Errors and Cancellation

My daughter, Julia, is planning her b-day party, and she decided she wants to have a slime theme. Her slime recipe asks for 0.4lb of glue, unfortunately I do not have any scale that can measure such small weight. She told me that if she cannot measure the glue accurately, her party will be a complete disaster!

Julia asked me if we can use the same technique that I use when measuring my suitcase weight before trips: I climb on the scale holding the suitcase, and then again without it, subtract both weights, and voila, I have the weight of the suitcase! My digital scale has readings with resolution of 0.1lb (this is the increment), and according to the manual, it gives measurements with up to 1% relative error.

What do you think I should do? Follow her advice, or immediately go to Amazon and order a small kitchen scale (and make sure I avoid unpleasant surprises during the party)?

Video 1: Intro to Floating point

(Unsigned) Fixed-point representation

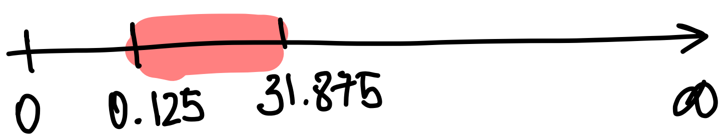
The numbers are stored with a fixed number of bits for the integer part and a fixed number of bits for the fractional part.

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

$$(\underbrace{1\ 0\ 1\ 1\ 1}_{\text{5 bits}}.\underbrace{0\ 1\ 1}_{\text{3 bits}})_2$$

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

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



<u>Smallest number:</u> $(00000.001)_2 = 2^{-3} = (0.125)_{10}$

<u>Largest number:</u> $(11111.111)_2 = (31.875)_{10}$

(Unsigned) Fixed-point representation

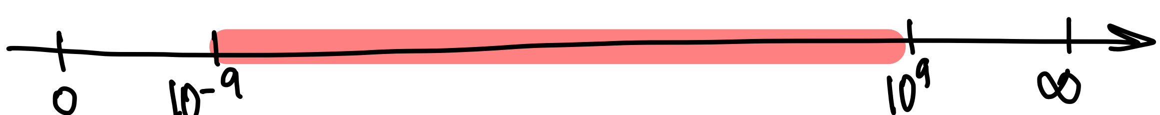
Suppose we have 64 bits to store a real number, where $\boxed{32 \text{ bits}}$ store the integer part and $\boxed{32 \text{ bits}}$ store the fractional part:

$$(\underbrace{\overset{2^{31}}{a_{31}} \dots \overset{2^{1}}{a_2}\overset{2^{0}}{a_1}a_0}_{32\text{ bits}}.\underbrace{\overset{2^{-1}}{b_1}\overset{2^{-2}}{b_2}\overset{2^{-3}}{b_3} \dots \overset{2^{-32}}{b_{32}}}_{32\text{ bits}})_2 = \sum_{k=0}^{31} a_k\, 2^k + \sum_{k=1}^{32} b_k\, 2^{-k}$$

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

Smallest number: $\overbrace{000\dots00}^{32}.\overbrace{00\dots01}^{32} = 2^{-32} \cong 10^{-9}$

Largest number: $(\overbrace{111\dots11}^{32}.\overbrace{11\dots11}^{32}) \cong 10^{9}$

