 1.b_1b_2 \times 2^m \text{ for } m \in [-4,4] \text{ and } b_i \in \{0,1\}$$



$\epsilon_m = 1.25 - 1. = 0.25$

in general: $x = \pm 1.f$ $(\underline{n})$

$(1)_{10} = 1.\underbrace{0000 \ldots 00} \times 2^0$

$\ominus\ 1.000 \ldots 001 \times 2^0$

$0.000 \ldots 001 \times 2^0 = 2^{-n} \times 2^0 = 2^{-n}$

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

Range of integer numbers

Suppose you have this following normalized floating point representation:

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

What is the range of integer numbers that you can represent exactly?

$1.00 \times 2^0 = 1$

$(10)_2 = 1.00 \times 2^1 = (2)_{10}$

$(11)_2 = 1.10 \times 2^1 = (3)_{10}$

$(100)_2 = 1.00 \times 2^2 = (4)_{10}$

$(101)_2 = 1.01 \times 2^2 = (5)_{10}$

$(110)_2 = 1.10 \times 2^2 = (6)_{10}$

$(111)_2 = 1.11 \times 2^2 = 7$

p

$(1000)_2 = 1.00 \times 2^3 = (8)_{10}$

$(1001)_2 = \underbrace{\qquad}$ $(9)_{10}$

$= 1.01 \times 2^3 = 10$

$\boxed{2^p}$