EE108B-Kyoto (Spr 2011-2012)

Lab 2

Processor Datapath Design

Due: Friday April 27th

**Introduction:**

Now that you have seen some of the benefits of the software approach to problems, we will spend the next three labs building a processor that is capable of executing MIPS instructions. In this lab, you will complete an implementation of a single cycle processor.

**Requirements:**

You will be completing the MIPS processor implementation using the starter files provided to you. This will involve editing parts of the instruction fetch unit, instruction decoder, and ALU. You will be required to:

1. Write testbenches that thoroughly demonstrate the correct execution of each instruction type.
2. Show and explain screenshots of your testbenches demonstrating correct functionality of all instruction types.

**Implementation Details:**

*Processor Model:*

Before you start coding, look at each of the modules we have provided and try to understand what each one of them does. Ask one of the TAs if you have any questions about the model.

* MIPS.v - top level module
* MIPStest.v – testbench (you may add more of these if you wish or you may write all test code here)
* IF.v - fetches the current instruction from memory
* Regfile.v – reads the two input registers and writes to the output register
* Decode.v – decode the instruction and determine control signals for the rest of the processor
* ALU.v – datapath for arithmetic/logic operations
* MemStage.v – interface with data memory and memory-mapped devices and determine what data should be written back into the register file
* A few other files that you should not need to worry about

*Getting Started:*

You will primarily be using ModelSim for this project. Before using ModelSim, you need to execute the following command from your UNIX terminal window in your home directory. This has to be done only once for all labs and is used to setup your environment to run the tool.

echo source /usr/class/ee108b/ee108.cshrc >> .cshrc

ModelSim is aliased to vsim, so you can start ModelSim by typing “vsim &” into the terminal.

**Verilog Code:**

Most of your work should be in the IF.v, Decode.v, and ALU.v modules. Indeed it is possible to implement the lab while modifying only the files above and without adding any new registers or wires. However, you are free to modify any files that you want in any way that you like, but you **must** document every file that you modify and the changes you make.

The processor we have provided will increment the program counter by 4 after each instruction, but does not allow for branches or jumps. In IF.v, calculate the program counter in the event of jumps or branches, and determine how to assign the next PC.

In Decode.v, you will have to determine which ALU operation should be performed for each instruction listed below. In many cases, the answer will be obvious (for the ADD instruction, select the add operation), but in some cases it will not be. You will also be calculating the 32-bit extended immediate value, which varies depending on the instruction. The instructions to decode are:

1. ADD
2. ADDU
3. ADDI
4. ADDIU
5. SUB
6. SUBU
7. SLT
8. SLTU
9. SLTI
10. SLTIU
11. AND
12. ANDI
13. OR
14. ORI
15. XOR
16. XORI
17. NOR
18. SRL
19. SRA
20. SLL
21. SRLV
22. SRAV
23. SLLV
24. LW
25. SW
26. BEQ
27. BNE
28. BLTZ
29. BLEZ
30. BGTZ
31. BGEZ
32. LUI

In ALU.v you will have to calculate the result for the different operations you have specified in Decode.v. Once again, some results will be straightforward to implement, while others will require more thinking.

**Simulation Instructions:**

In ModelSim you can create a new project and add all the .v files from the starter kit. Make sure you have the tcgrom, framebuffer, dataram, and irom files in the same directory as your ModelSim project. Before writing any code, type the following commands in the ModelSim command line (note, you may need to do this every time you restart ModelSim).

vmap unisims\_ver /afs/ir.stanford.edu/class/ee/mentor/vsim66/modeltech/xilinx\_libs/unisims\_ver

vmap xilinxcorelib\_ver /afs/ir.stanford.edu/class/ee/mentor/vsim66/modeltech/xilinx\_libs/xilinxcorelib\_ver

Also, to see the signals so that you can add them to the wave forms, issue the following command as well (NOTE: you need to run the command below at least once to force ModelSim to resolve linking issues with the BLKMEM libraries):

vsim –L xilinxcorelib\_ver –L unisims\_ver –t 1ps {-voptargs=+acc=bcglnprst –O0} work.MIPStest

Note that if you want to run a different testbench you need to change the above to work.OTHER\_TESTBENCH\_NAME.

Alternatively, you can also run simulations from the GUI.

**Submission Requirements:**

You must submit your lab report electronically. Please refer to the lab report guidelines online.

In your submission, you must include the following:

1. All your lab code (only the files you modified).
2. All testbenches that you wrote.
3. Screenshots of testbenches that demonstrate the correct functionality of the following instructions: ADD, ADDI, SLT, SLTI, LW, SW, AND, ANDI, OR, ORI, XOR, XORI, SLL, SRA, BEQ, BNE, LUI, BLTZ, BGEZ. These screenshots should be clear, concise, and annotated in such a way as to make them easy to understand.

You will be required to show all the above screenshots at the lab demo. Also, it is highly advised that you write testbenches for the other instructions as well since we will be running test scripts on your implementation to verify the correct functionality of all the instructions.

**Extensions:**

If you have completed the above requirements, you may choose to implement one or more of the following extensions:

* Overflow detection: Provide an output from ALU.v that is asserted whenever an operation results in overflow (be careful to distinguish between signed and unsigned ops). If you do this, you must provide a screenshot for ADDU that demonstrates this, and your screenshot for ADD must demonstrate the output stays low when overflow occurs in an ADD operation.
* Implement the instructions BGTZAL and BLEZAL (either do both or do neither, no EC for only doing one) – see Appendix A in the textbook for descriptions of these instructions.