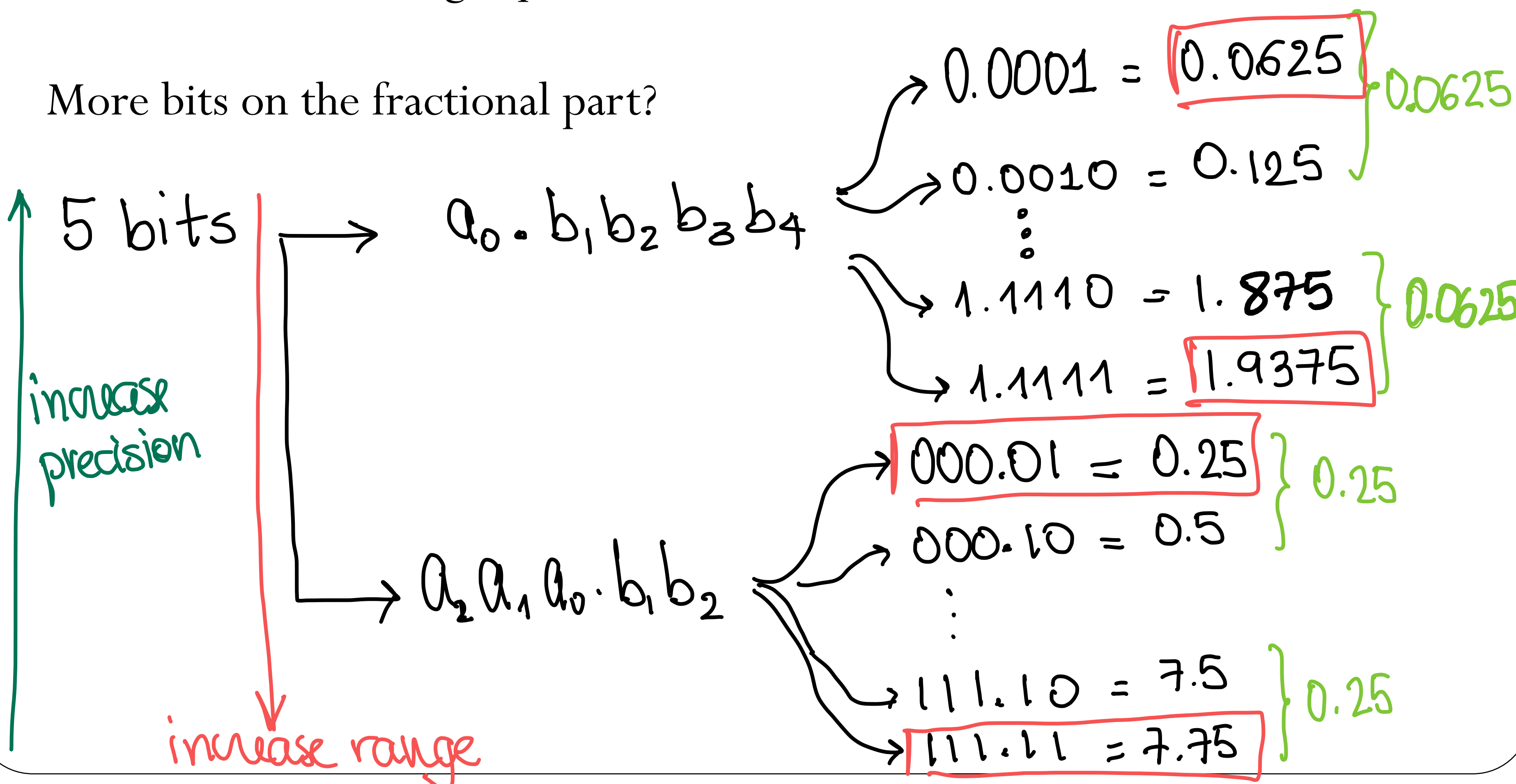


Fixed-point representation

How can we decide where to locate the binary point?

More bits on the integer part?

More bits on the fractional part?



(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!**

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

$$x = \pm \boxed{q} \times 2^{m}$$

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

m is the exponent

Floating-point numbers

Numerical Form:

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

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

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

Precision: p = $\underbrace{n + 1}$ → number of bits in significand

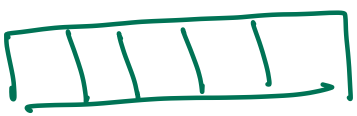
Normalized floating-point numbers

Normalized floating point numbers are expressed as

$$x = \pm \overset{\text{leading bit is always 1 (fixed!)}}{1.}\underbrace{b_1 b_2 b_3 \ldots b_n}_{n \text{ bits fractional}} \times 2^m = \pm 1.f \times 2^m$$

