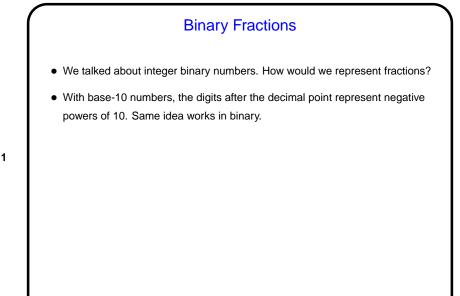




- So now you can probably guess how non-negative integers can be represented using ones and zeros — number in binary. Fixed size (so we can only represent a limited range).
- How about negative numbers, though? No way to directly represent plus/minus. Various schemes are possible. The one most used now 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 ...

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.



Computer Representation of Real Numbers

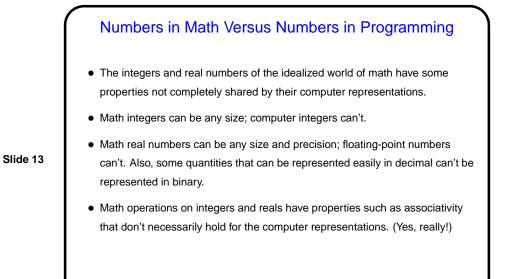
- How are non-integer numbers represented? usually as floating point.
- Idea is similar to scientific notation represent number as a binary fraction multiplied by a power of 2:

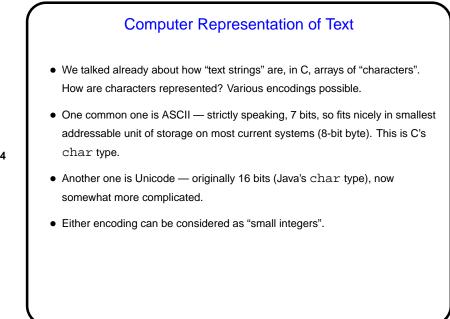
Slide 12

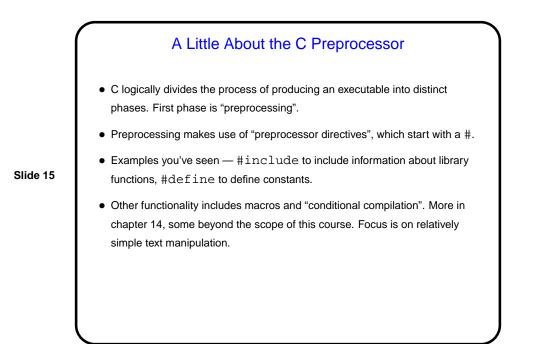
$$x = (-1)^{sign} \times (1 + frac) \times 2^{bias + exp}$$

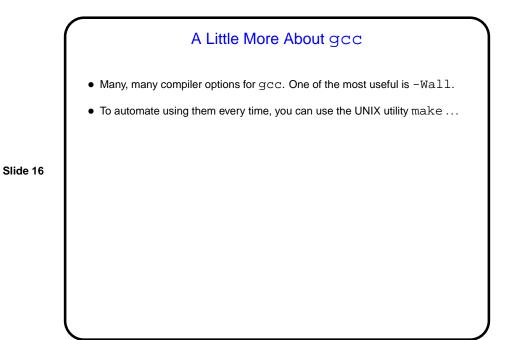
and then store $sign \ frac$, and exp. Sign is one bit; number of bits for the other two fields varies — e.g., for usual single-precision, 8 bits for exponent and 23 for fraction. Bias is chosen to allow roughly equal numbers of positive and negative exponents.

• Current most common format — "IEEE 754".











 Motivation: Most programming languages allow you to compile programs in pieces ("separate compilation"). This makes sense when working on a large program — when you change something, just recompile parts that are affected.

Slide 17

• Idea behind make — have computer figure out what needs to be recompiled and issue right commands to recompile it.

Makefiles

• First step in using make is to set up "makefile" describing how files that make up your program (source, object, executable, etc.) depend on each other and how to update the ones that are generated from others. Normally call this file Makefile or makefile.

Slide 18

• When you type make, make figures out (based on files' timestamps) which files need to be recreated and how to recreate them.

Simple example on sample programs page.

