CSE 378 Spring 2003

Machine Organization and Assembly Language Programming

Problem Set #6

Due: Friday May 23rd

In this assignment you'll modify your single cycle implementation of the MIPS machine that you did for Assignment #5 and build a pipeline implementation. You'll have to implement the same instructions and run the same .smokmem file (factorial).

You can work in teams of two if you so desire.

In essence what you need to do is have a Smok implementation resembling the pipeline in Figure 6.51 of your book. However, there are a few differences such as:

- In the book, target address computation and checking whether a branch is taken or not is done during the decode stage. The problem with that is that it puts more onus on the forwarding unit that was is shown in the figure. So while you can perform the target address computation at that stage, we strongly recommend that the decision to branch or not be done during the EX or MEM stage.
- In the book, and as discussed in class, we have been talking about writes to the register file during the first half of the clock cycle and reading during the second half. Unfortunately, Smok does not work that way. Therefore you'll need to expand the Forwarding unit so that there can be forwarding from the WB stage also.
- You can simplify the design of the Hazard detection unit for load dependencies as follows. You can decide to stall for one cycle if instruction i is a load and instruction $i+1$'s bits 25-21 or bits 20-16 match the $rt field of the load instruction (this means that you might stall more often than necessary but it simplifies the control).

We strongly recommend that you take a progressive approach in testing your design. For example:

- Start with a single instruction and see that it flows correctly through the pipeline.
- Test two instructions without dependencies
- Test two arithmetic instructions with dependencies to check the forwarding unit
- Test several arithmetic instructions with dependencies
- Test a load dependency case
- Check that your branch logic does the right thing
- etc...
- Run the factorial on a small example (e.g., factorial of 2)
- Test the whole program

What you have to **turnin**.

- a README file indicating briefly how to "operate" your machine in case it is not obvious, i.e., which files to load etc.. In case you are aware of bugs and did not have time to correct them, indicate it in the README file.

- a .smok file for your implementation (and any .smokcont for containers), a .smokpla file for the PLA if any, a .smokmem for the memory.
- More specifically, send e-mail to Tapan (tapan@cs) with subject 378ASS6, including files LASTNAME.README, LASTNAME.smok, LASTNAME.smokmem, and LASTNAME.smokpla. Also LASTNAME-ComponentType.smokcont for any required containers.

One turnin per set of partners is enough but be sure to indicate both names in the README file.