Slide 1

Administrivia

- Reminder: Homework 3 written problems due today (6pm).

  Programming problems due Monday (11:59pm). Sorry about the delaying in getting them posted — I thought I had done so last week! Also, I just made a small tweak to the first problem, asking you to print the whole array `D` rather than just the supposedly-changed elements, and recommended that you use different registers for `i` and `j`.

  (I will try to leave time at the end of class for last-minute questions.)

- Quiz 3 rescheduled for Wednesday. Likely topics from early parts of Chapter 3.

Slide 2

Minute Essay From Last Lecture

- Many interesting answers; most were really pretty good. (I wouldn't have bet on this based on what I observe in the O/S course.) Time permitting, I'll reply to individual responses later, but for now . . .
- (Caveat: To some extent I'm speculating too, but it's somewhat well-informed speculation?)

Slide 3

O/S Versus Object File Contents/Formats

- The machine code part should depend only on the architecture (and 32-bit versus 64-bits counts as part of "the architecture"). But the same compiler running under different operating systems might make different choices?
- Format seems like it would be similar but not identical across operating systems, but if there's no real incentive to standardize maybe it hasn't happened.

Slide 4

O/S Versus Executable File Contents/Formats

- Part of what's in an executable file is whatever information is needed for the O/S to "launch" the program. Windows `.exe` files don't run under Linux, right? and ELF executables don't run under Windows? (What about WINE? well, it's an emulator, isn't it?)
- So for example consider references to shared library code — Windows DLLs versus UNIX "shared libraries" versus . . .).
- Also might matter whether the linker can assume that programs will always be loaded starting at the same address.

Slide 5

O/S and System Calls

- At the hardware / machine code level, what matters is the architecture — switch to "all privileges" mode (if such a thing exists), transfer control to fixed location, noting return address.
- But then what's at that fixed location is supposed to be O/S code, and what it does clearly could vary. For example, for SPIM, to echo a line to standard output you set particular values in some registers and do the `syscall` instruction. Other O/S's for MIPS would provide this functionality in other ways (probably also involving a `syscall` but with different conventions for register usage, e.g.).

Slide 6

What About . . .

- . . . support for parallelism? I say either the architecture supports it, or it doesn't, so for a given architecture any related instructions should execute the same on any O/S.

  Where it matters is most likely to be in O/S or library code to manage multiple threads (including assigning them to processing elements).
- . . . execution time? Actual CPU time should depend only on the implementation of the architecture (as discussed in Chapter 1), but total runtime will depend on many other things, among them things that depend on the O/S.

Slide 7

Numbers and Arithmetic — Review/Recap

- Most architectures these days represent integers as fixed-length two's complement binary quantities.
- Most architectures these days represent real numbers using one or more of the formats laid out by the IEEE 754 standard. Based on a base-2 version of scientific notation, plus special values for zero, plus/minus "infinity", and "not a number" (NaN).

  (Worth noting, though, that historically there have been architectures that could represent fractional quantities using base-10 "fixed-point" notation, and this may be coming back.)

  ("Floating point is strange" examples from CSCI 1120.)

Slide 8

Implementing Arithmetic — Preview

- In the next chapter we start talking about hardware design (though still at a somewhat abstract level).
- For now it may be useful to know that the low-level building blocks are entities that can evaluate Boolean expressions — very simple ones at the lowest level, and slightly more complex ones one level up.
- So for example we can implement addition by first making a "one-bit adder" that maps three inputs (two operands and carry-in) to two outputs (result and carry-out), and then chaining together 32 of them.
- Multiplication and division, however, may need to be more complex, involving multiple steps and control-flow logic. (Historical(?) aside: Early implementations may have just done the simple dumb thing — repeated additions or subtractions. (!))

Slide 9

Integer Addition, Subtraction, and Negative Values

- Recall(?) how addition works — right to left with carry. Carry-in to rightmost bit is (of course?) 0.
- Recall(?) also how finding the negative of a number works — "flip all the bits" and add 1.
- Notice then how if we can build an adder, we can more or less get subtraction "for free" — compute $a - b$ by adding a and bitwise negation of b with a carry-in to the rightmost bit of 1. (This is one reason two's complement notation is attractive!)

Slide 10

Multiplication

- As with addition, first think through how we do this "by hand" in base 10. (Review terminology: In $a \times b$, call a the "multiplicand" and b the "multiplier".) Example?
- We can do the same thing in base 2, but it's simpler, no? computing the partial results is easier. This gives the textbook's first algorithm, shown in figures 3.3 through 3.6. (Work through example.)

  Notice also that overflow could be a lot worse here — so normally we'll compute a result twice as big as the inputs.

  (We can do better — later maybe.)
- What about signs? Algorithm works, if we extend the sign bit when we shift right.

Slide 11

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 12

Minute Essay

- (As usual?) Anything noteworthy about the homework due today? interesting, difficult, . . . ?