

Arithmetic for Computers

S3.1 Introduction

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

MK® Chapter 3 — Arithmetic for Computers — 2

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



Arithmetic for Multimedia

- Graphics and media processing operates on vectors of 8-bit and 16-bit data
 - Use 64-bit adder, with partitioned carry chain
 - Operate on 8×8-bit, 4×16-bit, or 2×32-bit vectors
 - SIMD (single-instruction, multiple-data)
- Saturating operations
 - On overflow, result is largest representable value
 - c.f. 2s-complement modulo arithmetic
 - E.g., clipping in audio, saturation in video



Multiplication

§3.3 Multiplication

- Start with long-multiplication approach

multiplicand → 1000
multiplier × 1001

 1000
 0000
 0000
 1000

product → 1001000

Length of product is the sum of operand lengths

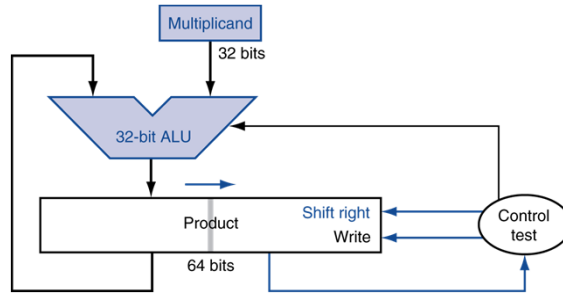
Chapter 3 — Arithmetic for Computers — 7

Multiplication Hardware

Chapter 3 — Arithmetic for Computers — 8

Optimized Multiplier

- Perform steps in parallel: add/shift

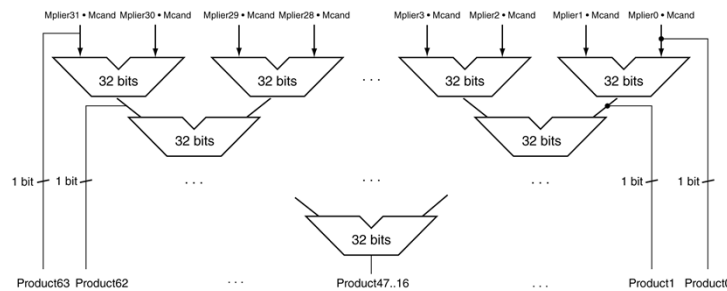


- One cycle per partial-product addition
 - That's ok, if frequency of multiplications is low