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

Suppose 5 bits



x

$\boxed{a_0.b_1b_2b_3b_4} \rightarrow$ precision $= 5$

x

$1.\boxed{b_1b_2b_3b_4b_5} \rightarrow$ precision $= 6$

"gain" 1 bit of precision $\rightarrow$ "hidden bit representation"

Converting floating points

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

$$(1.\boxed{001111011})_2 \times 2^{\boxed{5}}$$

fixed leading bit

Video 2: Normalized floating point representation

Normalized floating-point numbers

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

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

$$1.\underbrace{0000\cdots0}_{n} \times 2^L = \boxed{2^L}$$

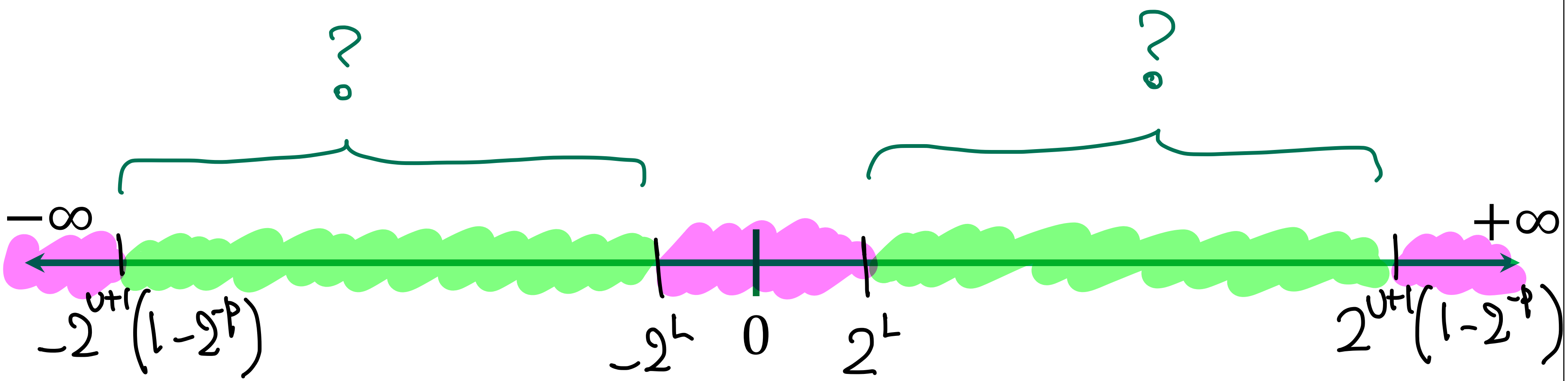
$\rightarrow$ only depends on exponent range!

- **Largest positive normalized FP number:**

$$1.\underbrace{11111\ldots1}_{n} \times 2^U = \boxed{2^{U+1}\left(1-2^{-p}\right)}$$

