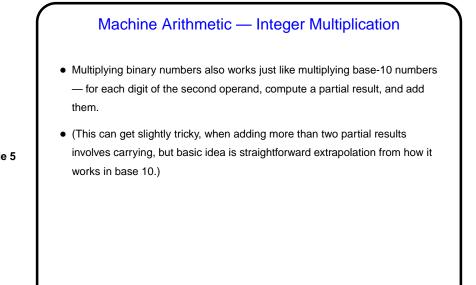
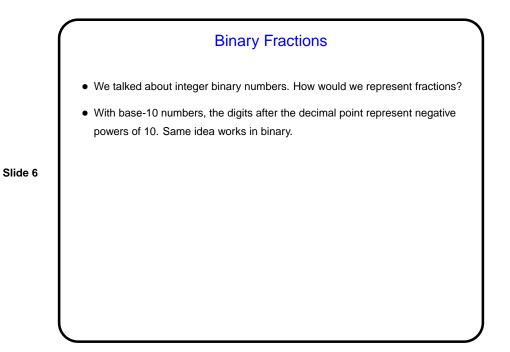
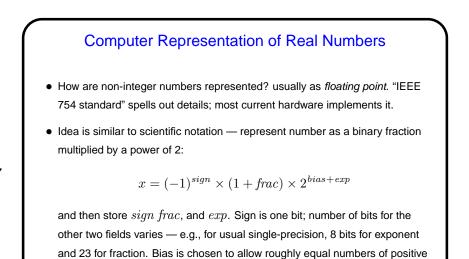


Machine Arithmetic — Integer Addition and Negative Numbers
Adding binary numbers works just like adding base-10 numbers — work from right to left, carry as needed. (Example.)
Two's complement representation of negative numbers is chosen so that we easily get 0 when we add -n and n. Computing -n is easy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





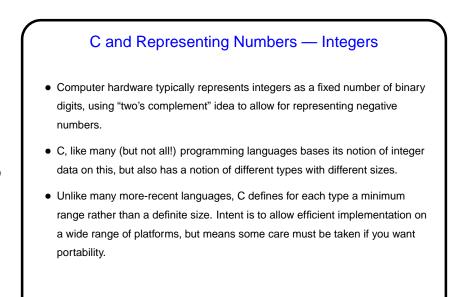


Numbers in Math Versus Numbers in Programming

- 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.

and negative exponents.

- 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 — Real Numbers

- Hardware also typically supports "floating-point" numbers, with a representation based on a base-2 version of scientific notation. This allows representing not only fractional quantities but also allows representing larger numbers than would be possible with fixed-length integers. Notice that only fractions that can be written with a denominator that's a power of two can be represented exactly.
- Again C goes along with this and provides different "sizes" (float and double). As with integers, exact sizes not specified, only minimum criteria.