Faster Multiplier

- Uses multiple adders
 - Cost/performance tradeoff



- Can be pipelined
 - Several multiplication performed in parallel



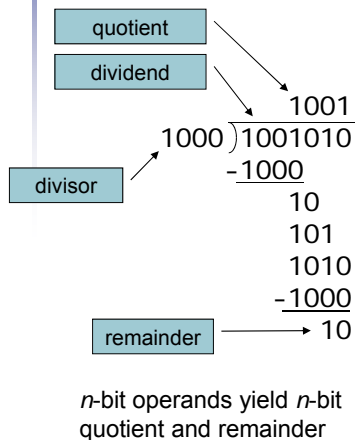
MIPS Multiplication

- 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



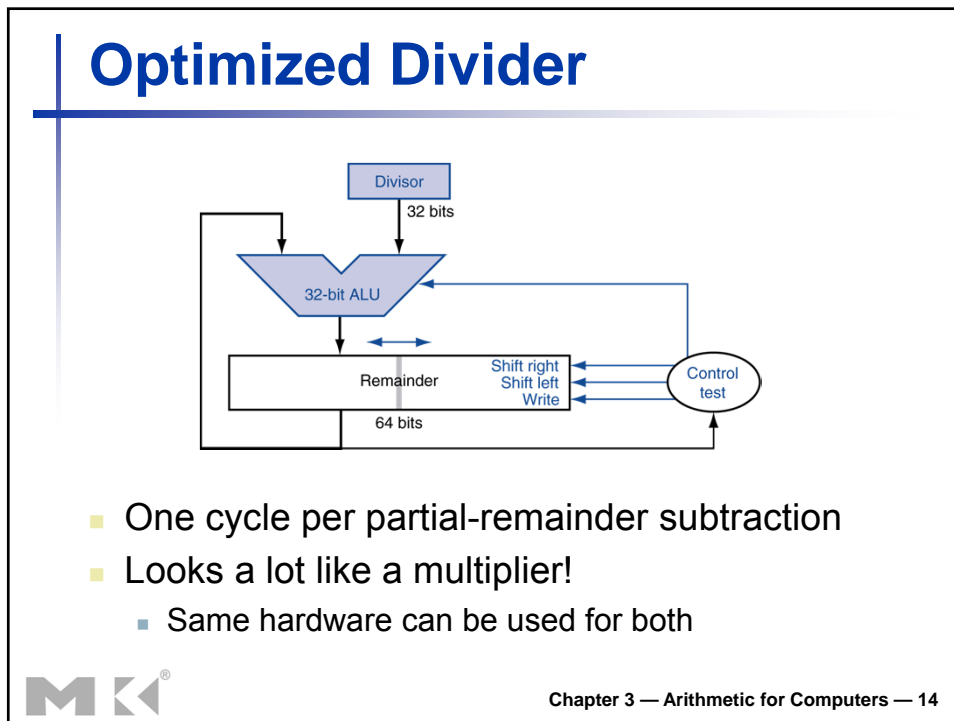
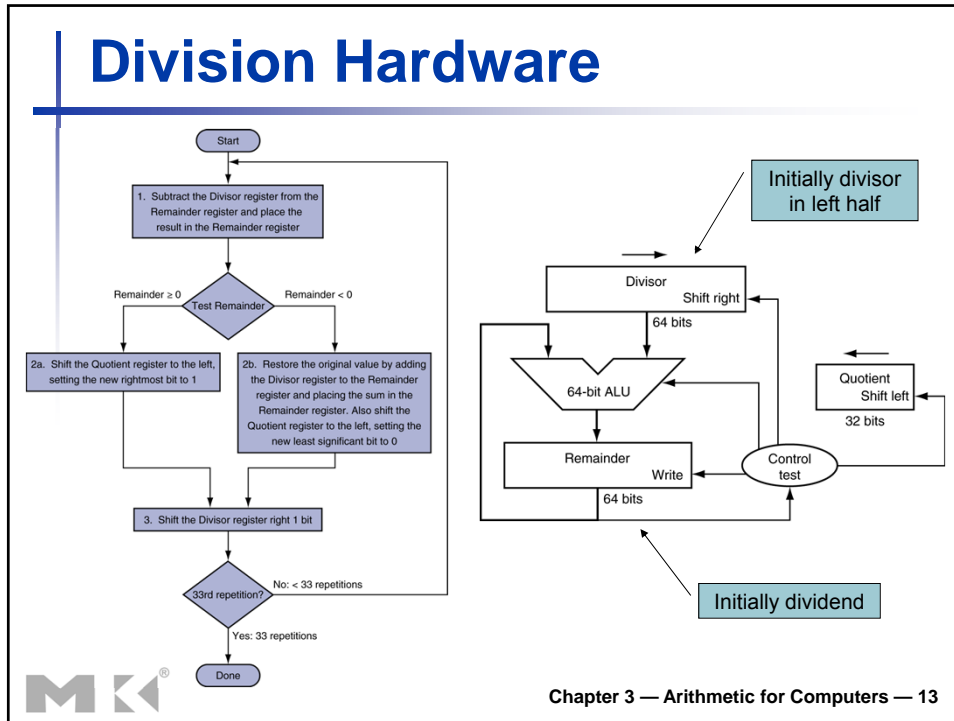
Division

§3.4 Division



- 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





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

