

Chapter 3

Computer Abstractions and Technology

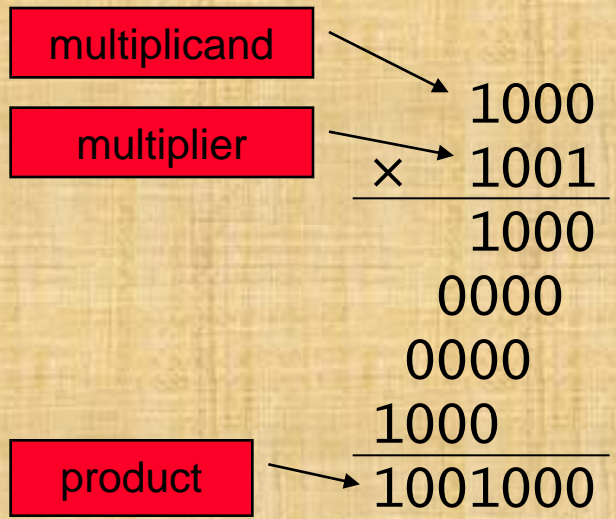
Lesson 7: Arithmetic for Computers (cont.)

So far....

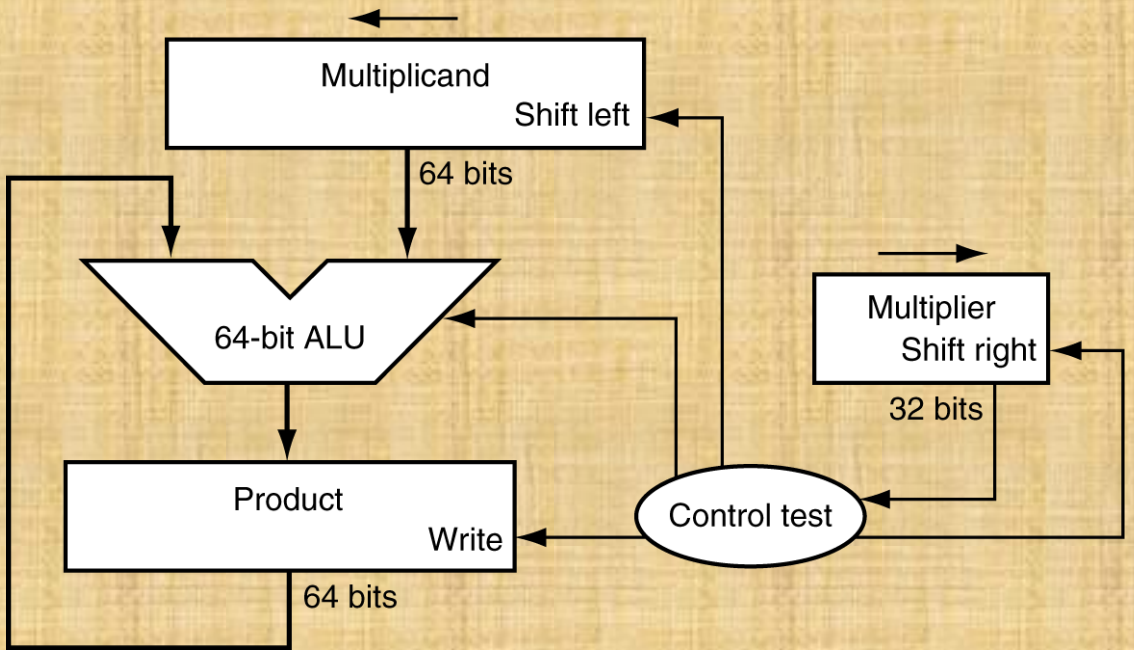
- ◆ Number representation
- ◆ Sign Extend/Unsigned
- ◆ 2's complement
- ◆ Operations on integers
 - Addition and subtraction
- ◆ Today....
 - Multiplication and division
 - Dealing with overflow
- ◆ Floating-point real numbers
 - Representation and operations

