Some aspects of how computers work

Memory

- CPU
- Registers
- Cache
- RAM
- Storage devices

Integers

- Suppose an integer is 32 bits
- 0 through $2^{31} - 1$ are stored in their standard binary representation
- Negative numbers are usually stored in a 2's complement form:
  - $x \in \{-2^{31} + 1, \ldots, -1\}$ is stored as $2^{32} + x$
  - this is equivalent to writing $|x|$ in binary and switching all the bits and adding one
  - note that first bit does act as a sign bit

$$\text{Example:} \quad -19 \;=\; 2^{32} - 19 = 4294967296 - 19 = 4294967277$$
$$=\; 11111111111111111111111111101101$$

Floating Point Numbers

- Any real number can be written as $x = (-1)^s * 2^I * M$ for $s = \pm 1$, some integer I, and some real number M, $1 \leq M < 2$
- A 32-bit `float` might have the following bit allocation:

  s: 1 bit

  I: 8 bits, $-2^7 < I \leq 2^7$ as $I + 2^7 - 1$

  M: 23 bits, as $M - 1$ in binary

- Examples:

$$\begin{aligned}
2 &= 0\ 10000000\ 00000000000000000000000 \\
-5.25 &= 1\ 10000001\ 01010000000000000000000 \\
13.7 &= 0\ 10000010\ 10110110011001100110011 = \mathrm{0x415B3333}
\end{aligned}$$

- Actual implementation varies by architecture
- Most real numbers can't be stored exactly
- Precision varies by magnitude of the number
- Sub-normal numbers

Floating Point Addition

Suppose we want to add two positive numbers, $x_3 = x_1 + x_2$
$x_1 = 2^{I_1} * M_1$, $x_2 = 2^{I_2} * M_2$, and assume $I_1 \geq I_2$

- First align the numbers: $x_2 = 2^{I_1}\left(M_2 * 2^{I_2 - I_1}\right)$
- Next add the fractions: $M_3 = (M_1 - 1) + (M_2 * 2^{I_2 - I_1} - 1) + 1$
- If $M_3 \geq 2$ then $M_3 = M_3/2$ and $I_3 = I_1 + 1$; else $I_3 = I_1$
- $x_3 = 2^{I_3} * M_3$

Overflow and Underflow

- Problems with overflow are apparent
- Underflow:
  - What is the smallest x such that $1 + x = 1$ in floating point arithmetic?
  - What is the largest x such that $x + 1 = x$?
- Rounding

`floatadd.c`

Double Precision

- Numbers are stored similarly to single precision
- A few more bits for I, many more bits for M
- Not much change in the range of possible numbers, but a big change in the limits of rounding and underflow

`doubleadd.c`

Ideas to Improve Precision

- Work with logarithms
- Prefer multiplication to division
- Divide/Multiply by a large constant
- Carefully order operations
- Don't use the optimizer