# CS224 - Spring 2018 - Lab #4 (Version 1: March 26 2018)

## MIPS Single-Cycle Datapath and Controller & Experiments on SystemVerilog

**Dates:** Section 1, Monday, 2 April, 13:40-17:30
Section 3, Tuesday, 3 April, 13:40-17:30
Section 2, Wednesday, 4 April, 13:40-17:30
Section 4, Thursday, 5 April, 13:40-17:30
Section 5, Friday, 6 April, 8:40-12:30
Section 6, Friday, 6 April, 13:40-17:30

**Purpose**: Understanding the design of the Single-Cycle MIPS Processor: What are the roles of each module in the datapath and the control units, how do control signals dictate the behavior of the processor.

**Summary**
**Part 1** (50 points): Understanding modules in the single-cycle processor architecture, preparation for the Xilinx Vivado simulations of the single-cycle MIPS processor.
**Part 2** (50 points): Writing test benches for individual modules, as well as the whole processor. Simulating MIPS-lite processor in Xilinx Vivado. Analyzing behaviors of the control signals with respect to the instruction that is being executed.

**TRY TO FINISH PART 2 BEFORE COMING TO THE LAB. MAKE SURE THAT YOU DO THE DEMO TO TA.**

**DUE DATE/TIME OF PART 1 --SAME FOR ALL SECTIONS**

a. Please drop your written Preliminary Design Report into the box provided in front of the lab by 13:40 on Monday April 2. No late submissions will be accepted!
b. Please **upload your Preliminary Design Report** to the Unilica Assignment for Preliminary Work by 13:40 on Monday April 2. Use filename **name_surname_SecNo_PRELIM.txt** [A NOTEPAD FILE as its extension suggests, which contains all the work done for the Preliminary Part]

**DUE TIME OF PART 2—DIFFERENT FOR EACH SECTION:**

a. You have to demonstrate your Part 2 lab work to the TA for grade by **12:15** in the morning lab and by **17:15** in the afternoon lab. Your TAs may give further instructions on this. If you wait idly and show your work last minute your TA may not accept it. Make sure that you follow your TAs' instructions.
b. At the conclusion of the demo for getting your grade, you will **upload your lab work** to the Unilica Assignment, for similarity testing by MOSS. Please see the related section below for further instructions on MOSS submission.

## Part 1. Preliminary Work / Preliminary Design Report (50 points)

You have to provide a neat presentation prepared by Word or a word processor with similar output quality. Handwritten answers will not be accepted. At the top of the paper on left provide the following information and staple all papers. In this part provide the program listings with proper identification. Please make sure that this info is there for proper grading of your work, otherwise some points will be taken off.

CS224
Section No.: ...
Spring 2018
Lab No.:
Your Full Name/Bilkent ID:

**1.** **[5 pts]** What does it mean for a processor to be a *single-cycle*? What are the differences between single-cycle, multi-cycle and pipelined architectures? Briefly explain.

**2.** **[10 pts]** In this lab, we will work on a MIPS architecture that executes instructions in a single cycle. A single-cycle processor consists of different units that can be divided into two main parts: the datapath and the control unit. *List* the modules that constitute each of these parts. *Explain* the functions of each module with your own words [Do not copy/paste the definitions in the textbook. You may omit the adders and multiplexers in the datapath for this part, and focus on the main units].

**3.** **[15 pts]** Looking at the complete processor (see the textbook, p.383) might be intimidating at first. That is why, it is important that you isolate each module and analyze them individually.
What are the inputs and outputs of each module? Write a descriptive signature for each module that contains its inputs and outputs, as well as how many bits each input/output has.

For example:

```
ExampleModule(input a, input[32] b, output[32] c)
```

means that the ExampleModule takes two inputs: a and b, consisting of 1 and 32 bits respectively; and an output c, which has 32 bits. [Accomplishing this is the key to be able to write SystemVerilog modules. Whenever you want to write a SystemVerilog module, treat it as a blackbox and ask the following questions: if this module was doing what it is supposed to be doing, (1) what would be its inputs and outputs, (2) how many bits each of them would have?

**4.** **[20 pts]** Determine the assembly language equivalent of the machine codes given in the imem module in the "imem.txt" file posted on Unilica for this lab. In the given SystemVerilog module for imem, the hex values are the MIPS machine language instructions for a small test program. Dis-assemble these codes into the equivalent assembly language instructions and give a 3-column table for the program, with one line per instruction, containing its location, machine instruction (in hex) and its assembly language equivalent. [Note: you may dis-assemble by hand or use a program for this purpose]

**[REMINDER!]** In this lab and the next one, we assume SystemVerilog knowledge, as well as the ability to use tools such as Xilinx Vivado and Digilent BASYS3 Board, since you all took CS223 – Digital Design. If you are not familiar with these, please get used to them as soon as possible.

