CS224 – Fall 2021 - Lab #5 (Version 3: Nov. 27, 2:34 pm)

Implementing the MIPS Processor with Pipelined Microarchitecture

Dates: Section 1, Wednesday, 1 December, 13:30-17:20
Section 2, Thursday, 2 December, 13:30-17:20
Section 3, Wednesday, 1 December, 8:30-12:20
Section 4, Friday, 3 December, 13:30-17:20
Lab Location: EA-Z04

TAs
Section 1: Kenan Çağrı Hırlak, Pouya Ghahramanian
Section 2: Kenan Çağrı Hırlak, Soheil Abadifard
Section 3: Ege Berkay Gülcan, Sepehr Bakhshi
Section 4: Alper Şahıstan, Soheil Abadifard (Tutor: Burak Öçalan)

TA/Tutor name (x No of labs) email address, Office Hour, Office
Alper Şahıstan (x1): alper.sahistan@bilkent.edu.tr, Friday 11:30-12:20, EA-407
Ege Berkay Gülcan (x1): berkay.gulcan@bilkent.edu.tr, Thursday 14:30-15:20, EA-507
Kenan Çağrı Hırlak (x2): cagri.hirlak@bilkent.edu.tr, Friday 12:30-13:20, EA-425
Pouya Ghahramanian (x1): ghahramanian@bilkent.edu.tr, Wednesday 12:30-13:20, EA-407
Sepehr Bakhshi (x1): sepehr.bakhshi@bilkent.edu.tr, Friday 13:30-14:20, EA-507
Tutor Burak Öçalan (x1): burak.ocalan@ug.bilkent.edu.tr, Tuesday 18:15-20:00, EA-Z04

Purpose: In this lab you will implement and test a pipelined MIPS processor using the digital design engineering tools (System Verilog HDL, Xilinx Vivado and FPGA, Digilent BASYS3 development board). To do this, you will need to add pipeline registers, forwarding MUXes, a hazard unit, etc to your datapath, and of course make the control pipelined as well. You will be provided a skeleton System Verilog code for the Pipelined MIPS processor and fill the necessary parts to make it work. Then, you will synthesize them and demonstrate on the BASYS3 board.

Summary
1. **Part 1** (60 points): Preliminary Report/Preliminary Design Report: Verilog model for Pipelined MIPS processor handling Original10 instruction (Due date of this part is the same for all).
2. **Part 2** (40 points): Simulation of the MIPS-lite pipelined processor and testing on BASYS3 board.

Important Notes for All Labs

1. Try to complete the lab part at home before coming to the lab. Make sure that you show your work to your TA and answer his questions to show that you know what you are doing before uploading your lab work and follow the instructions of your TAs. In all labs if you are not told you may assume that inputs are correct. For all works when needed please provide a simple user interface for inputs and outputs.
2. You are obliged to read this document word by word and are responsible for the mistakes you make by not following the rules. Your programs should be reasonably documented (purpose etc.) and must have a neat presentation in terms of variable names, subprogram names. indentation, comments, blank lines etc.
3. **If we suspect that there is cheating we will send the work with the names of the students to the university disciplinary committee.**

DUE DATE OF PRELIMINARY WORK: SAME FOR ALL SECTIONS

No late submission will be accepted.

- **a.** Please upload your programs of preliminary work to Moodle by 9:30 am on Wednesday December 1.
- **b.** For preliminary work upload a file that includes all programs in proper order with the filename **StudentID_FirstName_LastName_SecNo_PRELIM_LabNo.pdf** Only a PDF file is accepted. Any other form of submission receives 0 (zero).
- **c.** Note that the Moodle submission closes sharp at 9:30 am and no late submissions will be accepted. You can make resubmissions before the system closes, so do not wait for the last moment. Submit your work earlier and change your submitted work if necessary. Note that only the last submission will be graded.
- **d.** Do not send your work by email attachment, they will not be processed. They have to be in the Moodle system to be processed.
- **e.** At the beginning of your submission files include the following. Make sure that each of them is in a separate line: Course No.: CS224, Lab No., Section No., Your Full Name, and your Bilkent ID.

