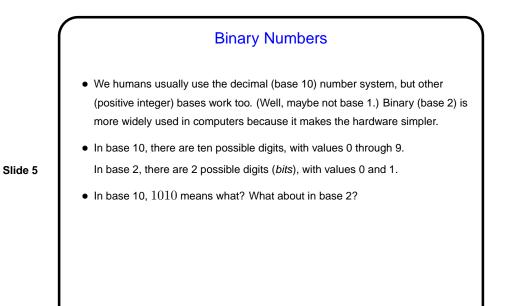
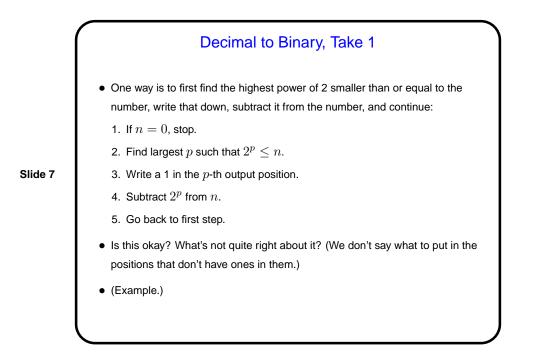
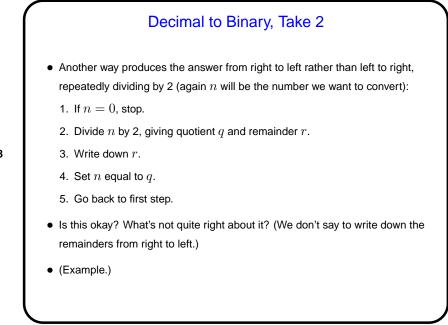


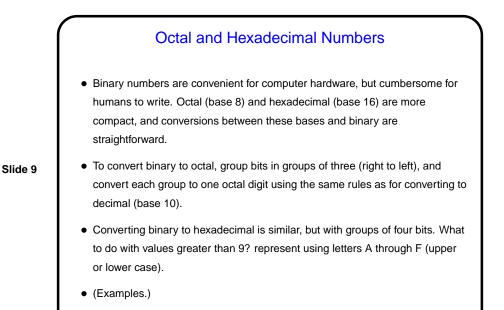
## Numeric Literals and Expressions should be fairly familiar. Notice that Scala (like many programming languages) makes a distinction between integers and numbers that have (or might have) a fractional part. Use the interpreter's REPL to try out things. Some things may be surprising — integer division, large numbers, calculations using fractions. To understand some of these it helps to know how the computer represents numbers.



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 ...



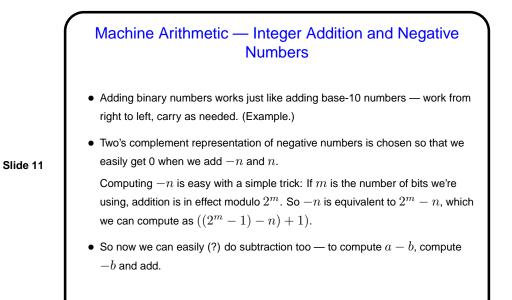




**Computer Representation of Integers** 

- 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).
- 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 ...

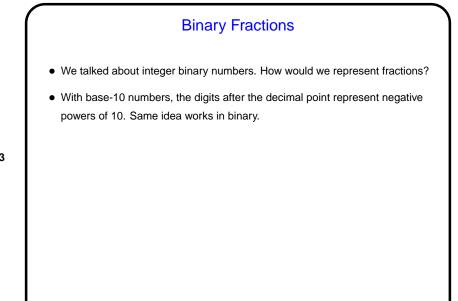




Machine Arithmetic — Integer Multiplication

Multiplying binary numbers also works just like multiplying base-10 numbers
 — for each digit of the second operand, compute a partial result, and add them.

• (This can get tricky, when adding more than two partial results involves carrying.)



Slide 13

## **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 14

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

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.

