So Far

- Can do logical, add, subtract, multiply, divide, ...
- But........
  - what about fractions?
  - what about really large numbers?

Binary Fractions

\[ 1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \]
so...
\[ 101.011_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} \]
e.g.,
\[ .75 = 3/4 = 3/2 = 1/2 + 1/4 = .11 \]

Recall Scientific Notation

<table>
<thead>
<tr>
<th>sign</th>
<th>decimal point</th>
<th>exponent</th>
</tr>
</thead>
<tbody>
<tr>
<td>+6.02 x 10</td>
<td></td>
<td>23</td>
</tr>
<tr>
<td>1.673 x 10^{24}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Issues:
- Arithmetic (+, -, *, /)
- Representation, Normal form
- Range and Precision
- Rounding
- Exceptions (e.g., divide by zero, overflow, underflow)
- Errors
- Properties (negation, inversion, if A = B then A - B = 0)

Floating-Point Numbers

Representation of floating point numbers in IEEE 754 standard:

<table>
<thead>
<tr>
<th>sign</th>
<th>exponent</th>
<th>mantissa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>23</td>
</tr>
</tbody>
</table>

- single precision
- exponent: excess 127
- binary integer

\[ N = (-1)^S \times 2^{E-127} \times (1.M) \]
0 < E < 255
- 0 = 0 00000000 0 . . . 0
- 1.5 = 1 01111111 10 . . . 0
- 325 = 101000110 X 2^8 = 1.010001101 X 2^8
- 0.02 = .001101101100 . . . X 2^8 = 1.1001101100 . . . X 2^-3
- 0.01111100 1001101100...

- range of about 2 X 10^{-38} to 2 X 10^{38}
- always normalized (so always leading 1, thus never shown)
- special representation of 0 (E = 00000000) (why?)
- can do integer compare for greater-than, sign
What do you notice?

- 0
- 1.5 * 2^{-100}
- 1.75 * 2^{-100}
- 1.5 * 2^{100}
- 1.75 * 2^{100}

Does this work with negative numbers, as well?

Double Precision Floating Point

Representation of floating point numbers in IEEE 754 standard:

\[ N = (-1)^s \times 2^{E-1023} \times (1.M) \]

- 52 (+1) bit mantissa
- range of about $2 \times 10^{-308}$ to $2 \times 10^{308}$

Floating Point Addition

- How do you add in scientific notation?
  \[ 9.962 \times 10^4 + 5.231 \times 10^3 \]

- Basic Algorithm
  1. Align
  2. Add
  3. Normalize
  4. Round

FP Addition Hardware

Control

Small ALU

Big ALU

Sign

Exponent

Significand

Sign

Exponent

Significand

Compare exponents

Shift smaller number right

Add

Increment or decrement

Shift left or right

Normalize

Round

Rounding hardware

Result
Floating Point Multiplication

- How do you multiply in scientific notation?
  \((9.9 \times 10^4)(5.2 \times 10^2) = 5.148 \times 10^7\)

- Basic Algorithm
  1. Add exponents
  2. Multiply
  3. Normalize
  4. Round
  5. Set Sign

FP Accuracy

- Extremely important in scientific calculations
- Very tiny errors can accumulate over time
- IEEE 754 FP standard has four rounding modes
  - always round up (toward \(+\infty\))
  - always round down (toward \(-\infty\))
  - truncate
  - round to nearest
    \(\Rightarrow\) in case of tie, round to nearest even
- Requires extra bits in intermediate representations

Extra Bits for FP Accuracy

- **Guard bits** -- bits to the right of the least significant bit of the significand computed for use in normalization (could become significant at that point) and rounding.
- IEEE 754 has three extra bits and calls them *guard*, *round*, and *sticky*.

Key Points

- Floating Point extends the range of numbers that can be represented, at the expense of precision (accuracy).
- FP operations are very similar to integer, but with pre- and post-processing.
- Rounding implementation is critical to accuracy over time.