## Part 2. Implementations and Simulations (50 points)

In this lab, you are given the file "NotSoComplete_MIPS_Model.txt" which consists of SystemVerilog modules of the units in the single-cycle MIPS processor (except its Instruction Memory as it is given as a separate document, called "imem.txt"), where some of its units have missing parts that needs to be filled in. This MIPS processor can handle only 10 instructions when it is completed, and that is why it is often times called MIPS-lite processor.

You will complete the SystemVerilog codes for MIPS-lite wherever necessary, write meaningful test benches for your modules to make sure that they are working correctly, and simulate your test benches in Vivado to analyze behaviors of modules and signals between the modules.

**Part 2.1. [20 pts]** *Completing and Simulating 32-Bit ALU Module*:

1) **[5 pts]** Complete the SystemVerilog code for the 32-bit ALU module (one is partly specified already in "NotSoComplete_MIPS_Model.txt", at the end of the file) and save this ALU module by itself in a new file with a meaningful name.

2) **[10 pts]** Make this file (the complete ALU) the basis of a new Xilinx Vivado project. Write a meaningful test bench that captures all of its desired behavior. It should include every operation that ALU can perform, with reasonable inputs that allows you to differentiate that it works correctly.

3) **[5 pts]** In simulation, check its syntax, and then simulate this ALU, using the test bench that you wrote in the previous part. Interpret what you are seeing in the simulation. Is your ALU working correctly? When you are sure that the 32-bit ALU is working correctly in simulation, you can now use it in the MIPS-lite datapath.

**Part 2.2. [30 pts]** *Completing Main Decoder and Simulating MIPS Processor:*

1) **[10 pts]** Complete the SystemVerilog code for the Main Decoder (the module that is named as **maindec** in "NotSoComplete_MIPS_Model.txt"). Assign the control signals that capture the instructions' behavior. To accomplish this, you must know which instruction each opcode corresponds to. The control signal for the R-TYPE instruction is already given in the module. The rest will be filled for the following instructions:

   *{LW, SW, BEQ, ADDI, J}*

   Once the Main Decoder is completed, we are ready to simulate the MIPS-lite.

2) **[5 pts]** Study the small test program that is loaded into Instruction Memory that you disassembled in the preliminary part. What is the program attempting to do?

3) **[10 pts]** Now make a SystemVerilog test bench file and simulate your MIPS-lite processor executing the test program. Study the results given in the simulation window. Find each instruction, and understand its values. For instance, why is *writedata* undefined for some of the early instructions in the program?

4) **[5 pts]** Can you modify the simulation in order to show more information? Make changes to the SystemVerilog modules as needed, so that 32-bit values of PC and the Instruction are made to be outputs of the top-level module. Then, modify the test bench file so that they are displayed in the simulation.

## Part 3. Submit your code for MOSS similarity testing

Submit your MIPS codes for similarity testing to the Unilica > Assignment specific for your section. You will upload one file: **name_surname_SecNo_Lab4_MIPS.txt** created in the relevant parts. Be sure that the file contains exactly and only the codes which are specifically detailed above, including Part 1 programs (your paper submission for preliminary work must match MOSS submission). Check the specifications! *Even if you didn't finish, or didn't get the MIPS codes working, you must submit your code to the Unilica Assignment for similarity checking.* Your codes will be compared against all the other codes in the class, by the MOSS program, to determine how similar it is (as an indication of plagiarism). So be sure that the code you submit is code that you actually wrote yourself ! All students must upload their code to Unilica > Assignment while the TA watches. Submissions made without the TA observing will be deleted, resulting in a lab score of 0.

## Part 4. Cleanup

1) After saving any files that you might want to have in the future to your own storage device, erase all the files you created from the computer in the lab.
2) When applicable put back all the hardware, boards, wires, tools, etc where they came from.
3) Clean up your lab desk, to leave it completely clean and ready for the next group who will come.

---

## LAB POLICIES

1. You can do the lab only in your section. Missing your section time and doing in another day is not allowed.
2. Students will earn their own individual lab grade. The questions asked by the TA will have an effect on your individual lab score.
3. Lab score will be reduced to 0 if the code is not submitted for similarity testing, or if it is plagiarized. MOSS-testing will be done, to determine similarity rates. Trivial changes to code will not hide plagiarism from MOSS—the algorithm is quite sophisticated and powerful. Please also note that obviously you should not use any program available on the web, or in a book, etc. since MOSS will find it. The use of the ideas we discussed in the classroom is not a problem.

4. You must be in lab, working on the lab, from the time lab starts until your work is finished and you leave.
5. No cell phone usage during lab.
6. Internet usage is permitted only to lab-related technical sites.
7. For labs that involve hardware for design you will always use the same board provided to you by the lab engineer.