Multiplication

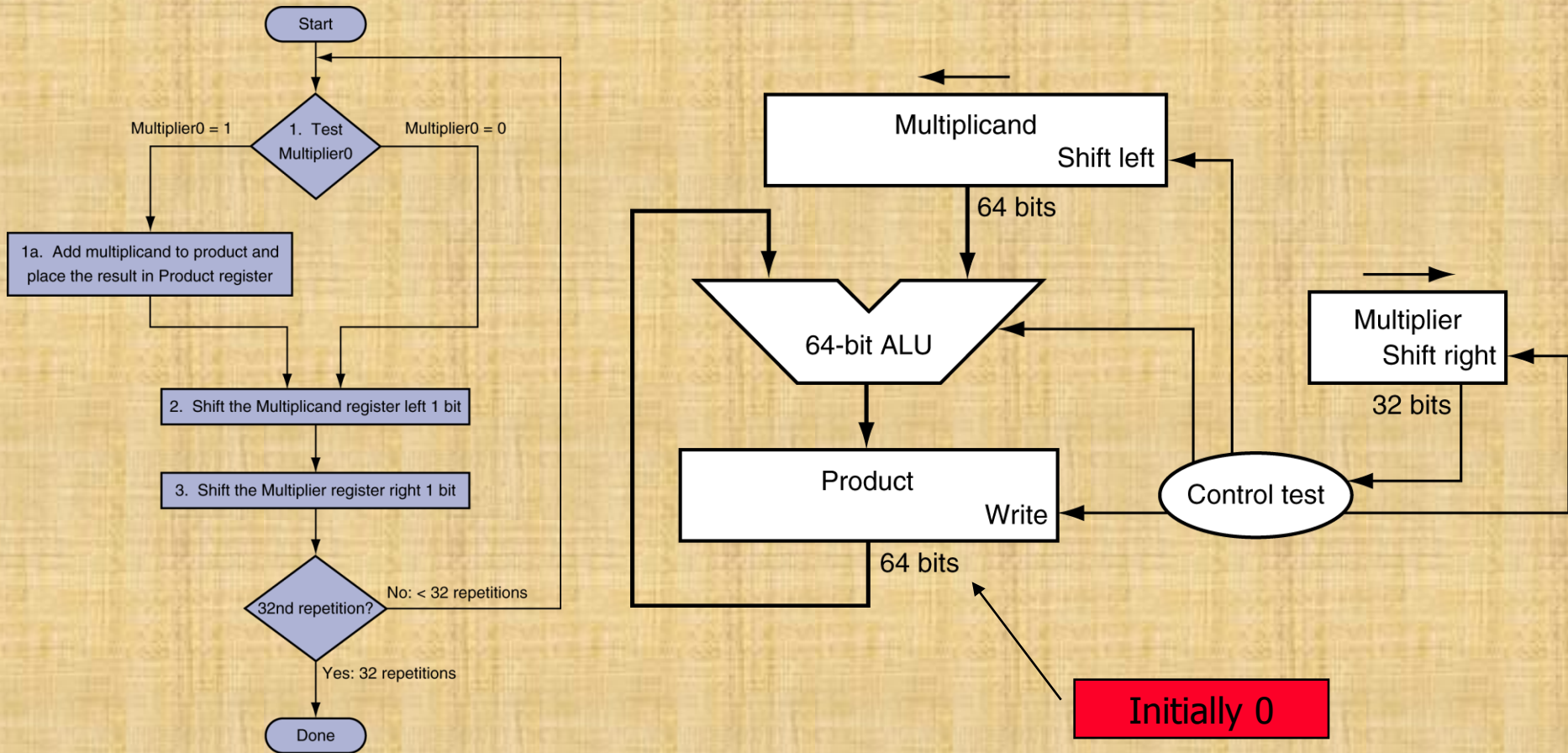
- ◆ Start with long-multiplication approach



