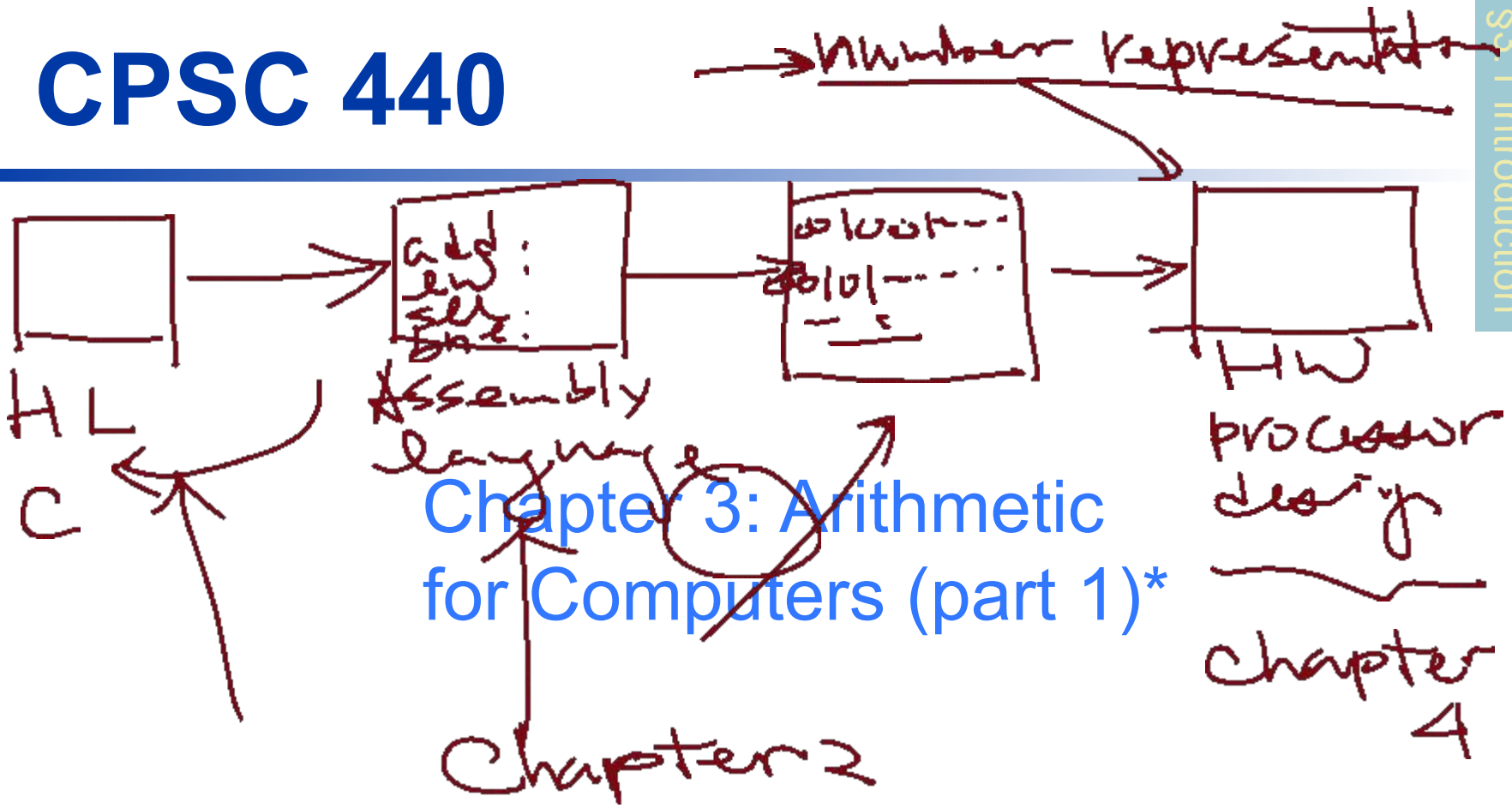
CPSC 440



Chapter 3: Arithmetic for Computers (part 1)*

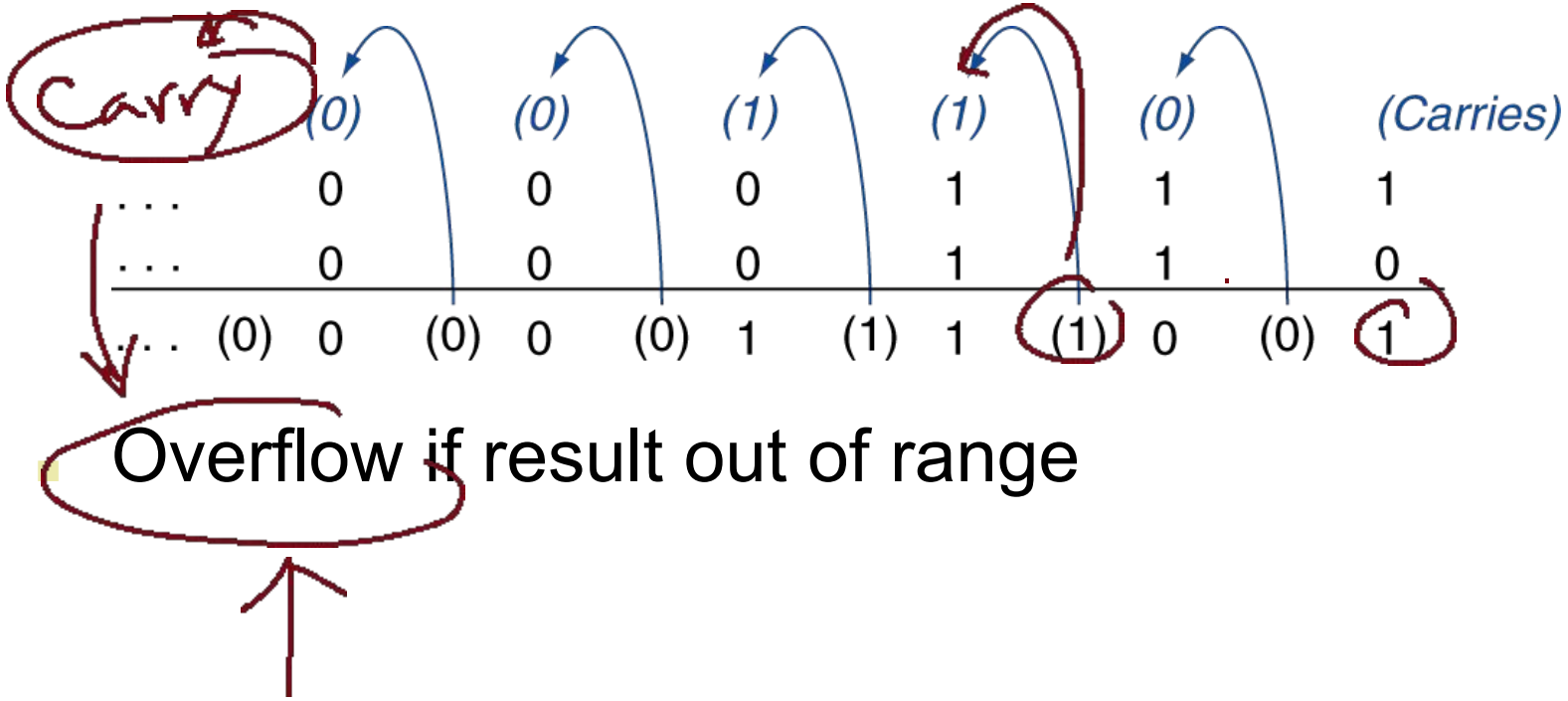
* This ppt is provided by the publisher of the textbook

Arithmetic for Computers

- Operations on integers
  - Addition and subtraction
  - Multiplication and division
  - Dealing with overflow
- Floating-point real numbers
  - Representation and operations

