

**Computer Architecture
ECE 361
Lecture 7: ALU Design : Division**

361 div.1

Outline of Today's Lecture

- Introduction to Today's Lecture
- Divide
- Questions and Administrative Matters
- Introduction to Single cycle processor design

361 div.2

Divide: Paper & Pencil

	1001	Quotient
Divisor 1000	1001010	Dividend
	<u>-1000</u>	
	10	
	101	
	<u>1010</u>	
	-1000	
	10	Remainder (or Modulo result)

See how big a number can be subtracted, creating quotient bit on each step

Binary => 1 * divisor or 0 * divisor

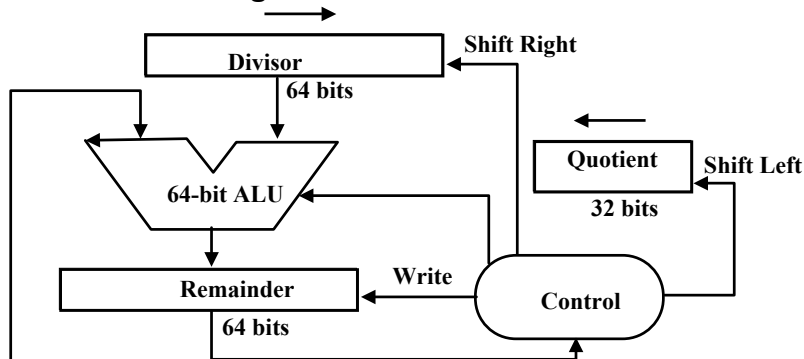
Dividend = Quotient x Divisor + Remainder
=> | Dividend | = | Quotient | + | Divisor |

3 versions of divide, successive refinement

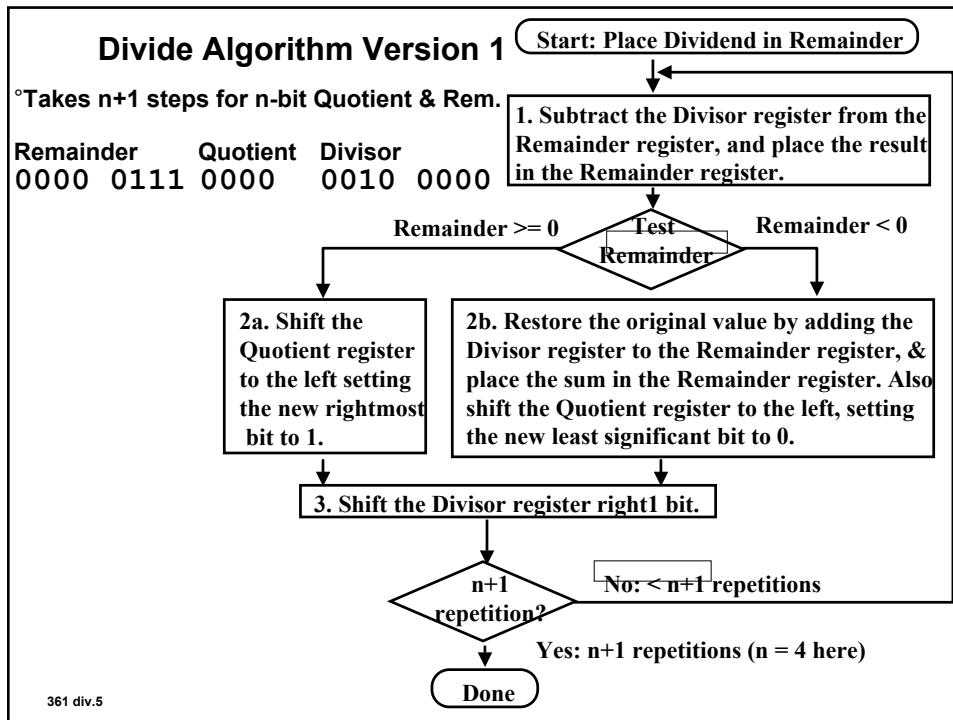
361 div.3

DIVIDE HARDWARE Version 1

° 64-bit Divisor reg, 64-bit ALU, 64-bit Remainder reg, 32-bit Quotient reg



361 div.4

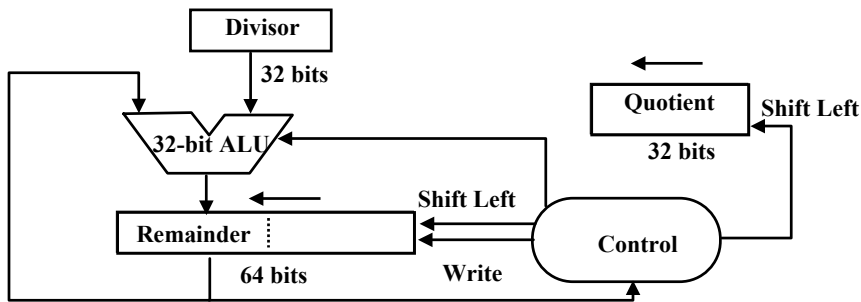


Observations on Divide Version 1

- ° 1/2 bits in divisor always 0
 => 1/2 of 64-bit adder is wasted
 => 1/2 of divisor is wasted
- ° Instead of shifting divisor to right,
 shift remainder to left?
- ° 1st step cannot produce a 1 in quotient bit
 (otherwise too big)
 => switch order to shift first and then subtract,
 can save 1 iteration

