MIPS Machine Assembly Language

2-7-2003

Opening Discussion

- What did we talk about last class?
- Have you seen anything interesting in the news?
- Do you have any questions about the graded quizzes?

Powers of 2 Loop

- Last class I asked you to write assembly for a loop that basically calculates powers of two.

```
LoopTop:
    add $s1, $s1, $s1
    sub $t0, $t0, 1
    beq $t0, $zero, LoopTop
```

Jump Statements

- In addition to the conditional branch statements beq and bne, there are non-conditional jumps.
- j *label* # always jumps to the given label.
- jr *register* # always jumps to the address for that register.
- These are helpful when the flow can't just be linear.

Basic Blocks

- This is a term that you see any time you are talking about program analysis with any language. These are blocks of code that always execute sequentially from top to bottom.
- They have no entry points in the middle and any branches or jumps are at the end.

Building Conditionals

- MIPS compilers build all of their conditions out of beq, bne, and slt with the help of the fixed register $zero.
- Obviously we can check for equality, in equality, and less than. Greater than just causes us to reverse the order of the arguments from less than.
- Who can we get something like less than or equal to?

Structures in MIPS Assembly

- Let's take a look at how we build each of the different control structures we would use in C/C++ in MIPS assembly.
- if/else
- do/while
- while

Introduction to SPIM

- Now let's take a quick look at the xspim program and try to write and run the Fibonacci numbers program.

Minute Essay

- Write assembly language to do the following C++ code. You can assume that some registers are attached to variables.

```
int a=0;
for(int j=6; j>0; j++) {
     a+=j;
}
```

- Don't leave without turning in assignment #2.