Integer Addition

- Example: 7 + 6



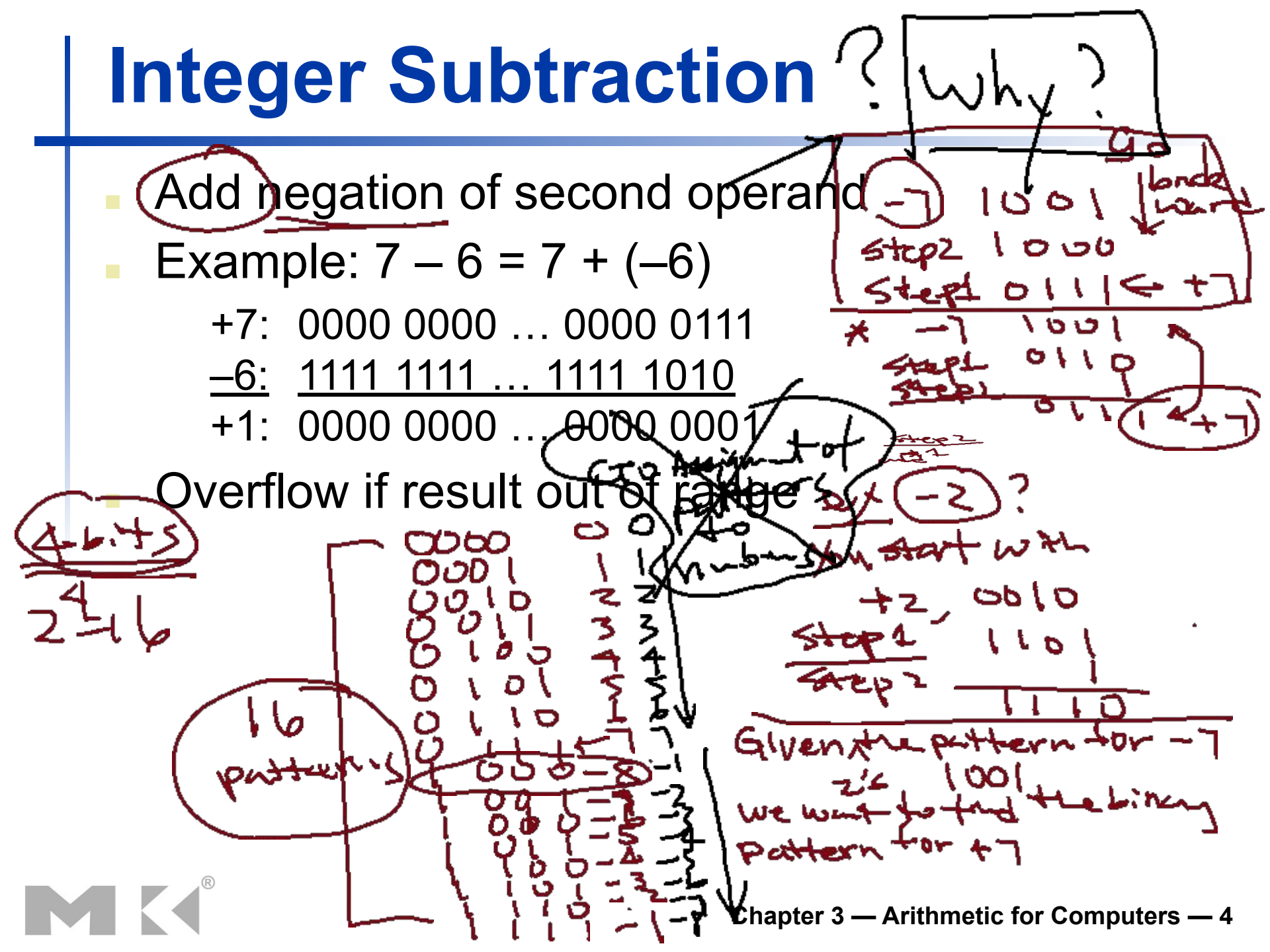
- Overflow if result out of range

Integer Subtraction

- Add negation of second operand
- Example: 7 – 6 = 7 + (–6)

  +7: 0000 0000 … 0000 0111
  –6: 1111 1111 … 1111 1010
  +1: 0000 0000 … 0000 0001

- Overflow if result out of range



Dealing with Overflow

- Some languages (e.g., C) ignore overflow
  - Use MIPS `addu`, `addui`, `subu` instructions
- Other languages (e.g., Ada, Fortran) require raising an exception
  - Use MIPS `add`, `addi`, `sub` instructions
  - On overflow, invoke exception handler
    - Save PC in exception program counter (EPC) register
    - Jump to predefined handler address
    - `mfc0` (move from coprocessor reg) instruction can retrieve EPC value, to return after corrective action

