Slide 1

Administrivia

- Reminder: Quiz 3 Wednesday.

Slide 2

Multiplication, Continued

- Recall basic strategy — same method as with base 10, but simpler because computing partial results is easier.

  This gives the textbook's first algorithm, figure 4.26. (Work through example.)

- Can then make improvements: First modify so we can use 32-bit rather than 64-bit addition (figure 4.29), then use 64-bit work area to hold product and multiplier both (figure 4.32). (Work through example.)

- What about signs? Last algorithm works, if we extend the sign bit when we shift right.

- A further improvement — "Booth's algorithm."

Slide 3

Multiplication, Continued

- Algorithms all involve shifts, adds, and some control/logic. What do we need in the way of hardware? See figure 4.31 for a sketch.

Slide 4

Multiplication, Continued

- In MIPS architecture, 64-bit product / work area is kept two special-purpose registers (`lo` and `hi`). Two instructions needed to do a multiplication and get the result:

  ```
  mult rs1, rs2
  mflo rdest
  ```

  Assembler provides a “pseudoinstruction”:

  ```
  mul rdest, rs1, rs2
  ```

- Notice, however, that a “smart” compiler might turn some multiplications into shifts. (Which ones?)

Slide 5

Division

- As with other arithmetic, first think through how we do this "by hand" in base 10. (Review terminology: We divide "dividend" a by "divisor" b to produce quotient q and remainder r, where $a = bq + r$ and $0 \leq |r| < b$.) Example?

  We can do the same thing in base 2; this gives the algorithm in figure 4.37. (Work through example.)

- Can then make improvements: First modify so we can do 32-bit rather than 64-bit arithmetic; then use 64-bit work area to hold both quotient and reminder (figure 4.40). (Work through example.)

- What about signs? Simplest solution is (they say!) to perform division on non-negative numbers and then fix up signs of the result if need be.

Slide 6

Division, Continued

- Hardware looks a lot like hardware for multiplication — figure 4.41.

Slide 7

Division, Continued

- In MIPS architecture, 64-bit work area for quotient and remainder is kept in same two special-purpose registers used for multiplication (`lo` and `hi`). After division, quotient is in `lo` and remainder is in `hi`. Two (or more) instructions needed to do a division and get the result:

```
div rs1, rs2
mflo rq
mfhi rr
```

  Assembler provides a "pseudoinstruction":

```
div rdest, rs1, rs2
```

- Notice, however, that a "smart" compiler might turn some divisions into shifts. (Which ones?)

Slide 8

Minute Essay

- What instruction would you use to multiply the number in register `$t0` by 16, treating it as an unsigned integer. Would you use the same instruction if you wanted to treat it as a signed integer?
- What instruction would you use to divide the number in register `$t0` by 16, treating it as an unsigned integer. Would you use the same instruction if you wanted to treat it as a signed integer?