CS224 - Spring 2021 - Lab #4 (Version 2: March 17, 21:32)
MIPS Single-Cycle Datapath and Controller

Dates:
Section 1: Mon, 22 Mar, 8:30-12:20 in EA-Z04
Section 2: Wed, 24 Mar, 13:30-17:20 in EA-Z04
Section 3: Tue, 23 Mar, 13:30-17:20 in EA-Z04
Section 4: Mon, 22 Mar, 13:30-17:20 in EA-Z04
Section 5: Fri, 19 Mar, 8:30-12:20 in EA-Z04
Section 6: Fri, 19 Mar, 13:30-17:20 in EA-Z04

TAs:
Section 1: Zülal Bingöl, Ergün Batuhan Kaynak
Section 2: Zülal Bingöl, Kenan Çağrı Hırlak
Section 3: Alper Şahıstan, Kenan Çağrı Hırlak
Section 4: Ege Berkay Gülcan, Ergün Batuhan Kaynak
Section 5: Hüseyin Eren Çalık, Yusuf Dalva
Section 6: Ziya Erkoç, Yusuf Dalva

TA name (x No of labs) email address:
Alper Şahıstan (x1): alper.sahistan@bilkent.edu.tr
Ege Berkay Gülcan (x1): berkay.gulcan@bilkent.edu.tr
Ergün Batuhan Kaynak (x2): batuhan.kaynak@bilkent.edu.tr
Hüseyin Eren Çalık (x1): eren.calik@bilkent.edu.tr
Kenan Çağrı Hırlak (x2): cagri.hirlak@bilkent.edu.tr
Yusuf Dalva (x2): yusuf.dalva@bilkent.edu.tr
Ziya Erkoç (x1): ziya.erkoc@bilkent.edu.tr
Zülal Bingöl (x2): zulal.bingol@bilkent.edu.tr

Lab Attendance and Rotation Policy Reminder and Time for the Next Lab:
All labs will be done online and attendance is mandatory. You have to attend the online labs and submit the lab work by following the instructions of your TA. Before submission your TA will do a brief interview with you to understand your knowledge of the work you plan to submit. Note that preliminary works are submitted before all labs and are graded only if you attend the lab and submit your lab work.

Next Lab - Lab5 Days: Due to the lab rotation policy the first Lab5 session will be with Section 2 on April 7, Wednesday and Sections 5, 6 on on April 9, Friday. The lab days for Sections 1, 3 & 4 are on the following week starting with April 12, Monday.

Purpose: In this lab you will use an online digital design tool, EDA Playground (https://www.edaplayground.com/), to modify the single-cycle MIPS processor. Please visit Moodle for a video and FAQ about EDAPlayground. You will expand the instruction set of the "MIPS-lite" processor by adding new instructions to it. To do this, you must first determine the RTL expressions of the new instructions then modify the datapath and control unit of the MIPS. Note that for the jr instruction datapath and controller changes will be readily available to you. Implementing the new instructions will require you to modify some SystemVerilog modules in the HDL model of the processor which is provided in the same folder as this one in Moodle. To test and prove correctness, you will simulate the microarchitecture on EDAPlayground.

Summary
Part 1 (50 points): SystemVerilog model for Original12, Implementing JR, Presenting RTL instructions, datapath and controller changes for the new instruction assigned to your section,
Part 2 (50 points): Simulation of the MIPS-lite processor and Implementation and Testing of the new instruction assigned to your section.

[REMINDER!] In this lab and the next one, we assume SystemVerilog knowledge, since you all took CS223 – Digital Design. If you are not familiar with these, please get used to them as soon as possible

Note try, study, and aim to complete lab part at home before coming to the online lab.