Multiplication

- Start with long-multiplication approach

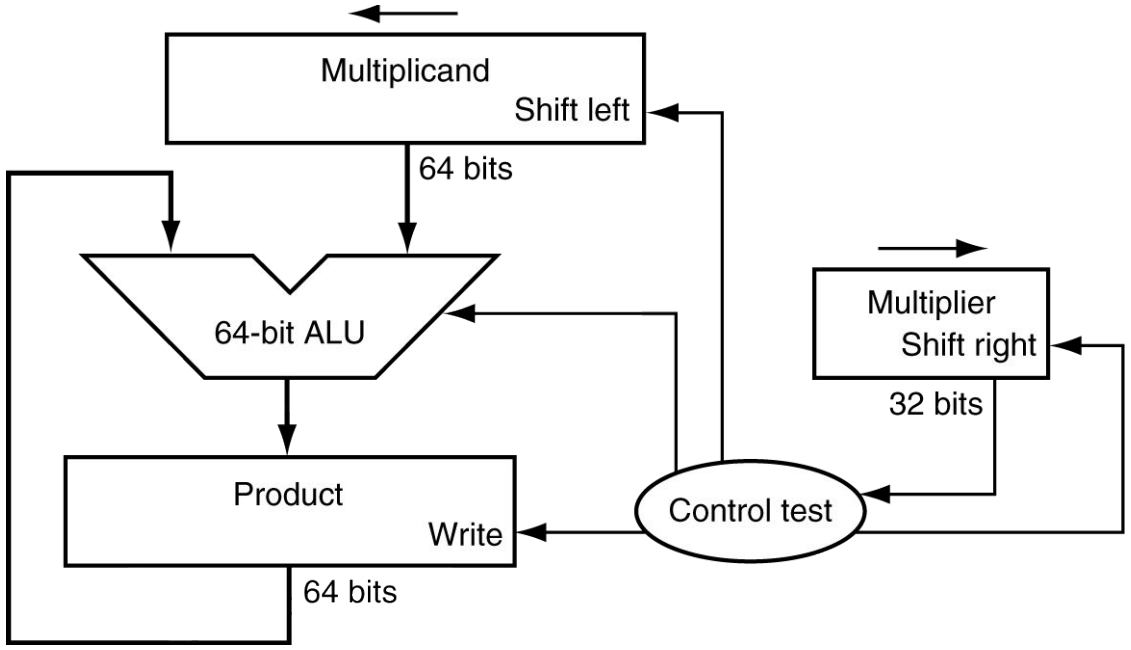
multiplicand

multiplier

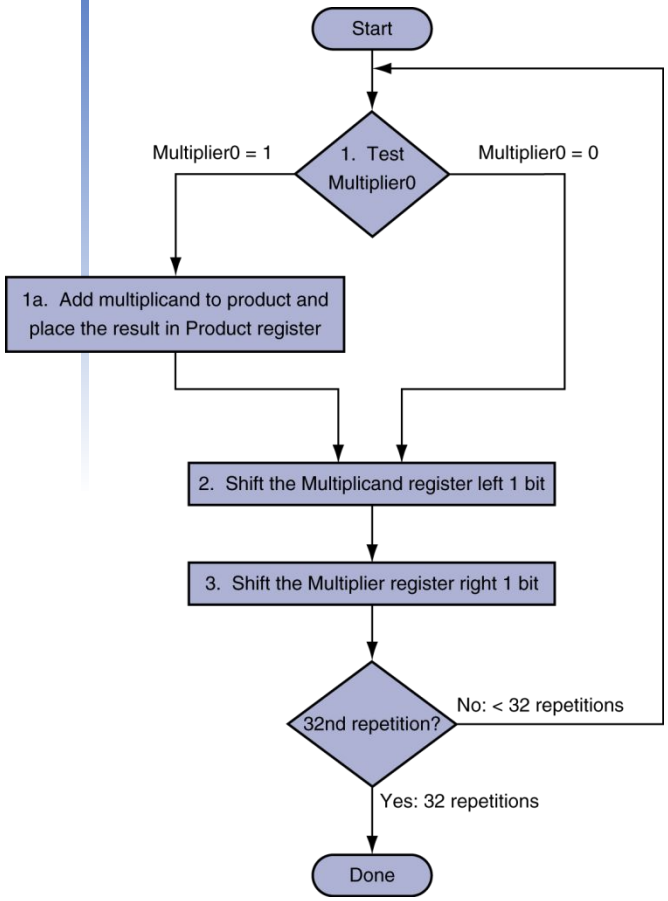
