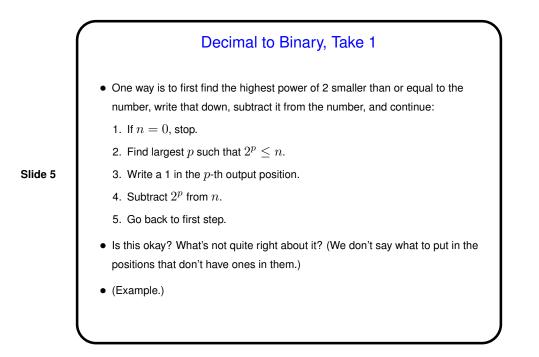


Slide 4

Converting Between Bases
Converting from another base to base 10 is easy if tedious (just use definition).
Converting from base 10 to another base? Let's try to develop an "algorithm" (procedure) for that ...

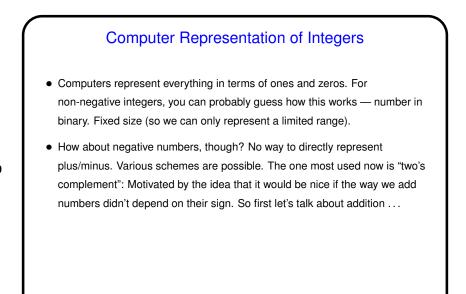


Decimal to Binary, Take 2 Another way produces the answer from right to left rather than left to right, repeatedly dividing by 2 (again n will be the number we want to convert): If n = 0, stop. Divide n by 2, giving quotient q and remainder r. Write down r. Set n equal to q. Go back to first step. Is this okay? What's not quite right about it? (We don't say to write down the remainders from right to left.) (Example.)



- 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 ...

Octal and Hexadecimal Numbers
Binary numbers are convenient for computer hardware, but cumbersome for humans to write. Octal (base 8) and hexadecimal (base 16) are more compact, and conversions between these bases and binary are straightforward.
Slide 8
To convert binary to octal, group bits in groups of three (right to left), and convert each group to one octal digit using the same rules as for converting to decimal (base 10).
Converting binary to hexadecimal is similar, but with groups of four bits. What to do with values greater than 9? represent using letters A through F (upper or lower case).
(Examples.)



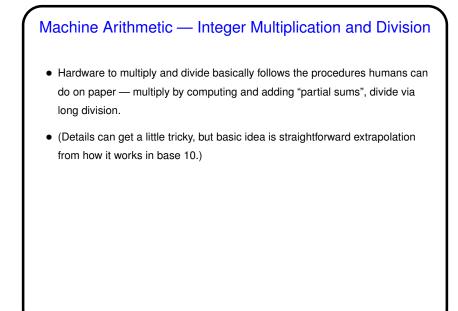
Machine Arithmetic — Integer Addition and Negative Numbers

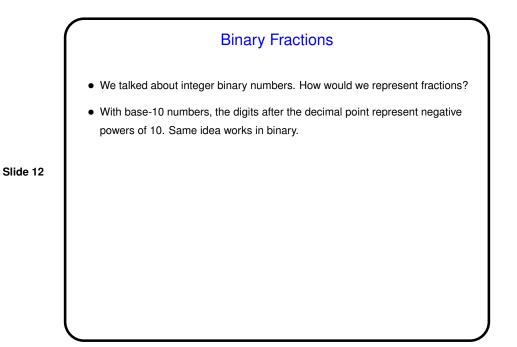
- Adding binary numbers works just like adding base-10 numbers work from right to left, carry as needed. (Example.)
- Two's complement representation of negative numbers is chosen so that we easily get 0 when we add -n and n.

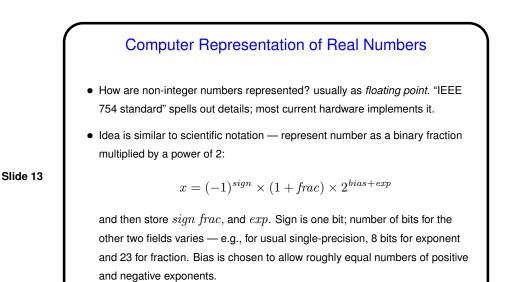
Computing -n is easy with a simple trick: If m is the number of bits we're using, addition is in effect modulo 2^m . So -n is equivalent to $2^m - n$, which we can compute as $((2^m - 1) - n) + 1)$.

• So now we can easily (?) do subtraction too — to compute a-b, compute -b and add.

Slide 9

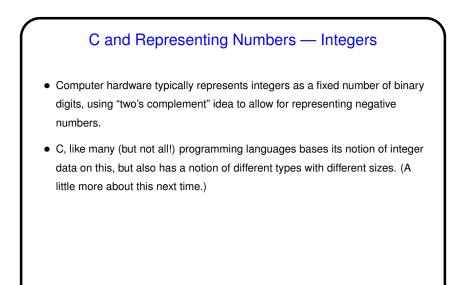






Numbers in Math Versus Numbers in Programming

- 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 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).