DUE DATE OF PART 1 & 2 (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 Friday, Mar 19th.
b. Please 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 the last moment. Submit your work earlier and change your submitted work if necessary. Note that only the last submission will be graded.
c. Do not send your work by email attachment they will not be processed. They have to be in the Moodle system to be processed.
d . You will need to submit both a report file (in pdf) and code file (in txt).
e. For the report, use filename **StudentID_FirstName_LastName_SecNo_PRELIM_LabNo_Report.pdf** Only a PDF (pdf file) is accepted. Any other form of submission receives 0 (zero).
f. For the code, use filename **StudentID_FirstName_LastName_SecNo_PRELIM_LabNo_Code.txt** Only a NOTEPAD FILE (txt file) is accepted. Any other form of submission receives 0 (zero).

DUE DATE PART 3 (LAB WORK): (different for each section) YOUR LAB DAY

a. You have to demonstrate (using Zoom) 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. If you wait idly and show your work last minute, your work may not be graded.
b. At the conclusion of the demo for getting your grade, you will **upload your Lab Work Part 3** to the Moodle Assignment, for similarity testing by MOSS. See Part 6 below for details.
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. We use zoom to track your lab attendance.

If we suspect that there is cheating we will send the work with the names of the students to the university disciplinary committee.

Part 1 & 2. Preliminary Work (50 points)

For the preliminary work, you need to prepare a PDF file containing your answers to Part 1.a, 1.e, 1.f, 1.g and Part 2. You also need to put your MIPS code developed for Part 1.d to a txt file. As a result, you will submit two different files (one pdf, one txt).

Provide following five lines at the top of your submission for preliminary (both txt and pdf) and lab work (make sure that you include the course no. CS224, important for ABET documentation).
CS224
Lab No.
Section No.
Your Full Name
Bilkent ID

Make sure that you identify what is what at the beginning of each program by using proper comments.

Important Notes

1. **Assume that all Inputs are Correct**. In all lab works, if it is not explicitly stated, you may assume that program inputs are correct.

Instruction	Opcode	RegWrite	RegDst	ALUSrcA	ALUSrcB	Branch	Mem Write	MemToReg	ALU Op	Jump
R-type	000000	1	01	0	0	0	0	00	10	0
srl	000000	1	01	1	0	0	0	00	10	0

