**Divide: Paper & Pencil**

\[
\begin{array}{c|c}
\text{Quotient} & 1101 \\
\hline
\text{Divisor} & 1101010 \\
\text{1000} & \\
\text{1010} & \\
\text{1000} & \\
\text{101} & \\
\text{-0000} & \\
\text{1010} & \\
\text{-1000} & \\
\text{10} & \\
\end{array}
\]

- See how big a number can be subtracted, creating quotient bit on each step
  - Binary => 1 * divisor or 0 * divisor
- Dividend = Quotient \times Divisor + Remainder

**DIVIDE HARDWARE**

**Version 1**
- 64-bit Divisor reg, 64-bit ALU, 64-bit Remainder reg, 32-bit Quotient reg

**DIVIDE HARDWARE**

**Version 3**
- 32-bit Divisor reg, 32-bit ALU, 64-bit Remainder reg, (0-bit Quotient reg)
Observations on Divide
Version 3

- Same Hardware as __________: just need ALU to add or subtract, and 63-bit register to shift left or shift right
- Hi and Lo registers in MIPS combine to act as 64-bit register for multiply and divide
- Signed Divides: Simplest is to remember signs, make positive, and complement quotient and remainder if necessary
  - Note: __________ and _____________ must have same sign
  - Note: Quotient negated if Divisor sign & Dividend sign disagree

So Far

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

Binary Fractions

1011_2 = 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0
so...
101.011_2 = 1x2^2 + 0x2^1 + 1x2^0 + 0x2^-1 + 1x2^-2 + 1x2^-3
e.g.,
.75 = 3/4 = 3/2 = 1/2 + 1/4 = .11

Recall Scientific Notation

\[ +6.02 \times 10^{23} = \frac{1}{2} + \frac{1}{4} = 0.11 \]

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>single precision</th>
<th>sign</th>
<th>E</th>
<th>M</th>
</tr>
</thead>
</table>

- **Exponent:**
  - Excess 127 binary integer
  - (actual exponent is \( e = E - 127 \))

- **Mantissa:**
  - Sign + magnitude, normalized binary significand with hidden integer bit: 1.M

\[
N = (-1)^S \times 2^{E-127} \times (1.M)
\]

- \( 0 < E < 255 \)
- \( 0 = 0.0000000000... \)
- \( 1.5 \times 2^{-100} = 0.00110111 10... \)
- \( 1.75 \times 2^{-100} = 0.00110111 1100000000... \)
- \( 1.5 \times 2^{100} = 0.11100011 1000000000... \)
- \( 1.75 \times 2^{100} = 0.11100011 1100000000... \)

- **What do you notice?**
  - \( 0 \)
  - \( 1.5 \times 2^{-100} \)
  - \( 1.75 \times 2^{-100} \)
  - \( 1.5 \times 2^{100} \)
  - \( 1.75 \times 2^{100} \)

- Does this work with negative numbers, as well?

Double Precision Floating Point

Representation of floating point numbers in IEEE 754 standard:

<table>
<thead>
<tr>
<th>double precision</th>
<th>sign</th>
<th>E</th>
<th>M</th>
</tr>
</thead>
</table>

- **Exponent:**
  - Excess 1023 binary integer

- **Mantissa:**
  - Sign + magnitude, normalized binary significand with hidden integer bit: 1.M

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

- \( 0 < E < 2048 \)
- \( 52 (+1) \) bit mantissa
- \( \text{range of about } 2 \times 10^{-308} \text{ to } 2 \times 10^{308} \)

Floating Point Addition

- **How do you add in scientific notation?**
  - \( 9.962 \times 10^4 + 5.231 \times 10^2 \)

- **Basic Algorithm**
  1. 
  2. Add
  3. 
  4. Round
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.
  3. Normalize
  4.
  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
    => 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

- Multiplication and division take much longer than addition, requiring multiple addition steps.
- 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.