product

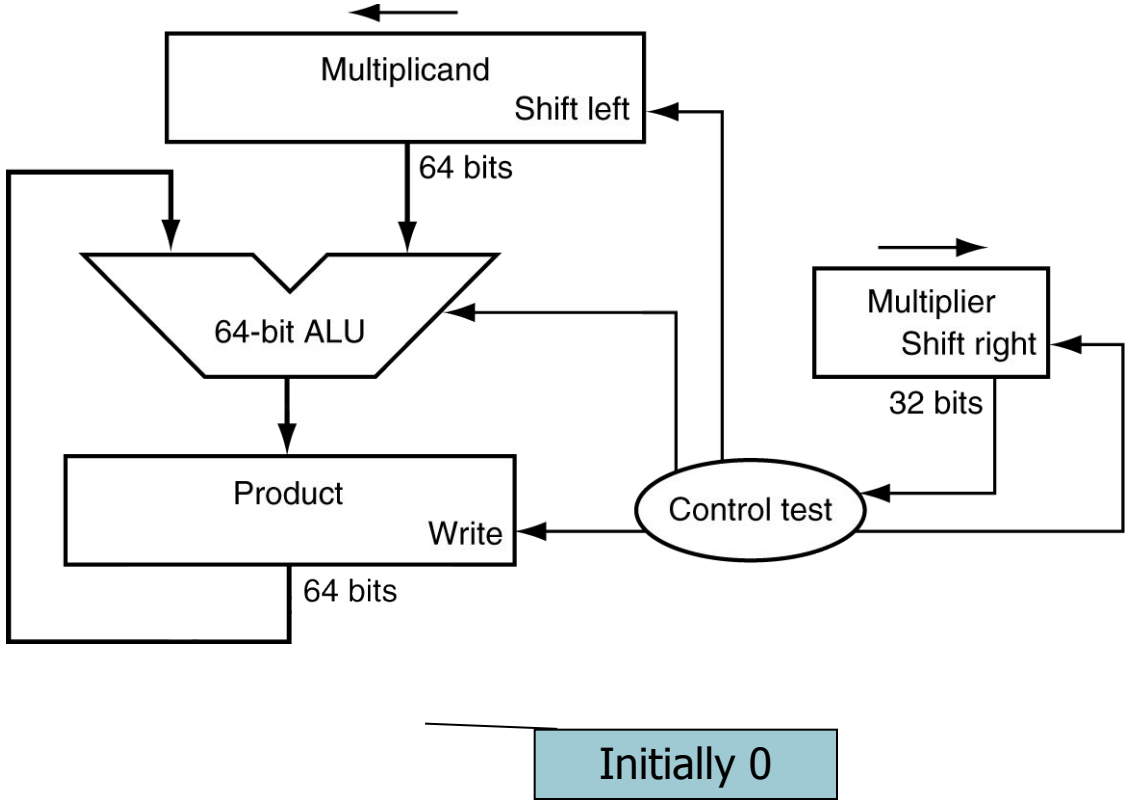
```
      1000
×     1001
------------
      1000
     0000
    0000
   1000
------------
   1001000
```