$\rightarrow$ depends on exponent range and precision

Normalized floating point number scale



Floating-point numbers: Simple example

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

$\boxed{m=0}$

$X = 1.00 \times 2^0 = 1$

$1.01 \times 2^0 = 1.25$

$1.10 \times 2^0 = 1.5$

$1.11 \times 2^0 = 1.75$

$\boxed{m=1}$ . . . . . $\boxed{m=4}$

$\times 2^1$

$\times 2^1$

$\vdots$

$\boxed{m=-1}$ . . . $\boxed{m=-4}$

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

$n=2 \rightarrow p=3$

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

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

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

$= 28$

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

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

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

4

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

0.125

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

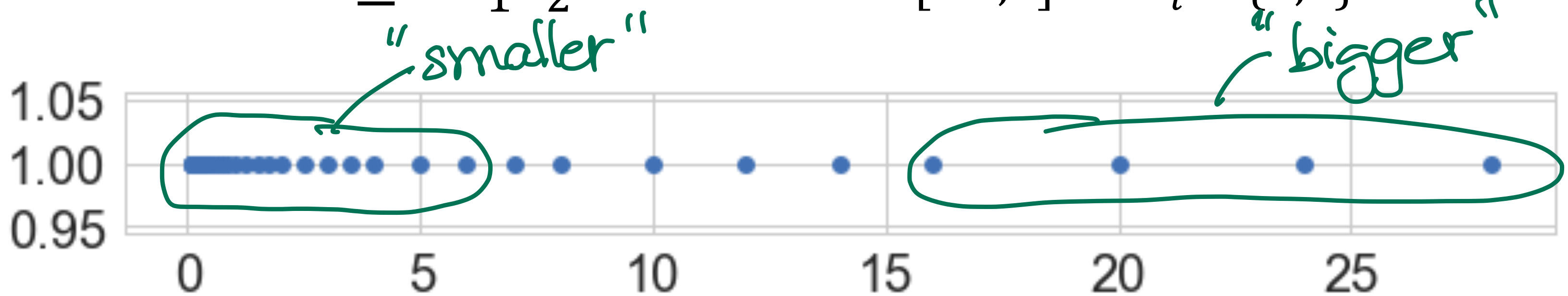
$(1.00)_2 \times 2^{-3} = 0.125$
$(1.01)_2 \times 2^{-3} = 0.15625$
$(1.10)_2 \times 2^{-3} = 0.1875$
$(1.11)_2 \times 2^{-3} = 0.21875$

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

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:

$$2^L = 0.0625$$

- Largest normalized positive number:

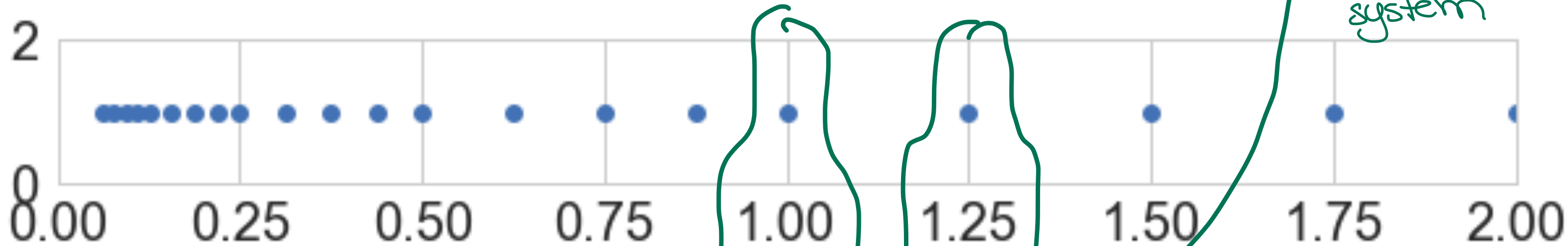
$$2^{U+1}(1-2^{-p}) = 28$$

Machine epsilon

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

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

next larger representable number in this floating point system



$\epsilon_m = 0.25$

binary :

$1.00 \times 2^0 = 1$

$1.01 \times 2^0 = 1.25$

$0.01 \times 2^0 = 2^{-2} = 0.25$

in general

$$1.\underbrace{000 \ldots 0}_{n} \times 2^0$$

$$-$$

$$1.000 \ldots 01 \times 2^0$$

$$0.\underset{2^{-1}}{0}\underset{2^{-2}}{0}0 \ldots \underset{2^{-n}}{01} \times 2^0 = 2^{-n}$$

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

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?

$1.00 \times 2^0 = 1$

$10.0 = 1.00 \times 2^1 = 2$

$11.0 = 1.10 \times 2^1 = 3$

$100 = 1.00 \times 2^2 = 4$

$101 = 1.01 \times 2^2 = 5$

$110 = 1.10 \times 2^2 = 6$

$111 = 1.\underbrace{11} \times 2^{2} = 7$

$1000 = 8 = 1.00 \times 2^{3}$

$2^3 \quad 2^2 \quad 2^1 \quad 2^0$

n+1

last integer

$\boxed{2^p}$

$9 = \times$

$10 = 1.01 \times 2^3$