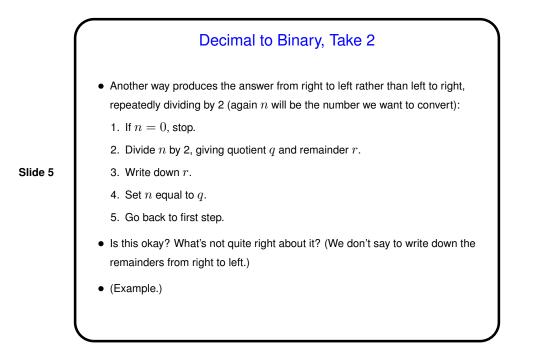
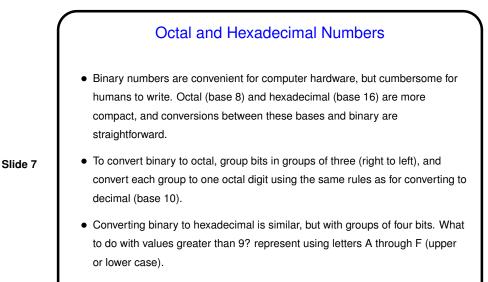


Decimal to Binary, Take 1• One way is to first find the highest power of 2 smaller than or equal to the
number, write that down, subtract it from the number, and continue:1. If n = 0, stop.2. Find largest p such that $2^p \le n$.3. Write a 1 in the p-th output position.4. Subtract 2^p from n.5. Go back to first step.• Is this okay? What's not quite right about it? (We don't say what to put in the
positions that don't have ones in them.)• (Example.)



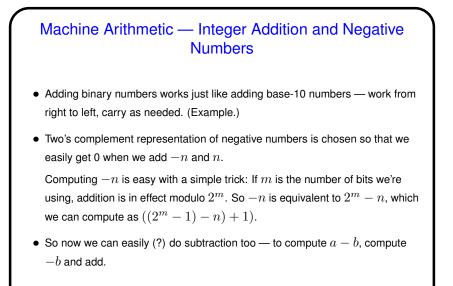
Recap Key ideas here — break problem down into a sequence of steps that we hope don't require much intelligence, just an ability to calculate, with some decision-making and repeating. Before moving back to programming and C, a little more about different number bases and how binary numbers are used to represent data ...



• (Examples.)

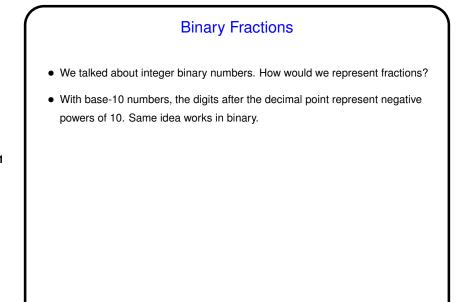
Computer Representation of Integers

- Computers represent everything in terms of ones and zeros. For non-negative integers, you can probably guess how this works — number in binary. Fixed size (so we can only represent a limited range).
- Slide 8
- 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 didn't depend on their sign. So first let's talk about addition ...



Machine Arithmetic — Integer Multiplication and Division

- Hardware to multiply and divide basically follows the procedures humans can do on paper - multiply by computing and adding "partial sums", divide via long division.
- (Details can get a little tricky, but basic idea is straightforward extrapolation from how it works in base 10.)



Computer Representation of Real Numbers

- How are non-integer numbers represented? usually as *floating point*. "IEEE 754 standard" spells out details; most current hardware implements it.
- Idea is similar to scientific notation represent number as a binary fraction multiplied by a power of 2:

Slide 12

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

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.



- 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.
- Slide 13
- 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 exactly in binary.
- Math operations on integers and reals have properties such as associativity that don't necessarily hold for the computer representations. (Yes, really!)

C and Representing Numbers — Integers

- Computer hardware typically represents integers as a fixed number of binary digits, using "two's complement" idea to allow for representing negative numbers.
- Slide 14
- C, like many (but not all!) programming languages bases its notion of integer data on this, but also has a notion of different types with different sizes. (A little more about this next time.)