Length of product is the sum of operand lengths

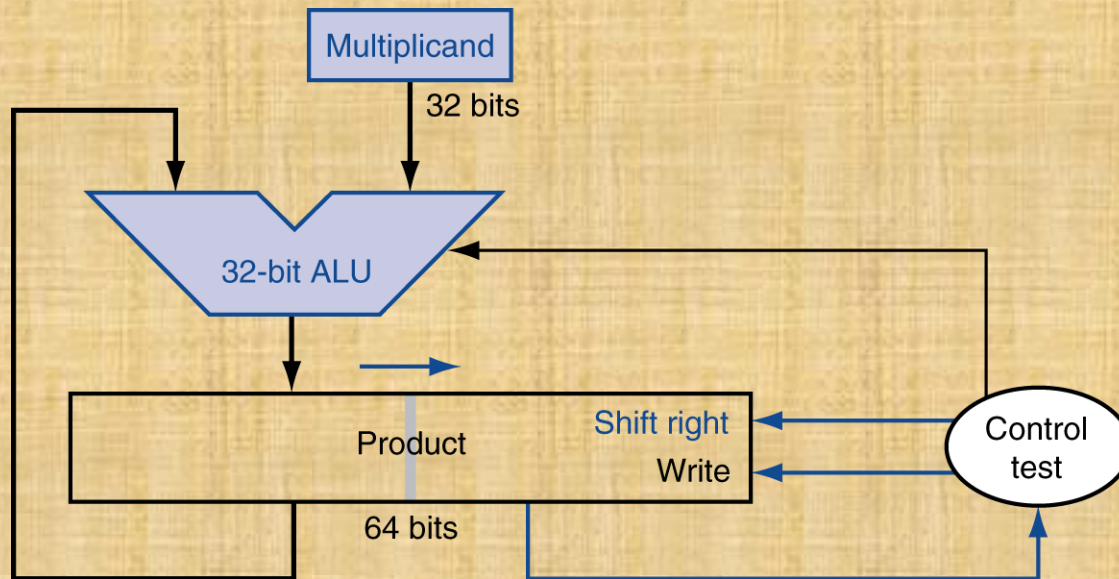


Multiplication Hardware



Optimized Multiplier

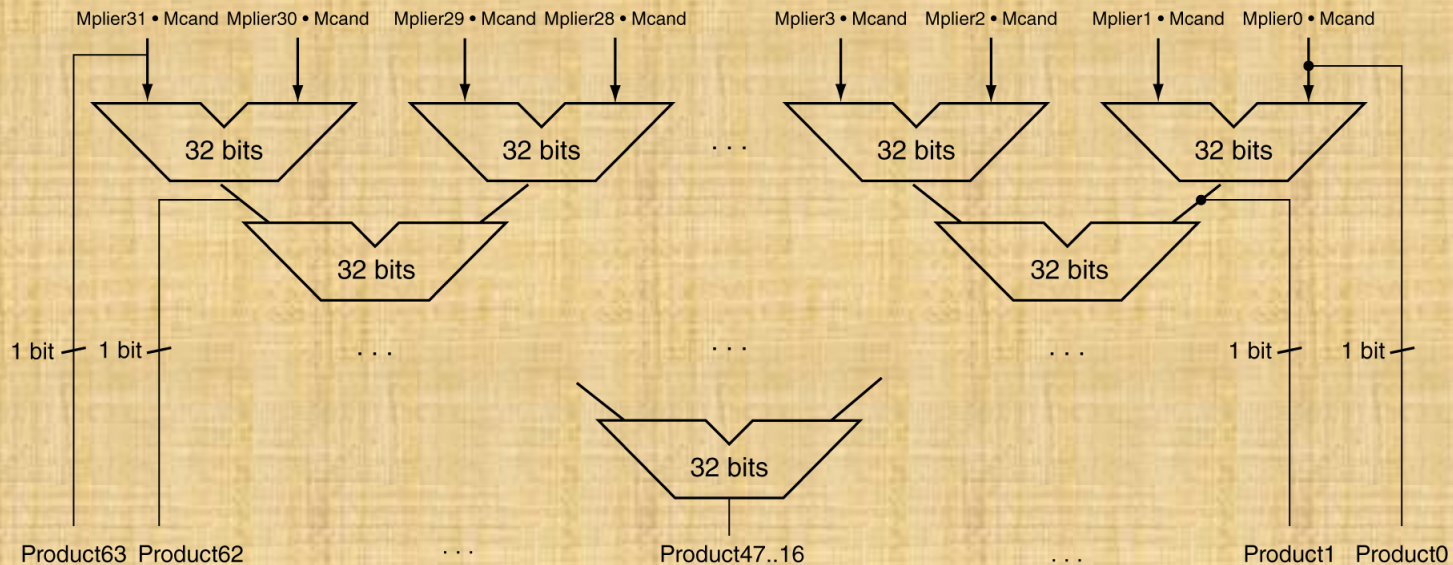
- ◆ Perform steps in parallel: add/shift



