13

Floating Point Arithmetic

Want: Something like the real numbers... in a computer

Have: Integers, made of bits

$$23 = \underbrace{0}\cdot \overset{32}{2^5} + \underbrace{1}\cdot \overset{16}{2^4} + \underbrace{0}\cdot \overset{8}{2^3} + \underbrace{1}\,\overset{4}{2^2} + \underbrace{1}\cdot \overset{2}{2^1} + \underbrace{1}\,\overset{1}{2^0}$$

$$(10111)_2 = (23)_{10}$$

How should we even represent fractions?

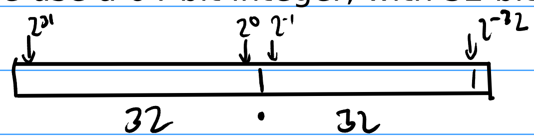
Idea: Keep going down past exponent zero

$$23.625 = 0\cdot \overset{32}{2^5} + 1\cdot \overset{16}{2^4} + 0\cdot \overset{8}{2^3} + 1\,\overset{4}{2^2} + 1\cdot \overset{2}{2^1} + 1\,\overset{1}{2^0}$$
$$+ 1\,\underset{0.5}{2^{-1}} + 0\,\underset{0.25}{2^{-2}} + 1\cdot \underset{0.125}{2^{-3}}$$

So: Could store
- a fixed number of bits with exponents >= zero
- a fixed number of bits with exponents < zero

Suppose we use a 64-bit integer, with 32 bits >= 1 and 32 bits < 1.

"fixed point" →



What is the smallest number we can represent?

$$2^{-32} \approx 10^{-10}$$

What is the biggest number we can represent?

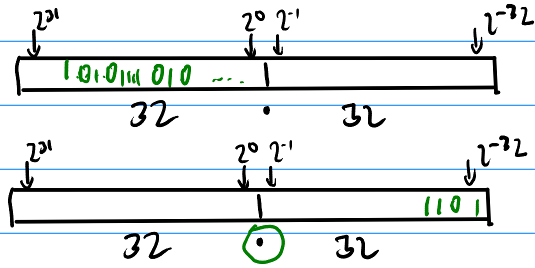
$$2^{31} + 2^{30} + 2^{29} + \dots \approx 10^9$$

What's our range then?

$$10^{-10} \dots 10^{-9}$$

This is called fixed-point arithmetic, and it's pretty bad.

$$\underbrace{2^{-29}+2^{-30}+2^{-32}}+2^{-35}$$



Should be able to do better.

Idea: Set a few bits aside to store the largest exponent. How?

$$2^{29}+2^{27} = (2^0+2^{-2})\cdot 2^{29} = (1.01)_2 \cdot 2^{29}$$

$$2^{-29}+2^{-30}+2^{-32}+2^{-35} = (2^0+2^{-1}+2^{-3}+2^{-6})\cdot 2^{-29} = (1.101001)_2 \cdot 2^{-29}$$

"Floating point"

"significand" "exponent"