Why integer overflow "wraps around"

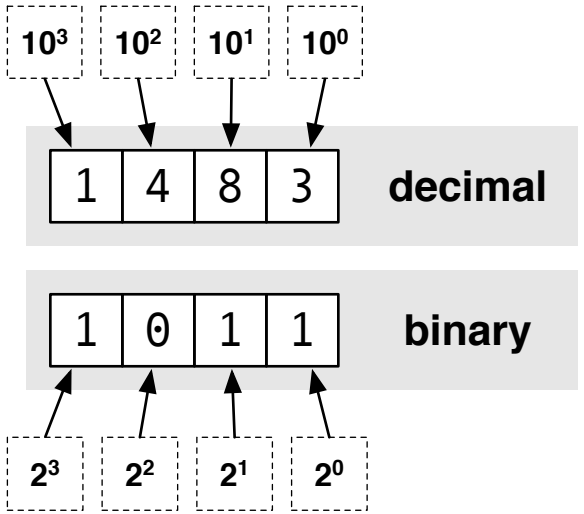
handout for CS 302 by Will Benton (willb@cs)

The whole-number types in Java (e.g. `int`, `long`, etc.) have limited ranges. Of course, any type representable in a finite computer has a limited range, but you're likely to actually run into the limits of the Java primitive types before you have to deal with a number that you can't represent on a computer.

Manipulating a variable so that its value would exceed the range of its type results in *integer overflow*. When this happens, the value of the variable "*wraps around*" to the opposite end of the range, just as a classic video game character might wrap around to the other side of the screen upon crossing an edge.

This handout will explain why this happens. To do so, we'll first need to consider how computers represent numbers. (If you find this interesting, you'll enjoy a computer organization class!)

Computers represent numbers in *binary*, or base-2. Humans typically deal with numbers in *decimal*, or base-10. Both kinds of numbers have "places," in which a digit denotes some quantity of a power of the base. Let's examine two different four-digit numbers to see the difference:



The decimal number has the value 1483: $1 * 10^3 + 4 * 10^2 + 8 * 10^1 + 3 * 10^0$. The binary number has the value 11: $1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$.

Note that with n bits, you can represent a number up to $b^n - 1$, where b is the base.