Notes for 9/9/09

Two's complement:
A negative of a number is a large power of two minus that number

For example, with a 7-bit machine:
0100011 = 35
1011101 = 2^8 – abs(x) = -35

Given n bits, you can represent...
-2^{n-1} to $2^{n-1} - 1$

Two's complement is different from one's complement because two's complement has only one representation of zero. Also, with two's complement, the circuitry to implement the arithmetic of the signed is the same as the circuity to implementation of the arithmetic of the unsigned. Comparisons, such as less than and greater than, are still different between signed and unsigned numbers.

Addition and subtraction:

```
 20   0011110 To subtract, just add the negative
-30  +1100001 Note: to go to two's complement, flip every bit and then
            1       add one
---   -------
-10   1100010

       1111   <- carry bits
 23   0010111
+14  +0001110
---   -------
 37   0100101
```

With a ripple carry adder, you have to wait for the carry bit for each bit addition, even if there isn't one. With a carry lookahead adder, you can look ahead to see if the next bit addition can propogate a carry bit.

```
    1     1
 97   1100001 Overflow.  In C, the extra carry bit is discarded.
+81  +1010001
---   -------
      0110010
```

In MIPS assembly:

add, addi, sub	Create overflow exception
addu, addiu, subu	No exception (these are used in C)

Difference between exception and interrupt:
We have seen overflow exceptions (above), system calls as in Lab 1, and segmentation faults—these are all exceptions.
Interrupts come from outside the processor. For instance, if you request data from the hard drive, when

the data is ready, the hard drive controller will interrupt the processor. A timer can interrupt the processor at a scheduled time.

So, as a general rule, exceptions come from inside the processor, and interrupts come from outside.

Multiplication:

```
     1000 As in decimal, just multiply each bit and shift.
    *1001
    -----
     1000
    0000
   0000
 +1000
 --------
```

With multiplication, you can have up to 32 bits of overflow. MIPS has two registers that are not architecturally visible, but, when you multiply, the result goes into $hi and $lo. To get the result out of these, you can use the mfhi and mflo instructions.

Note that signed and unsigned multiplication are different.

In MIPS assembly:

mul, div	Signed multiplication and division
mulu, divu	Unsigned multiplication and division

Division can create an exception when you divide by zero. On Linux, this will crash your program, unless you have overridden the default divide-by-zero signal handler.

Floating point representation:
$(-1)^{\text{sign}} * (1 + \text{fraction}) * 2^{\text{exponent}}$

If both fraction and exponent are zero, then this is interpreted as zero.

There are special representations for infinity, negative infinity, and NaN. NaN is for indeterminate operations, like 0.0/0.0.

When you overflow the fraction, the exponent becomes larger. When you overflow the exponent, the value becomes infinity. Note that there are no exceptions in floating point. Underflow is when the exponent becomes too small, and then the value becomes zero.