DIVIDE HARDWARE Version 2

- ° **32-bit Divisor reg, 32-bit ALU, 64-bit Remainder reg, 32-bit Quotient reg**

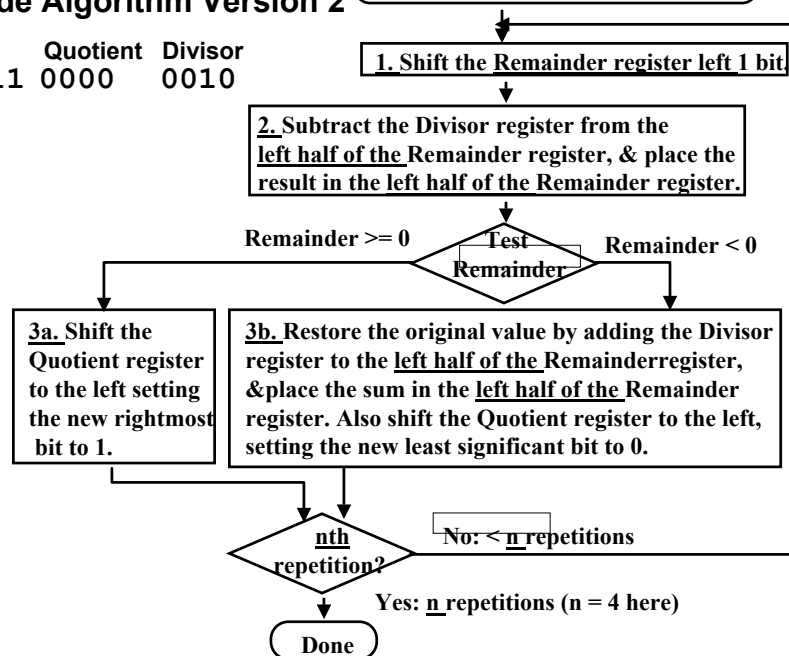


361 div.7

Divide Algorithm Version 2

Start: Place Dividend in Remainder

Remainder Quotient Divisor
0000 0111 0000 0010



361 div.8

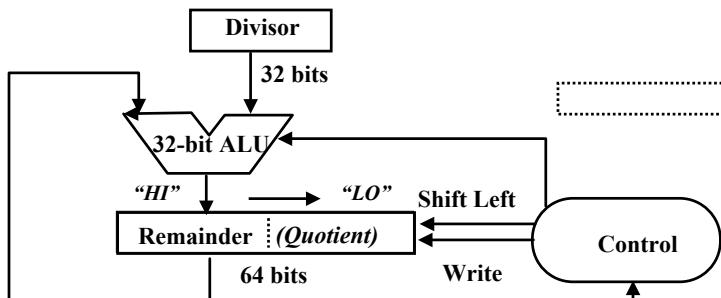
Observations on Divide Version 2

- Eliminate Quotient register by combining with Remainder as shifted left
 - Start by shifting the Remainder left as before.
 - Thereafter loop contains only two steps because the shifting of the Remainder register shifts both the remainder in the left half and the quotient in the right half
 - The consequence of combining the two registers together and the new order of the operations in the loop is that the remainder will shifted left one time too many.
 - Thus the final correction step must shift back only the remainder in the left half of the register