- One cycle per partial-product addition
 - That's ok, if frequency of multiplications is low

Faster Multiplier

- ◆ Uses multiple adders
 - Cost/performance tradeoff

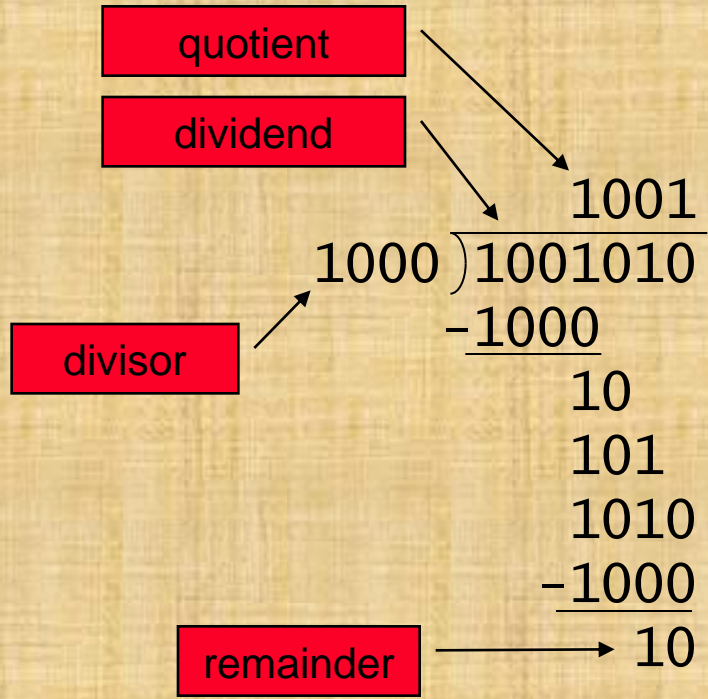


- Can be pipelined
 - Several multiplication performed in parallel

MIPS Multiplication

- ◆ Two 32-bit registers for product
 - HI: most-significant 32 bits
 - LO: least-significant 32-bits
- ◆ Instructions
 - `mult rs, rt` / `multu rs, rt`
 - 64-bit product in HI/LO
 - `mfhi rd` / `mflo rd`
 - Move from HI/LO to rd
 - Can test HI value to see if product overflows 32 bits
 - `mul rd, rs, rt`
 - Least-significant 32 bits of product → rd

