Computer Arithmetic

2-17-2003

Opening Discussion

- What did we talk about last class?
- Have you seen anything interesting in the news?

Tracing Recursion

- Before we leave the topic of MIPS assembly let's take a few minutes to look at a little recursive function to do factorials and trace what happens in memory and with the stack as it runs.

Moving On

- We are done down with our direct discussion of the MIPS assembly language. That doesn't mean that you are done using it though. We have laid the foundation for low level problem solving with it and now you will mainly just be using that.
- Starting today we look more at the details of what the machine is doing to bring our code to life.

Binary in MIPS

- As you are well aware, the MIPS architecture uses 32-bit words. This puts a very real restriction on the sizes of numbers that it can represent. This is one of the most significant differences between computer arithmetic and true math.
- If we are just interested in representing natural numbers this gives us numbers between 0 and 2^{32}-1 (4,294,967,295).

Overflow

- If an operation tries to produce a number that doesn't fit in this range, it is called an overflow and bits are lost. In this chapter we will be talking about commands that help us deal with this if it happens.
- Fortunately, it is something that we can stay clear of a lot of the time because 4 billion is a reasonably large number. It was much harder when an int only went to 64k.

What are Negatives?

- Computers also need to be able to deal with negative values. One approach to this is to use sign and magnitude. Here we reserve one bit for the sign and the others are the magnitude. This has a number of drawbacks.
- To come up with another approach we should think about what it means for a number to be negative and use that definition to help us.

Two's Compliment

- Our definition helps us come up with an alternate definition of negative binary numbers for our machine. We want it so that a+(-a)=0.
- Negative numbers have the largest bit on and leading ones. This effectively makes the 2^{31} bit negative and all the others positive.
- When loading bytes we might have to add leading 1s to preserve a negative value.

Signed and Unsigned

- Not all numbers need to be signed. Memory addresses in particular are much better unsigned. Because of this we have to introduce some new comparison instructions to do comparisons that are specifically unsigned.
- slt -> sltu
- slti -> sltiu

Hexadecimal Numbers

- Writing out a bunch of zeros and ones is tedious to say the least so we would like to have something that doesn't take as many characters, but that can be converted to binary more quickly than decimal can.
- Using base 16 we can put bits into groups of four and do the quick conversion to the characters 0-F.

Minute Essay

- Write the number 153 in 16 bit binary, then show its negative and how it would be represented in hex.
- Quiz #3 is at the beginning of next class.