Length of product is the sum of operand lengths



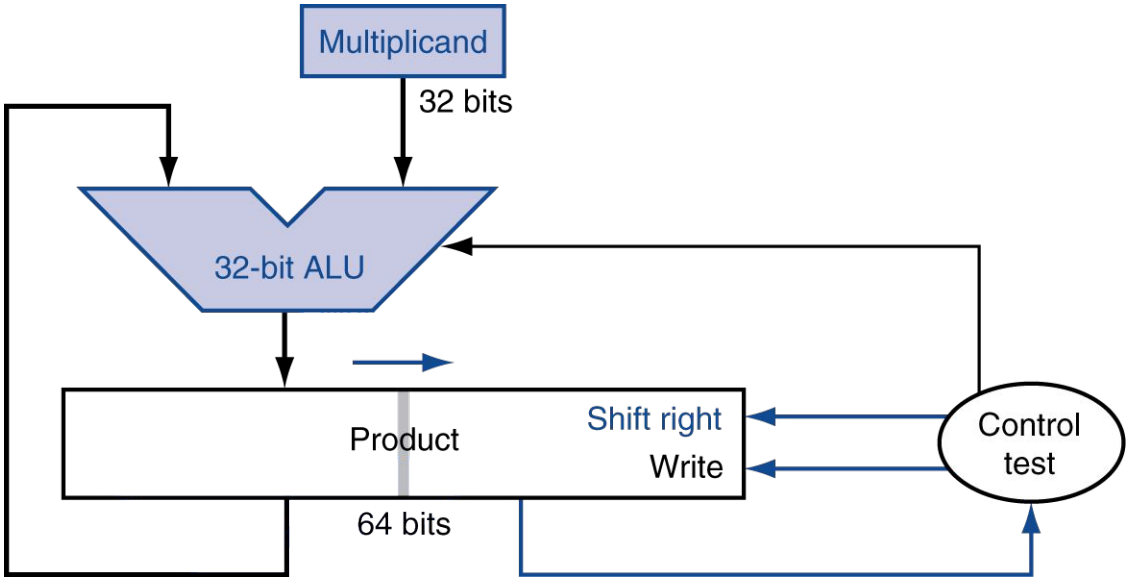
Multiplication Hardware





Optimized Multiplier

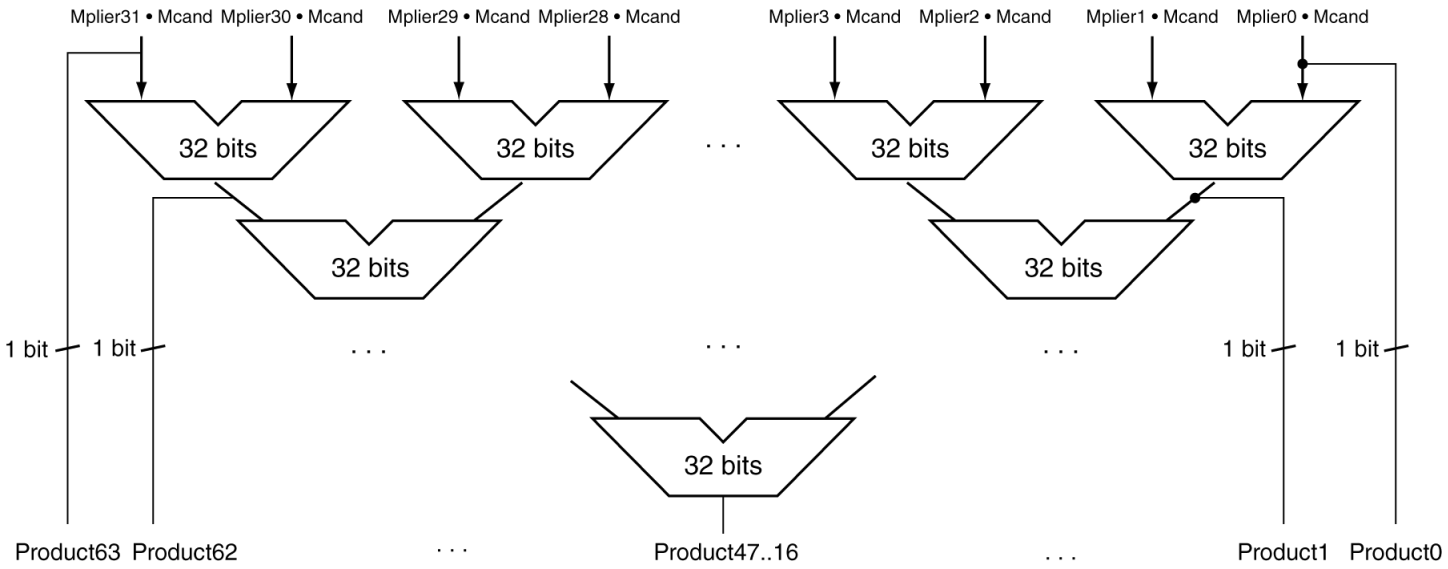
- Perform steps in parallel: add/shift



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

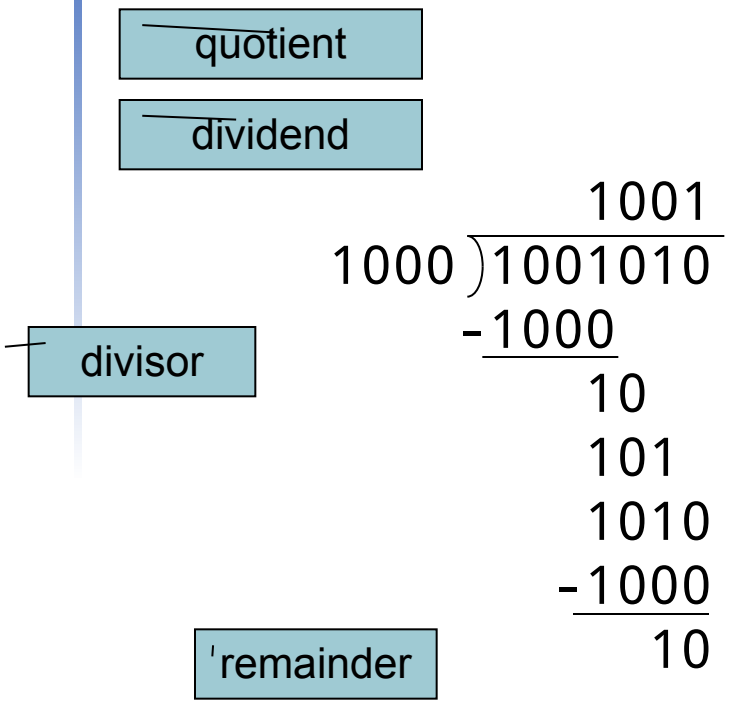
Faster Multiplier

- Uses multiple adders