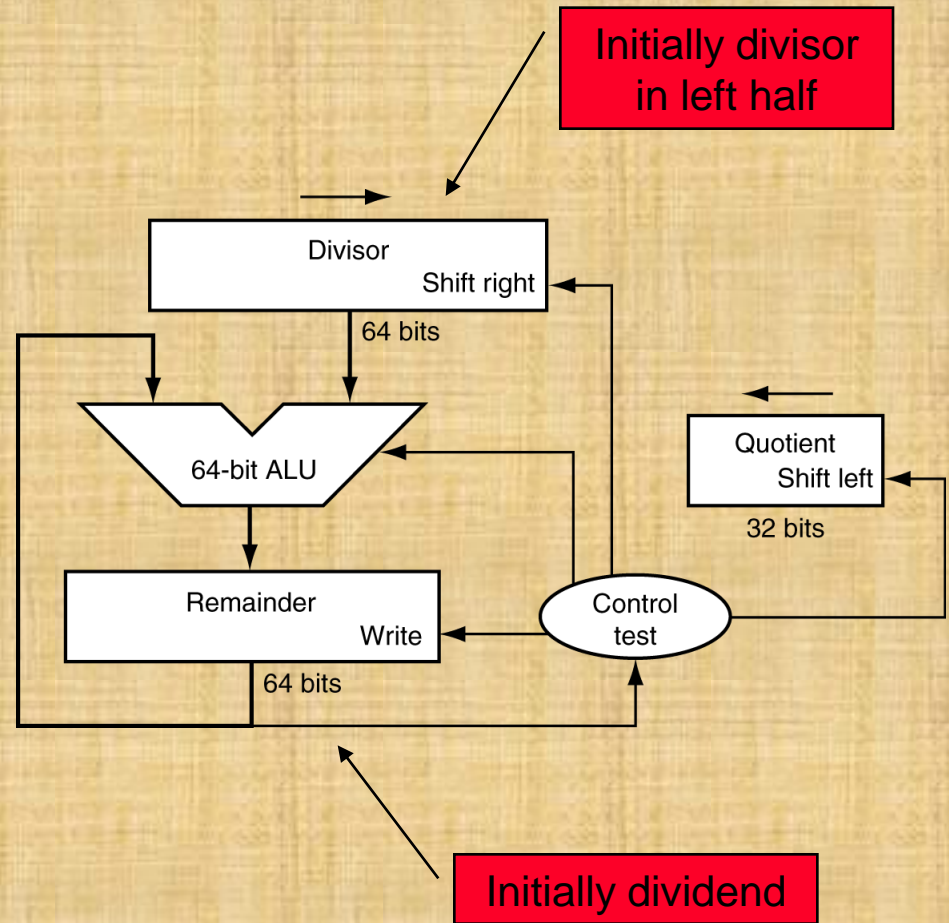
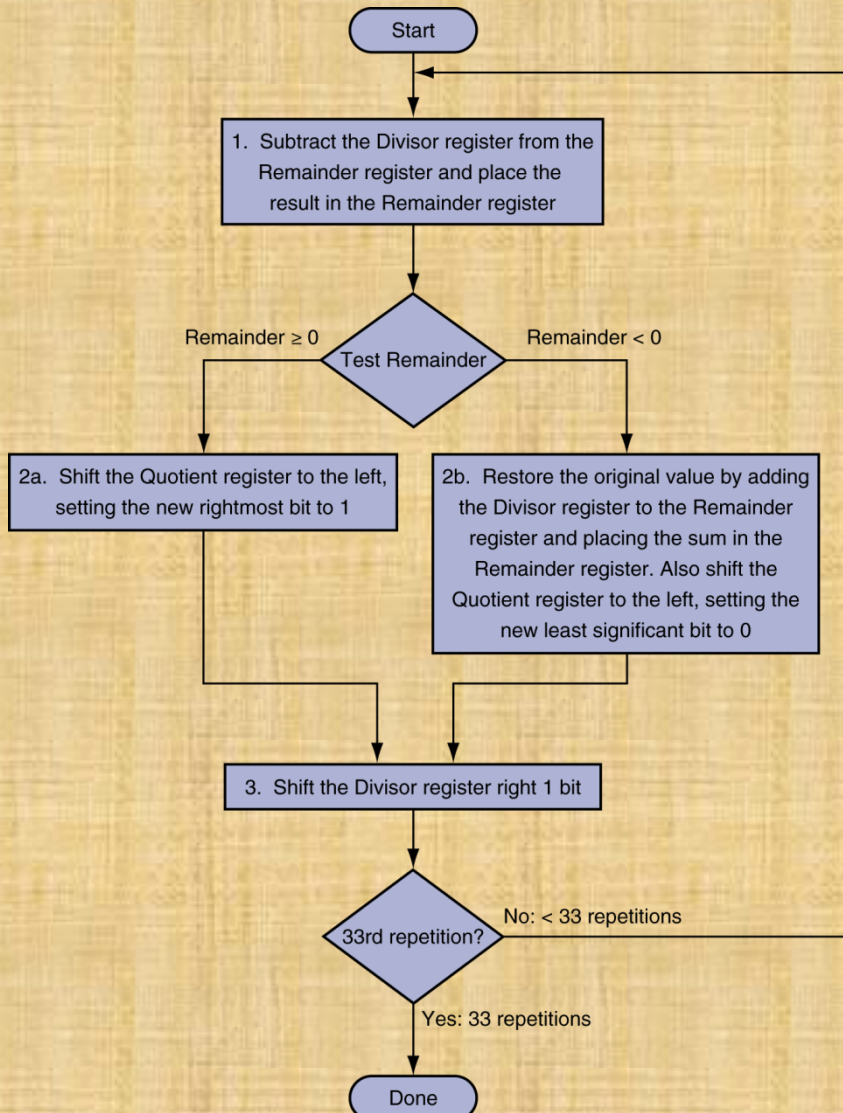
Division