DUE DATE OF LAB WORK): (different for each section) YOUR LAB DAY

- **a.** You have to demonstrate your lab work to your TA for grading. Do this by **12:00** in the morning lab and by **17:00** in the afternoon lab. Your TAs may give further instructions on this and they may make changes. If you wait idly and show your work at the last minute, your work may not be graded. Make sure that you follow your TA's instructions.
- **b.** At the conclusion of the demo for getting your grade, you will **upload your Lab Work** to the Moodle Assignment, for similarity testing by MOSS. See lab part submission details below.
- **c.** Aim to finish all of your lab work before coming to the lab, but make sure that you upload your work after making sure that your work is analyzed by your TA and/or you are given the permission by your TA to upload.

d. At the beginning of your submission files include the following. Make sure that each of them is in a separate line: Course No.: CS224, Lab No., Section No., Your Full Name, and your Bilkent ID.

Part 1. Preliminary Work / Preliminary Design Report (60 points)

At the end of this lab, you will have implemented the pipelined MIPS architecture that can be seen in the file that is provided as ***PipelineDatapath.PNG*** (Notice that there is no early branch prediction in this pipeline. Hence, the branch resolution is done in the Execute stage.). Be sure to have a printout of the pipelined processor with you, to use during the lab. Your PDR should contain the following items:

a) Cover page, with university name, department name, and course name and number at the top, "Preliminary Design Report", Lab # (e.g 5), Section #, and your name and ID# in the middle, and the date of your lab at the bottom.

b) **[10 points]** The list of all hazards that can occur in this pipeline. For each hazard, give its type (data or control), its specific name ("compute-use" "load-use", "load-store" "J-type jump", "branch" etc), the pipeline stages that are affected, the solution (forwarding, stalling, flushing, combination of these), and explanation of what, when, how.

c) **[5 points]** The logic equations for each signal output by the hazard unit, as a function of the input signals that come to the hazard unit. This hazard unit should handle all the data and control hazards that can occur in your pipeline (listed in c) so that your pipelined processor computes correctly.

d) **[30 points]** You are given a skeleton System Verilog code for your pipelined MIPS processor in the file ***PipelinedMIPSProcessorToFill.txt***. The places in the code that needs to be modified are shown with comment blocks above them. Fill them and **highlight the changes you made** in the code **in your report**. You can use a different text highlight color (like this) for this purpose. You don't need to follow the skeleton code point by point. If you think your design is better, you are welcome to try it in your code, as long as your version of the code works, too.

e) **[15 points]** Write small test programs, in MIPS assembly, that will show whether the pipelined processor is working or not. Each of your test programs should be designed to catch problems, if there are any, in the execution of MIPS instructions in your pipelined machine. Write:

- A test program with no hazards (to verify that there are no problems with the connections in your pipeline etc.)
- A test program that has one type of hazard, and another, and another...

In the end, have at least 4 test programs (testing at least 3 hazards) with their machine code (in hex).

You can use the student-written assembler tool available online to help you quickly implement your test programs[1]*. Remember that the goal of testing is to verify that all the instructions are fully working, and that all the instructions are working even in the presence of hazards.*

You may check the samples given in “Sample tests for hazards.txt” for intuition. However, you must write your own tests for Part 1.g

Part 2: Simulation and Implementation

Simulation of the Pipelined MIPS-lite processor (20 points)

a) Complete the System Verilog model of pipelined MIPS-lite by applying the modifications that you have done in Part 1.d to the file ***PipelinedMIPSProcessorToFill.txt***. Create a top module to connect *mips*, *imem* and *dmem* modules.

b) Make a New Project, giving it a meaningful name, for your pipelined MIPS-lite. Do Add Source for the System Verilog modules given in ***PipelinedMIPSProcessorToFill.txt*** (completed by you), and Save everything.

c) Now make a System Verilog testbench file and using Xilinx Vivado, simulate your pipelined MIPS-lite processor executing the test program given in *imem* module. Study the results given in the simulation window. Find each instruction, and understand its values in each pipeline stage.

d) For each module that you wrote or modified, you will want to test it in simulation, to make sure it works functionally. When you have integrated all the System Verilog modules together and your whole pipelined MIPS-lite is working in simulation with the test programs you wrote, call the TA and show it for grade. To get full points from this part, you must know and understand everything about what you have done.

