CS224 - Spring 2018 - Lab #5 (Version 1: March 27 2018)

Extending Single-Cycle MIPS Processor & Experiments on SystemVerilog and BASYS3 Board

Dates: Section 1, Monday, 9 April, 13:40-17:30
Section 3, Tuesday, 10 April, 13:40-17:30
Section 2, Wednesday, 11 April, 13:40-17:30
Section 4, Thursday, 12 April, 13:40-17:30
Section 5, Friday, 13 April, 8:40-12:30
Section 6, Friday, 13 April, 13:40-17:30

[THIS LAB CAN BE DONE IN PAIRS. HOWEVER, BOTH OF THE STUDENTS ARE RESPONSIBLE FOR WHAT IS GOING ON IN ALL PARTS OF THE LAB. TEAM MEMBERS MAY GET DIFFERENT GRADES BASED ON THE QUALITY OF ANSWERS THEY GIVE TO THE TAS.]

Purpose: Extending MIPS-lite processor by adding new instructions to the datapath. Bringing new capabilities to the processor by adding new control signals, muxes etc. Simulating the extended processor both on software (Xilinx Vivado) and hardware (BASYS3 Board).

Summary

Part 1 (50 points): Examining the new instructions thoroughly and designing the changes that need to be made in the datapath and control units. Applying these changes in the SystemVerilog modules.
Part 2 (50 points): Implementing the extensions for the new instructions to single-cycle MIPS-lite processor. Simulating the new capabilities of the processor on Xilinx Vivado and on BASYS3 Board.

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 9. 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 9. 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:

In this lab, you are going to extend the single-cycle MIPS processor that you have designed in the previous lab, for new instructions that are given below in the "Table of Instructions to Implement". If your MIPS-lite processor still has problems with its ALU or Main Decoder (that you filled in the previous lab), fix those before starting this lab.

Each section will be given two unique instructions to extend the basic instruction set of MIPS-lite, which previously consisted of 10 instructions (referred as Original10).

Before starting, for each new instruction, make sure that you understood what the instruction really does. Then, follow the information flow for that instruction through the datapath, infer wherever a new control signal or a new multiplexer is necessary, and do the necessary changes in the datapath. Meanwhile, make sure that the Original10 instructions have not been affected by the changes you made in the datapath in any step.

1. **[5 pts]** Register Transfer Level (RTL) expressions for each of the new instructions that you are adding (see list below for your section), including the fetch and the updating of the PC.

2. **[10 pts]** Make any additions or changes to the datapath which are needed in order to make the RTLs for the instructions possible. The base datapath should be in black, with changes marked in red and other colors (one color per new instruction). You can take a photo / scan your modified datapath and add it to your Preliminary Report, as long as the changes you have done are clear and understandable.

3. **[5 pts]** Make a new row in the main control table for each new instruction being added, and if necessary add new columns for any new control signals that are needed (input or output). Be sure to completely fill in the table—all values must be specified. If any changes are needed in the ALU decoder table, give this table in its new form (with new rows, columns, etc). The base table should be in black, with changes marked in red and other colors. {Note: if you need new ALUOp bits to encode new values, you should also give a new version of Table 7.1, showing the new encodings}

4. **[10 pts]** Write a test program in MIPS assembly language, that will show whether the new instructions are working or not , and that will confirm that all existing old instructions still continue to work. Don't use any pseudo-instructions; use only real MIPS instructions that will be recognized by the new control unit.

5. **[20 pts]** Write a list of the System Verilog modules that will need changes in order to make these new instructions part of the single-cycle MIPS processor's instruction set. For each module in the list, determine the new System Verilog model that will be needed in order for the instructions to be added. Give the System Verilog code for each module that needs to be changed.

