Lecture 4b: Binary Arithmetic

CS105: Great Insights in Computer Science
Michael L. Littman, Fall 2006

Subtraction

```
 011 1 01   01
 100 1 00 1 01
-10110010
  1110011
```

- As in decimal, proceed right to left, borrowing if not doing so would force us to subtract a bigger number from a smaller one.

Overflow

- When working with numbers made of a fixed number of bits, carries can "overflow", meaning we might not be able to represent the full sum. Example (8 bits):

```
  01010011
 +10110010
(1)00000101
```

Negation

- Overflow provides an interesting way to think of negation.
- Recall in algebra, an additive inverse of x is the number y such that $x+y = 0$. So, $y = -x$.

```
 01111111 1
 ~~100000000~~      10101101
-01010011          +01010011
 10101101         (1)00000000
```

Two's Complement

- To find the negation of a number, flip all the bits, then add one:

```
 01010011   83
 10101100  172 = 255−83
+00000001
 10101101  173 = 256−83 = "−83"
```

Subtraction as Negate/Add

- Combining these ideas, we can subtract one number from another by taking the two's complement and adding!

Multiplication

- Of course, multiplication can be carried out by repeated addition, but it's a very inefficient way to go with big numbers.
- Our standard grade-school approach to multiplication carries forward to binary numbers as well.

Multiplication

- Boils down to:
  - copy, shift, add

```
   10101101
 x 01010011
   10101101
  10101101
 10101101
+10101101
 11100000010111
```