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.

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?)

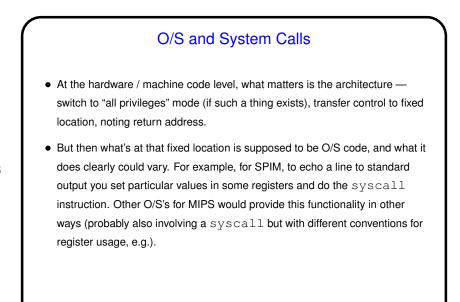


- The machine code part should depend only on the architecture (and 32-bit versus 64-bits counts as part of "the architecture"). But the same compiler running under different operating systems might make different choices?
- Format seems like it would be similar but not identical across operating systems,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

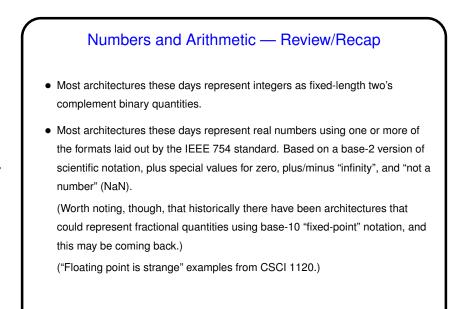
O/S Versus Executable File Contents/Formats

 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.

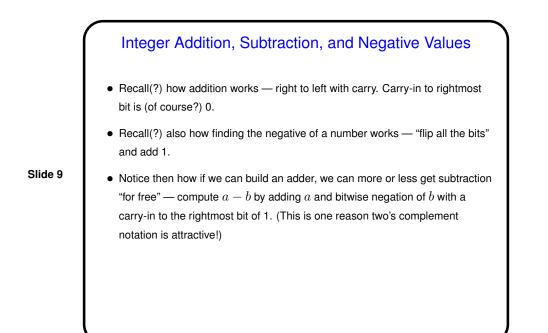


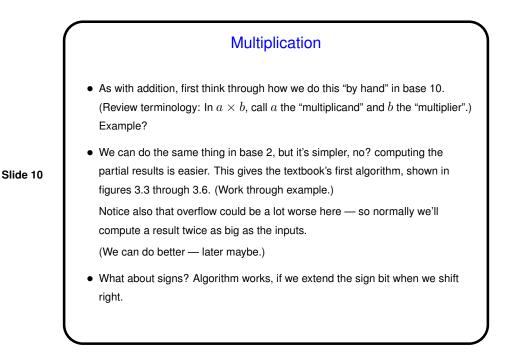
What About ...
...support for parallelism? I say either the architecture supports it, or it doesn't, so for a given architecture any related instructions should execute the same on any O/S.
Where it matters is most likely to be in O/S or library code to manage multiple threads (including assigning them 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.

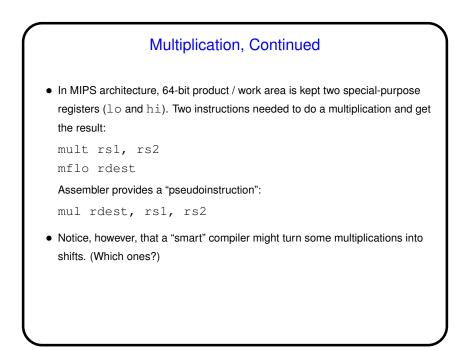


Implementing Arithmetic — Preview
In the next chapter we start talking about hardware design (though still at a somewhat abstract level).
For now it may be useful to know that the low-level building blocks are entities that can evaluate Boolean expressions — very simple ones at the lowest level, and 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 7







Minute Essay • (As usual?) Anything noteworthy about the homework due today? interesting, difficult, ...? Slide 12