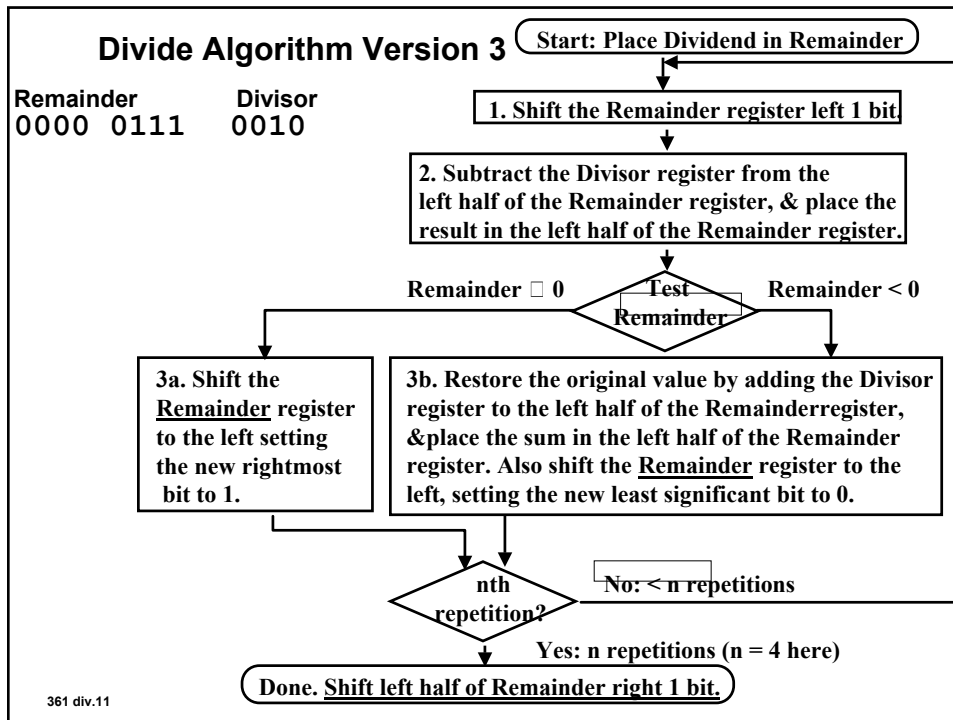
361 div.9

DIVIDE HARDWARE Version 3

- 32-bit Divisor reg, 32 -bit ALU, 64-bit Remainder reg, (0-bit Quotient reg)



361 div.10



Observations on Divide Version 3

- Same Hardware as Multiply: 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: Dividend and Remainder must have same sign
 - Note: Quotient negated if Divisor sign & Dividend sign disagree
e.g., $-7 \div 2 = -3$, remainder = -1
- Possible for quotient to be too large: if divide 64-bit integer by 1, quotient is 64 bits ("called saturation")

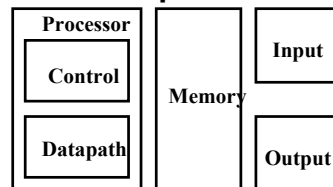
Summary

- Bits have no inherent meaning: operations determine whether they are really ASCII characters, integers, floating point numbers
- Divide can use same hardware as multiply: Hi & Lo registers in MIPS

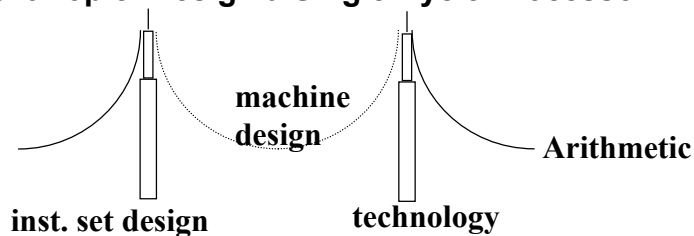
361 div.13

The Big Picture: Where are We Now?

- The Five Classic Components of a Computer



- Next Topic: Design a Single Cycle Processor

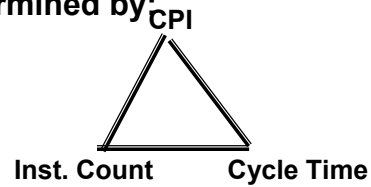


361 div.14

The Big Picture: The Performance Perspective

- ° Performance of a machine is determined by:

- Instruction count
- Clock cycle time
- Clock cycles per instruction



- ° Processor design (datapath and control) will determine:

- Clock cycle time
- Clock cycles per instruction

- ° Next Class:

- Single cycle processor:
 - Advantage: One clock cycle per instruction
 - Disadvantage: long cycle time

361 div.15

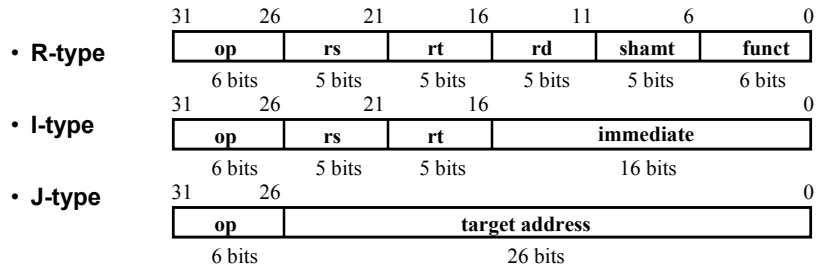
How to Design a Processor: step-by-step

- ° 1. Analyze instruction set => datapath requirements
 - the meaning of each instruction is given by the *register transfers*
 - datapath must include storage element for ISA registers
 - possibly more
 - datapath must support each register transfer
- ° 2. Select set of datapath components and establish clocking methodology
- ° 3. Assemble datapath meeting the requirements
- ° 4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.
- ° 5. Assemble the control logic

361 div.16

The MIPS Instruction Formats

- ° All MIPS instructions are 32 bits long. The three instruction formats:



- ° The different fields are:
 - op: operation of the instruction
 - rs, rt, rd: the source and destination register specifiers
 - shamt: shift amount
 - funct: selects the variant of the operation in the “op” field
 - address / immediate: address offset or immediate value
 - target address: target address of the jump instruction