Now save the System Verilog model of pipelined MIPS-lite, plus the System Verilog code of the testbench module used for your simulation in their final form. Put these modules together in a text file named **StudentID_FirstName_LastName_SecNo_LAB_LabNo.txt**. It should contain all the System Verilog codes from these modules, copy-pasted together in one .txt file. In Part 3, you will upload your file to the Moodle Assignment for your section, for similarity checking with MOSS.

Implementation and Testing (20 points)

a) To implement the modified processor, you will need to assemble the modified System Verilog modules that model the changes you made to implement the pipelined microarchitecture. You should also consider the following: to slow down the execution to an observable rate, the clock signal should be hand-pushed, to be under user control. One clock pulse per push and release means that instructions advance one stage in the pipeline. Once the pipeline is full, it means that each push will cause one

[1] https://github.com/bilkentCraps/mips (as of 27 Nov 2018)

instruction to finish unless a flush or stall caused nothing to complete that cycle. Similarly the reset signal should be under hand-pushed user control. So these inputs need to come from push buttons, and to be debounced and synchronized. The memwrite and regwrite outputs (along with any other control signals that you want to bring out for viewing) can go to LEDs, but the low-order bits of writedata (which is RF[rt]) and of dataaddr (which is the ALU result) should go to the 7-segment display, in order to be viewed in a human-understandable way. [Consider why it isn't necessary to see all 32 bits of these busses, just the low-order bits are enough.]

For the sake of this part, you must use the modules given in ***pulse_controller.txt*** and ***display_controller.txt.*** You don't need to modify the given modules. You should read the descriptions at the top of files before using the modules. You will create a new top-level System Verilog module containing the top module as well as 2 instantiations of pulse_controller (for clock and reset pulses), and 1 instantiation of display_controller (to send outputs to 7-segment display). The instantiations of pulse_controller will take 100 Mhz clock and pushbutton switches of BASYS3 as their inputs and will output synchronized debounced clock and reset pulses respectively. Your top level pipelined MIPS module should take the clock and reset pulses created by pulse_controller instantiations as its input. You should give the low order bits of writedata and of dataaddr as the inputs of display_controller. The outputs of display_controller should go to the 7-segment display of BASYS3.

b) Make the constraint file that maps the inputs and outputs of your new top-level System Verilog model to the inputs (100 Mhz clock and pushbutton switches) and outputs (AN and CA signals to the 7-segment display, and memwrite (plus others?) going to a LED) of the BASYS3 board and its FPGA.

c) Now implement your project on the BASYS3 board, and test it. When your pipelined instructions are working correctly in hardware, call the TA and show it for grade. Note: the TA will ask questions to you, in a single 20-point demo and Oral Quiz, to determine how much of the 20 points for this part of Part 2 (implementation) is deserved, based on your demo, your knowledge and ability to explain it and the System Verilog code and the reasons behind it, and what would happen if certain changes were made to it. To get full points from the Oral Quiz, you must know and understand everything about what you have done.

Part 3. Submit Lab Work for MOSS Similarity Testing

1. Submit your Lab Work MIPS codes for similarity testing to Moodle.
2. You will upload one file. Use filename **StudentID_FirstName_LastName_SecNo_LAB_LabNo.txt**
3. <u>Only a NOTEPAD FILE (txt file) is accepted.</u> No txt file upload means you get 0 from the lab. Please note that we have several students and efficiency is important.
4. *Even if you didn't finish, or didn't get the MIPS codes working, you must submit your code to the Moodle Assignment for similarity checking.*
5. 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!

6. At the beginning of your submission files include the following. Make sure that each of them is in a separate line: CS224, Lab No., Section No., Your Full Name, Bilkent ID.
7. For your preliminary and lab work to be graded you must attend the lab.

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 it another day is not allowed.
2. The questions asked by the TA will have an effect on your lab score.
3. Lab score <u>will be reduced to 0</u> 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 the lab, working on the lab, from the time the lab starts until your work is finished and you leave.
5. No cell phone usage during the lab.
6. Internet usage is permitted only to lab-related technical sites.