Slide 1

Administrivia

- Many people did not turn in Homework 5 and/or Homework 6. Not too late to get *some* points, if you have not looked at a solution.
- I'm grading Homework 6 and will send out a grade summary when done (this week sometime).
- Homework 7 due 12/14. Notice that the deadline is a real one. Should there be a help session? during the reading days?

Slide 2

Data Representation — "It's All Ones and Zeros"

- At the hardware level, all data is represented in binary form — ones and zeros. (Why? hardware for that is simpler to build.)
- How then do we represent various kinds of data? First a short review of binary numbers . . .

Slide 3

Binary Numbers

- Humans usually use the decimal (base 10) number system, but other (positive integer) bases work too. (Well, maybe not base 1.)
- In base 10, there are ten possible digits, with values 0 through 9.
  In base 2, there are 2 possible digits ("bits"), with values 0 and 1.
- Everything in base 2 works the same as base 10, if you think about how base 10 actually works, so to speak.

Slide 4

Computer Representation of Integers

- So now you can probably guess how non-negative integers can be represented using ones and zeros — number in binary. Fixed size (so we can only represent a limited range).
- How about negative numbers, though? No way to directly represent plus/minus. Various schemes are possible. The one most used now is *two's complement*: Motivated by the idea that it would be nice if the way we add numbers doesn't depend on their sign. So first let's talk about addition . . .

Slide 5

Machine Arithmetic — Integer Addition and Negative Numbers

- Adding binary numbers works just like adding base-10 numbers — work from right to left, carry as needed. (Example.)
- Two's complement representation of negative numbers is chosen so that we easily get 0 when we add $-n$ and n.

  Computing $-n$ is easy with a simple trick: If m is the number of bits we're using, addition is in effect modulo 2^m. So $-n$ is equivalent to $2^m - n$, which we can compute as $((2^m - 1) - n) + 1)$.
- So now we can easily (?) do subtraction too — to compute $a - b$, compute $-b$ and add.

Slide 6

Binary Fractions

- We talked about integer binary numbers. How would we represent fractions?
- With base-10 numbers, the digits after the decimal point represent negative powers of 10. Same idea works in binary.

Slide 7

Computer Representation of Real Numbers

- How are non-integer numbers represented? usually as *floating point*.
- Idea is similar to scientific notation — represent number as a binary fraction multiplied by a power of 2:

$$x = (-1)^{sign} \times (1 + frac) \times 2^{bias+exp}$$

  and then store $sign$ $frac$, and exp. Sign is one bit; number of bits for the other two fields varies — e.g., for usual single-precision, 8 bits for exponent and 23 for fraction. Bias is chosen to allow roughly equal numbers of positive and negative exponents.
- Current most common format — "IEEE 754".

Slide 8

Numbers in Math Versus Numbers in Programming

- The integers and real numbers of the idealized world of math have some properties not completely shared by their computer representations.
- Math integers can be any size; computer integers can't.
- Math real numbers can be any size and precision; floating-point numbers can't. Also, some quantities that can be represented easily in decimal can't be represented in binary.
- Math operations on integers and reals have properties such as associativity that don't necessarily hold for the computer representations. (Yes, really!)
- (Two "floating point is strange" examples.)

Slide 9

Course Topics — Recap

- Basic C programming, for people who already know how to write programs in some other language. Especially useful (I think!) for those who start in a very abstract/high-level language.
- Review of the Linux/UNIX command-line environment and command-line development tools.
- Review of basics of computer arithmetic and data representation. A little more about floating-point representation.

Slide 10

Why Learn C? (For Java/Python/Scala Programmers — Recap)

- Scala and Python (and Java, less so) provide a programming environment that's nice in many ways — lots of safety checks, nice features, extensive standard library. But they hide a lot about how hardware actually works.
- C, in contrast, has been called "high-level assembly language" — so it seems primitive in some ways compared to many other languages. What you get (we think!) in return for the annoyances is more understanding of hardware — and if you do low-level work (e.g., operating systems, embedded systems), it may well be in C. (Performance *may* also be better, though "measure and be sure".)

Slide 11

Quotes of the Day/Week/?

- From a key figure in the early days of computing:

  "As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent finding mistakes in my own programs." (Maurice Wilkes: 1948)

- From someone in a discussion group for the Java programming language:

  "Compilers aren't friendly to anybody. They are heartless nitpickers that enjoy telling you about all your mistakes. The best one can do is to satisfy their pedantry to keep them quiet :)"

Slide 12

Minute Essay

- None — sign in. (But also tell me if you're interested in review/help session and if so when during the reading days you could be available.)