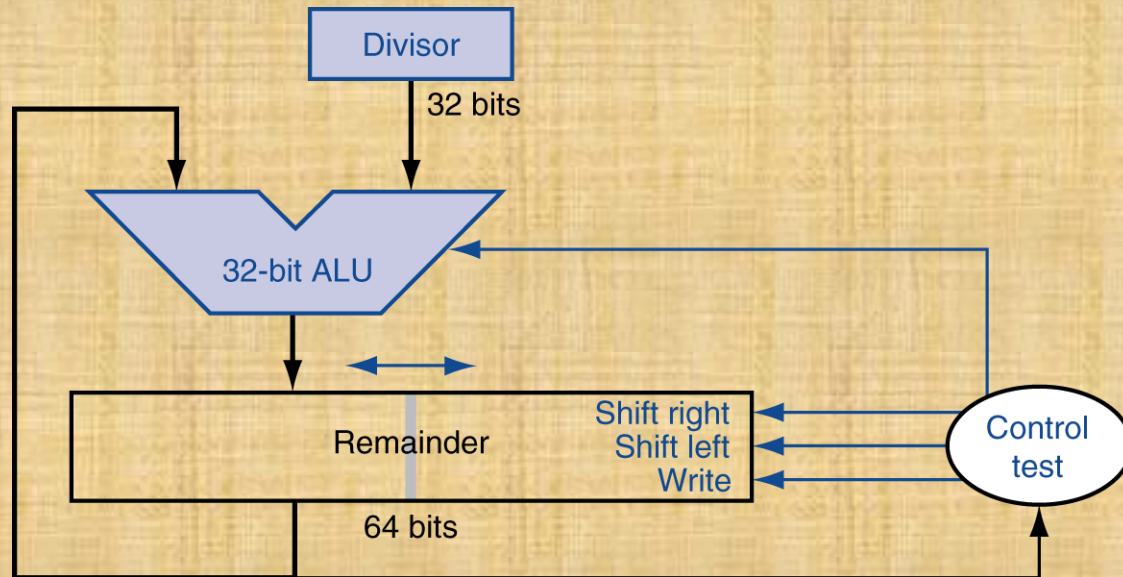
n-bit operands yield *n*-bit quotient and remainder

- ◆ Check for 0 divisor
- ◆ Long division approach
 - If divisor ≤ dividend bits
 - 1 bit in quotient, subtract
 - Otherwise
 - 0 bit in quotient, bring down next dividend bit
- ◆ Restoring division
 - Do the subtract, and if remainder goes < 0, add divisor back
- ◆ Signed division
 - Divide using absolute values
 - Adjust sign of quotient and remainder as required

Division Hardware



Optimized Divider



- ◆ One cycle per partial-remainder subtraction
- ◆ Looks a lot like a multiplier!
 - Same hardware can be used for both

Faster Division

- ◆ Can't use parallel hardware as in multiplier
 - Subtraction is conditional on sign of remainder
- ◆ Faster dividers (e.g. SRT division) generate multiple quotient bits per step
 - Still require multiple steps

MIPS Division

- ◆ Use HI/LO registers for result
 - HI: 32-bit remainder
 - LO: 32-bit quotient
- ◆ Instructions
 - `div rs, rt / divu rs, rt`
 - No overflow or divide-by-0 checking
 - Software must perform checks if required
 - Use `mfhi`, `mflo` to access result

Floating Point

- ◆ Representation for non-integral numbers
 - Including very small and very large numbers
- ◆ Like scientific notation
 - -2.34×10^{56} ← normalized
 - $+0.002 \times 10^{-4}$ ← not normalized
 - $+987.02 \times 10^9$ ← not normalized
- ◆ In binary
 - $1.xxxxxxx_2 \quad 2^{yyyy}$
- ◆ Types `float` and `double` in C

Next time

- ◆ Floating point arithmetic
 - Floating-point approximation to reals
- ◆ Bounded range and precision
 - Operations can overflow and underflow