**ECE 2500  Computer Organization and Architecture**

# Fall 2015

# Project 2 (Due Friday October 30th, 9:00 PM)

CHECK THE LATE POLICY ON SYLLABUS

# 1000 points

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

**Starting Point:** The zipped directory P2-start includes the starting datapath. The instructions addi and lui are already implemented within this datapath. The datapath control also recognizes the instruction with opcode and func field of 0 and sets the control lines to something reasonable so that nothing terrible happens. 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 are not expected to understand how each module has been implemented. However, you need to recognize inputs, outputs and functionality of each module. Some of these modules are already instantiated in the top module MIPS. Some, still need to be instantiated to complete the project.

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

* Group 1: add, sub, slt, andi, nor [100 points]
* Group 2: lw, sw [200 points]
* Group 3: beq, bne [200 points]
* Group 4: j, jr, jal [250 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.**

**What to turn in:**

1. Verilog modules [750 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 [100 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 starting point directory. 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 [150 points]: Write up a report that includes a brief description of your implementation and testing process. Your report should include the following segments:
   1. 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.
   2. For each instruction group you implemented, show how you tested them by completing the below steps:
      1. Simulate your datapath running the test case corresponding to that instruction group.
      2. Create a table that contains the **necessary information** for understanding the waveform. You can use Microsoft spreadsheets in order to generate the tables and write up the rest of your report in the spreadsheet file as well. The list of necessary information for each instruction group is given below:
         * 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 scholar.

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