Hints for Homework 10, EE 361 Fall 2003, University of Hawaii
November 7, 2003

You will be building the single cycle MIPS processor in verilog. Actually, the verilog module that you build is the processor excluding the Instruction Memory named "Mips". This is illustrated in Figure 1.

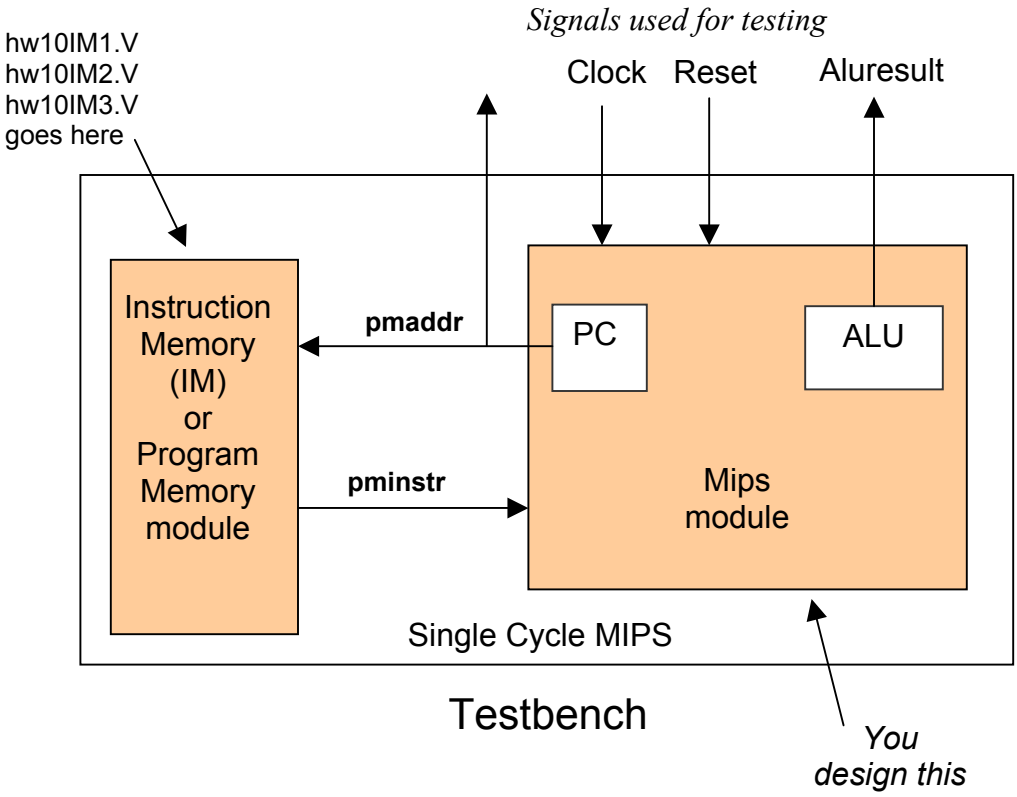

Figure 1. Block diagram of project.

When you turn in your projects, the IM should be hw10IM1.V, hw10IM2.V, or hw10IM3.V. The testbench file should be hw10Main.V. These files can be found at the homework web site. You can also download a version of the register file at the web site. Caution: some of these files may be buggy. :)

Note that these files are used for grading and not necessarily for debugging and testing. To test and debug the Mips module, you may design and use your own instruction memories (with programs you choose) and testbenches. Often you should display more internal signals to find bugs.

The following is a suggestion of how to proceed with the first project. For the first project, the IM has a program that has only R-type instructions (add, and, or, slt, sub). Now we need a plan to build the MIPS which should includes how we will test the

circuit. To simplify, we can initially eliminate "loops" in the circuit, and then add them in later.

Notice that there are two loops of circuitry. The top loop updates the program counter, and the bottom loop processes data. The top loop is easier to check since we only expect the PC to be incremented by 4, so we'll check that first. We'll break up the bottom loop, as shown in Figure 2, by fixing the Writedata input of the register file to 1.

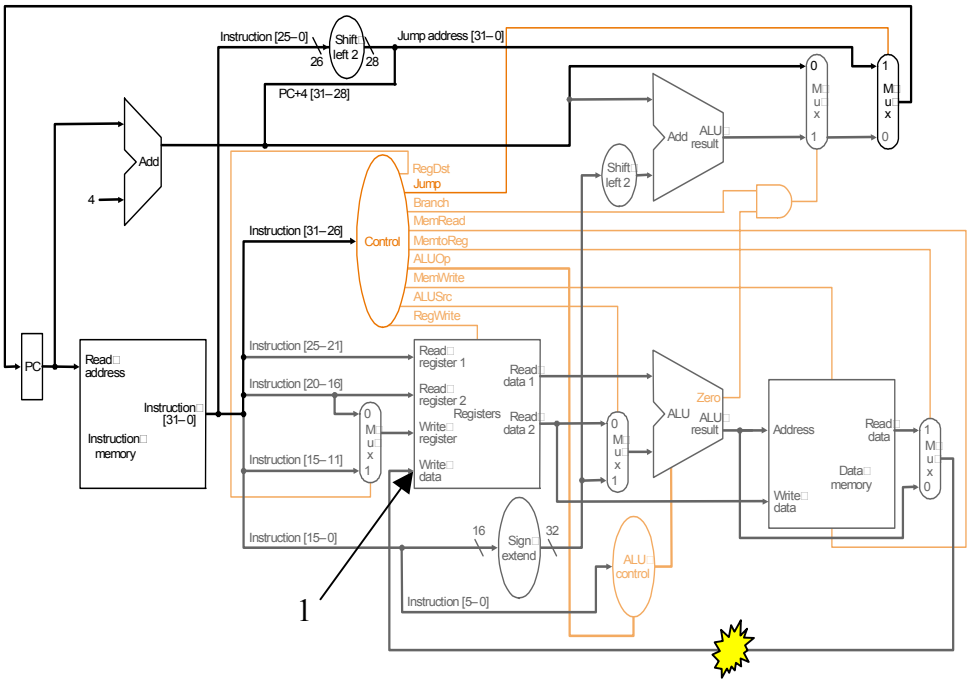

Figure 2. Simplified MIPS

Let your test bench run so that the processor resets, and then stop the simulation a time unit after that. The PC value should be 0. Check the output of the instruction memory and the controller. Make sure all the outputs are correct. Check if the input to the PC is PC+4. To do all this checking, you need display statements in your verilog code.

Now check the bottom loop of the circuit. Make sure the outputs of the register file and ALU are correct (recall that the register file has been initialized to certain values). Make sure the output of the MemtoReg multiplexer is equal to the output of the ALU.

Now have the simulation run a little longer so that the first instruction is complete. Check if the value "1" (at the Writedata input of the register file) was written into the appropriate register. If everything works then reconnect the bottom loop. Then simulate the first instruction. If that works then simulate up to the next instruction and so forth. Break up the circuit if necessary to find bugs.