Table of Instructions to Implement (by section)

Section	MIPS instructions	Section	MIPS instructions
Sec 1	Base: Original10 New: "ble", "sw+"	Sec 4	Base: Original10 New: "jm", "blt"
Sec 2	Base: Original10 New: "nop", "subi"	Sec 5	Base: Original10 New: "jalm", lui
Sec 3	Base: Original10 New: "bge", "swapRM"	Sec 6	Base: Original10 New: jalr, "bgt"

(The Original10 instructions in "MIPS-lite" are `add, sub, and, or, slt, lw, sw, beq, addi` and `j`. Definitions of the new instructions are given below.)

Instructions **in quotes** (e.g."sw+") are not defined in the MIPS instruction set. They don't exist in any MIPS documentation, they are completely new to MIPS. You will create them, according to the definitions below, then implement them.

`nop:` This I-type instruction does nothing, changes no values, takes one clock cycle. Except for the opcode (which you need to assign a code to), the I-type instruction field values are irrelevant. Example: nop {Note: an R-type nop exists in real MIPS, it is sll $0, $0, 0. But the nop here is different, so it is "nop" !}

`subi:` This I-type instruction subtracts, using a sign-extended immediate value. subi $t2, $t7, 4

`jm:` This I-type instruction is a jump, to the address stored in the memory location indicated in the standard way. Example: jm 40($s3)

`jalm:` This I-type instruction is a jump, to the address stored in the memory location indicated in the standard way. But it also puts the return address into the register specified. Example: jalm $t5, 40($s3)

`bge, bgt, ble, blt:` These I-type instructions do what you would expect—branch to the target address, if the condition is met. Otherwise, the branch is not taken. Example: bge $t2, $t7, TopLoop

swapRM: This I-type instruction exchanges the data in two locations: the exchange is between a register value and a memory value (addressed in the standard way). Example: swapRM $v0, 1004($sp)

sw+: [done in class] This I-type instruction does the normal store, as expected, plus an increment (by 4, since it is a word transfer) of the base address in RF[rs]. {Note: these kind of auto-increment instructions are useful when moving through an array of data words.}

Part 2. Implementations and Simulations (50 points)

Part 2.1. [25 pts] *Simulating the Extended MIPS Processor*:

1) **[10 pts]** Implement the modified processor by making the necessary changes to the System Verilog modules in your Preliminary Design Report, part 5. Integrate these all together into a new System Verilog model for the MIPS single-cycle processor that does the Original10 instructions plus the new instructions required for your section.

2) **[5 pts]** Modify the test program (or write a new one, in hex) in the Instruction Memory so that it includes at least one line of each newly introduced instruction. This small program will be simulated to check whether both the old 10 and the new 2 instructions are working properly or not.

3) **[10 pts]** Run a test bench on the Extended MIPS processor, with the Instruction Memory now having lines involving new instructions. Analyze and interpret the simulation results. Be able to explain the values of PC, instruction, writedata, dataaddr, memwrite etc. for any instruction.

Part 2.2. [25 pts] *Simulating the Extended MIPS Processor on BASYS3 Board:*

To demonstrate the processor on BASYS3 Board, 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 one instruction is fetched-decoded-executed. 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 output (along with any other control signals that you want to bring out for viewing) can go to a LED, 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.]

1) **[10 pts]** In view of the above, create a new top-level System Verilog module, to model the system that contains an instantiation of the MIPS computer (in module `top`), as well as 2 instantiations of `pulse_controller`, and 1 instantiation of `display_controller`. Your system should include some hand-pushed signals coming from push buttons, and some anode and cathode outputs going to the 7-segment display unit on the BASYS3 board and memwrite (and possibly other outputs) going to a LED.

2) **[5 pts]** Make the constraint file that maps the inputs and outputs of your top-level SystemVerilog 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.

3) **[10 pts]** Now create a New Project, and implement it on the BASYS3 board, and test it. When both of your new instructions are working correctly in hardware, and all 10 of the old instructions are also still working, call the TA and show it for grade. The TA will ask questions to you about your SystemVerilog code, 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 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_Lab5.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.