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:

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

<u>Smallest number:</u>

<u>Largest number:</u>

(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:

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

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

<u>Smallest number:</u>

<u>Largest number:</u>

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?

(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

$(a_2 a_1 a_0 . b_1 b_2 b_3)_2$ 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!**

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.

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

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

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

Floating-point numbers

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: $\mathrm{p} = n + 1$

Normalized floating-point numbers

Normalized floating point numbers are expressed as

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

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

Converting floating points

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

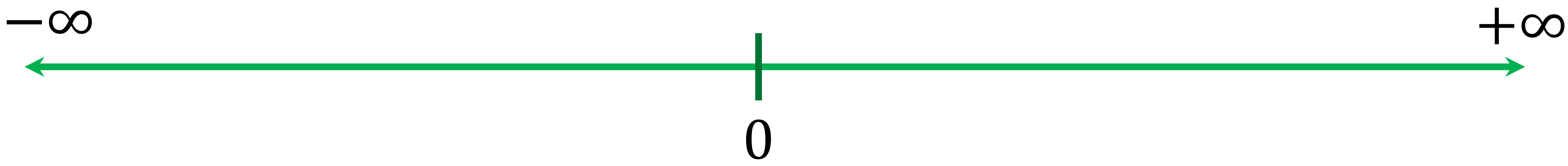
Video 2: Normalized floating point representation

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**:
- **Precision**:
- **Smallest positive normalized FP number:**
- **Largest positive normalized FP number:**

Normalized floating point number scale



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

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

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

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

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

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

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

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

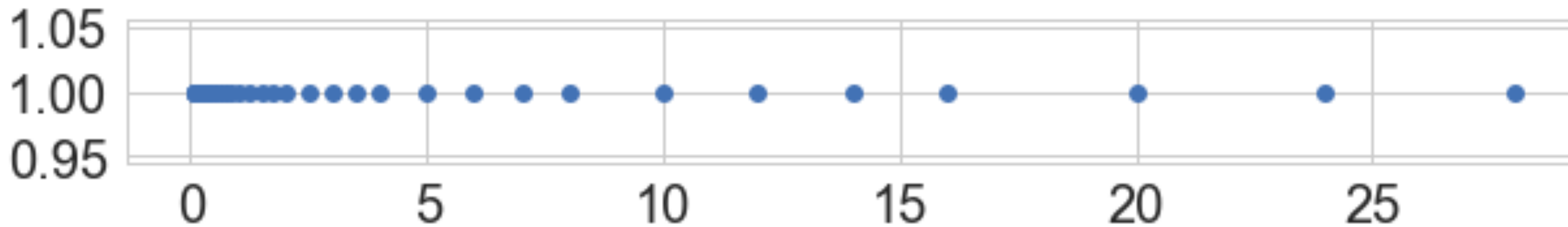
$(1.00)_2 \times 2^{-2} = 0.25$
$(1.01)_2 \times 2^{-2} = 0.3125$
$(1.10)_2 \times 2^{-2} = 0.375$
$(1.11)_2 \times 2^{-2} = 0.4375$

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

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

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:

- Largest normalized positive number:

Machine epsilon

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

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

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?