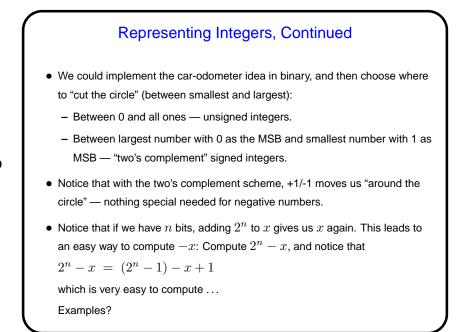


Representing Integers (Review?)
Representing non-negative integers is easy — convert to binary and pad on the left with zeros.
What about negative integers?
Could try using one bit for sign, but then you have +0 and -0, and there are other complications.
Or ... consider a car odometer — in effect, representable numbers form a circle, since adding 1 to largest number yields 0.

Slide 7



Slide 10
Slide 10
Example: Suppose we have ox0000000 in \$t0 oxffffff2 in \$t1 What happens if we execute slt \$t2, \$t0, \$t1? What happens if we execute sltu \$t2, \$t0, \$t1? (Same bits, different interpretations!)



For non-negative numbers, easy.

For negative numbers, also not too hard — consider taking absolute value, extending it, then taking negative again.

Slide 11

- In effect "extend" by duplicating sign bit.
- (Notice that not all instructions that include a 16-bit constant do this.)

## Two's Complement and Addition/Subtraction

- Addition in binary works much like addition in decimal (taking into account the different bases). Notice what happens if one number is negative. (Try an example or two.)
- Subtraction could also be done the way we do in decimal. Or how else could we do it? (Again, try some examples.)
- But there is one catch, related to the fact that operands and result are all *n* bits. What is it?