  - Cost/performance tradeoff



- Can be pipelined
  - Several multiplication performed in parallel

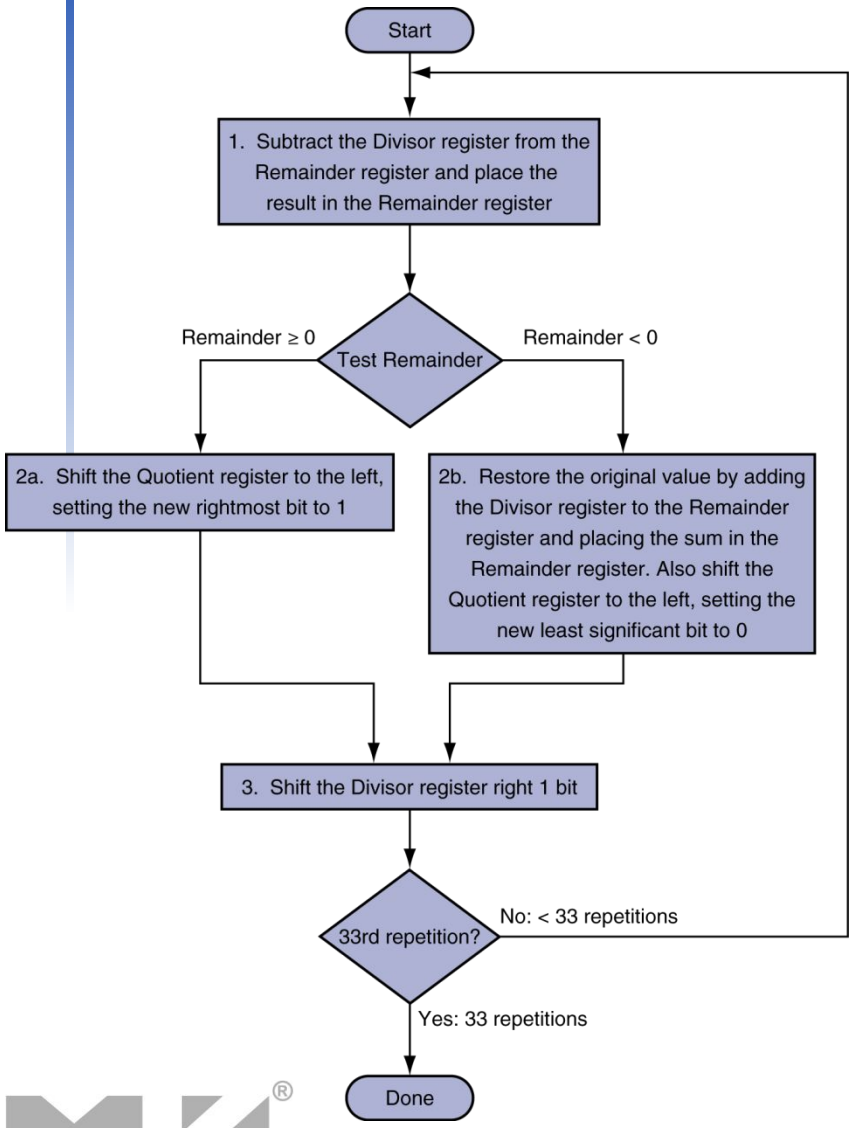
Division

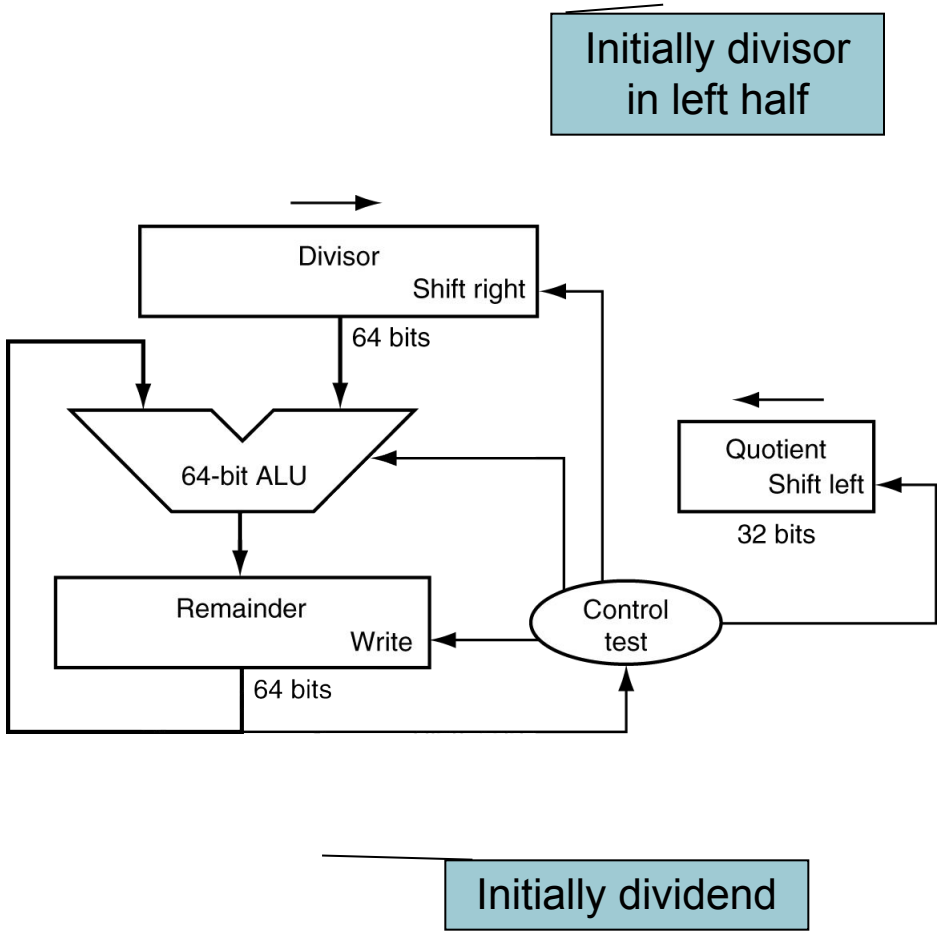


n-bit operands yield *n*-bit quotient and remainder

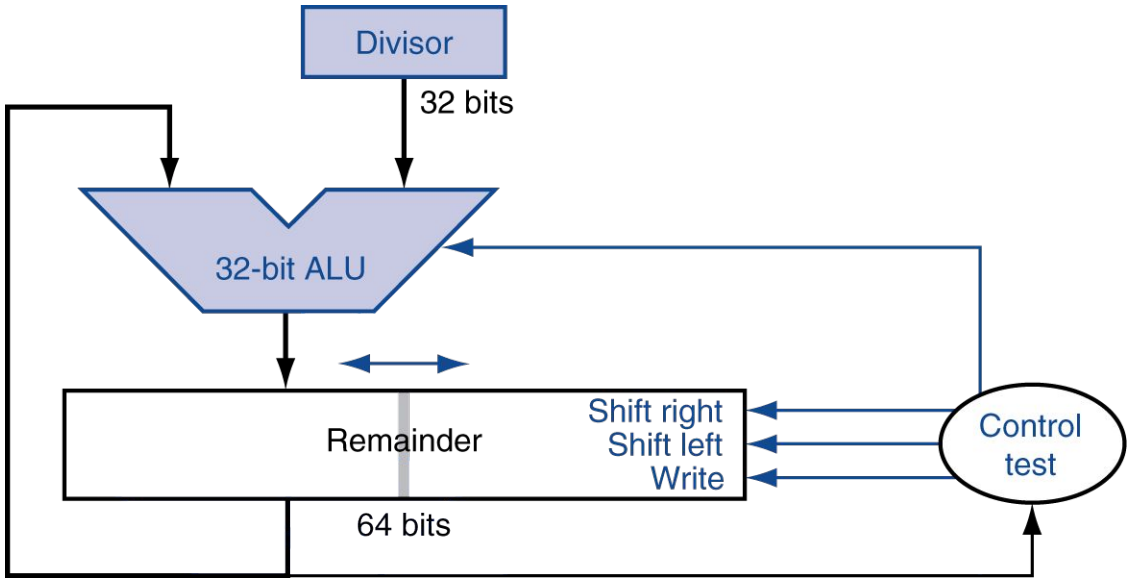
- Check for 0 divisor
- Long division approach
  - If divisor ≤ dividend bits
    - 1 bit in quotient, subtract
  - Otherwise
    - 0 bit in quotient, bring down next dividend bit
- Restoring division
  - Do the subtract, and if remainder goes < 0, add divisor back
- Signed division
  - Divide using 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 devision) generate multiple quotient bits per step
  - Still require multiple steps