ECE 2500: Spring 2018
Project 2 (Due Friday April 6th, 9:00 PM)

CHECK THE LATE POLICY ON SYLLABUS

100 points

For this project, you will extend and evaluate the datapath for a single-cycle MIPS architecture in Verilog to include several additional instructions.

Starting Point: The zipped directory P2-start gives you a starting point for your datapath implemention. The instructions `addi` and `lui` are already implemented within this starter code. All the modules that are needed to finish this project are given to you. The comments at the beginning of each file give you the description and important information about the module defined in that file. The modules for this project are written in behavioral Verilog. You do not necessarily need to know the internal working of each module. However, you do need to recognize the interface (i.e., inputs, outputs and functionality) of each module.

Some of these modules are already instantiated in the top module MIPS (in the file mips.v). Others will need to be instantiated to complete the project.

Note: The datapath control also recognizes the instruction with opcode and func field of `0` (sll) and sets the control lines to something reasonable so that nothing terrible happens.

Objectives: Our goal is to extend this initial datapath to include the following MIPS instructions:

- Group 1: `add, sub, slt, andi, nor` [10 points]
- Group 2: `lw, sw` [15 points]
- Group 3: `beq, bne` [15 points]
- Group 4: `j, jr, jal` [20 points]

The implementation of these instructions will involve the extension of the control component and the addition of some buses and other components. **In order to do so, you need to modify two modules only: MIPS and MIPS_CONTROL, which are defined in mips.v and mips-control.v respectively.**

The starter code also contains a testbench module. Please follow the instructions given in *Modelsim_quickstart.pdf* to see how to run the testbench simulations and visualize the timing waveforms. You can also see a screencast of how to run the testbench online. We will also go over this in class.

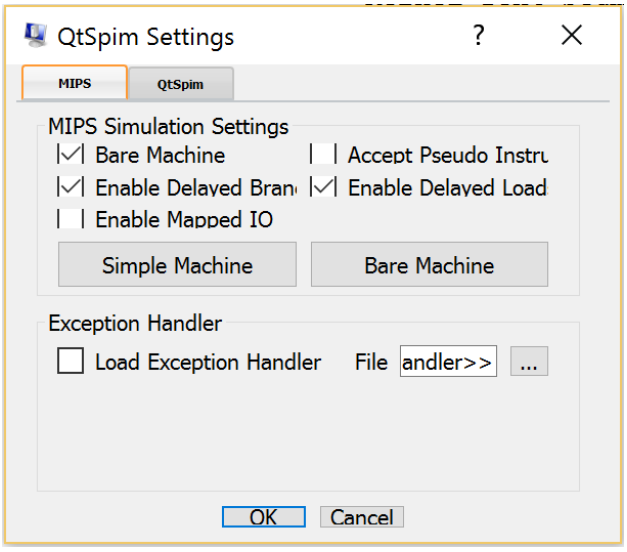
When you start the simulation with a testbench, the instruction memory module reads a text file called program.txt which contains the instructions to be executed (given in 32-bit hex format). The default starter code has 4 32-bit machine codes given in program.txt. The corresponding MIPS instructions are in the file program.s. As you can observe, this simple program uses the instructions (`addi`, `lui`, `sll`) which have already been implemented in the starter code. You can load and run the simulation, and by default, you should be able to observe the contents of the PC, control signals, registers, etc. in Modelsim.

In addition to modifying mips.v and mips-control.v, you will also be required to create assembly test cases, one corresponding to each new group of instructions, and use them to simulate the datapath.

What to turn in:

1. Verilog modules [60 points]: Create a directory called **P2_*<PID>*** and put all the verilog files associated with this part, **including the ones that you did not modify**, in this directory.

2. Test cases [20 points]: For each group of the instructions that you implemented, provide an assembly program that demonstrates their correct implementation. Using QtSpim, generate the hex instructions and strip them to the format similar to program.txt, which is given to you in the starter directory. Note that 0x



should be removed from the instructions, otherwise an error will occur when reading the file. Copy your test cases to program.txt to simulate them. Include both the assembly and the stripped version of hex instructions in your directory. Call them test_*i*.s and test_*i*.txt, where *i* represents the group of instructions the program is testing.

3. Report [20 points]: Write up a report that includes a brief description of your implementation and testing process. Your report should:
    a. A brief explanation of the changes made to the data path to implement each group of the instructions along with any problems you were faced and were not able to solve.
    b. For each instruction group you implemented, show how you tested them by completing the below steps:
        i. Simulate your datapath running the test case corresponding to that instruction group. Add a screenshot of the timing waveform.
        ii. Create a table that contains the **necessary information** for understanding the waveform. The list of necessary information for each instruction group is given below. All of these quantities can be found when you run the simulation in Modelsim.
            - Group 1: PC, instruction, opcode and func along with write address, write data and write control for register file.
            - Group 2: PC, instruction, imm, write address/data/control for register file along with address, output data, write data, read control and write control for the data memory.
            - Group 3: PC, instruction, zero signal, jump and branch signals (outputs of the control module).
            - Group 4: PC, instruction, jump and branch signals, write address/data/control for register file

Your report should be entitled ***project2_PID*** and put into **P2_*PID*** directory. By the due date and time, zip it up, and upload it to canvas.

Start early. This may take longer than you think. Good luck!