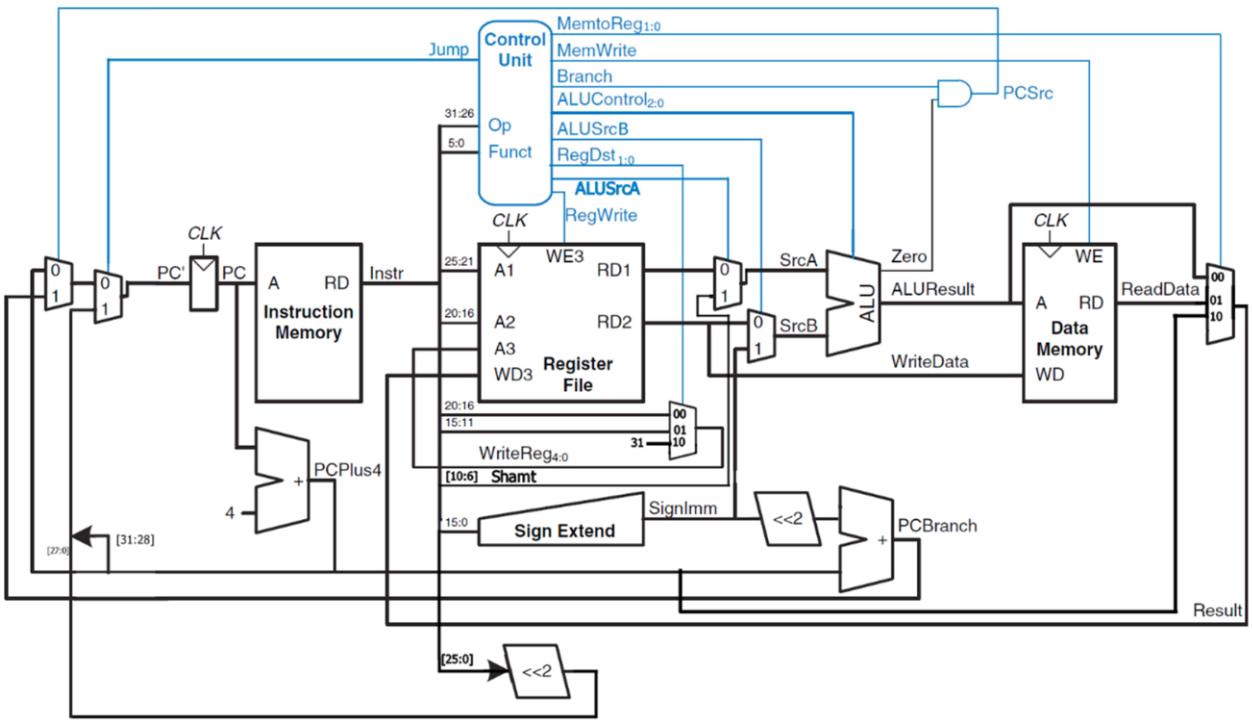

Figure 1: Datapath for Original12

Instruction	Opcode	RegWrite	RegDst	ALUSrcA	ALUSrcB	Branch	Mem Write	MemToReg	ALU Op	Jump
lw	100011	1	00	0	1	0	0	01	00	0
sw	101011	0	X	0	1	0	1	XX	00	0
beq	000100	0	X	0	0	1	0	01	01	0
addi	001000	1	00	0	1	0	0	00	00	0
j	000010	0	X	X	X	X	0	XX	XX	1
jal	000011	1	10	X	X	X	0	10	XX	1

Table 1: Main Decoder for Original12

ALUOp	Funct	ALUControl
00	X	010 (add)
X1	X	110 (subtract)
1X	100000 (add)	010 (add)
1X	100010 (sub)	110 (subtract)
1X	100100 (and)	000 (and)
1X	100101 (or)	001 (or)
1X	101010 (slt)	111 (set less than)
1X	000010 (srl)	011 (srl)

Table 2: ALU Decoder for Original12

Part 1.

a) **[3 Points]** Determine the assembly language equivalent of the machine codes given in the imem module in the "Complete MIPS model.txt" file posted on Moodle for this lab. In the given System Verilog 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 do dis-assembly by hand or use a program tool.]

b) Make a new Playground, giving it a meaningful name (from the "Add a title to help you find your playground box in the share tab below), for your single-cycle MIPS-lite. Create design files for the SystemVerilog modules given in "Complete MIPS model.txt", and Save everything (you don't have to use Complete MIPS model.txt, you can write your code if you find it more convenient). This file contains implementation of Original12 instructions in "MIPS-lite" which are the following: **add**, **sub**, **and**, **or**, **slt**, **lw**, **sw**, **beq**, **addi**, **j**, **jal**, **srl**.

c) Study the code and convince yourselves that the *datapath* module exactly corresponds to Figure 1 and *controller* module (including *maindec* and *aludec* modules) to Table 1 and Table 2. In the decoder tables, X means it is a "don't care". That is, it does not matter if it is 0 or 1. Although srl is an R-type instruction, in the Table 1, we added a new row for srl since its ALUSrcA bit is different than other R-type instructions.

d) **[10 Points]** Now make the changes in the code to support the "jr" instruction whose necessary datapath and controller changes are given in the "JR Changes" file.

e) **[10 Points]** Now make a SystemVerilog testbench file on EDAPlayground and simulate your MIPS-lite processor executing the test program (Refer to *How to Visualize Waveform in EdaPlayground*). Do not change the instructions in the imem module use it as is. Study the generated waveform. Find each instruction, and understand its values. **Take a screenshot of the waveform and paste it to the report**. Your waveform should be similar to the below image (see Figure 2, some part of the image is obfuscated intentionally). Make sure that your output shows the exact same variables. Be aware that if you run the "Complete MIPS Model.txt" without implementing "jr" instruction you will see continuous X values towards the end.

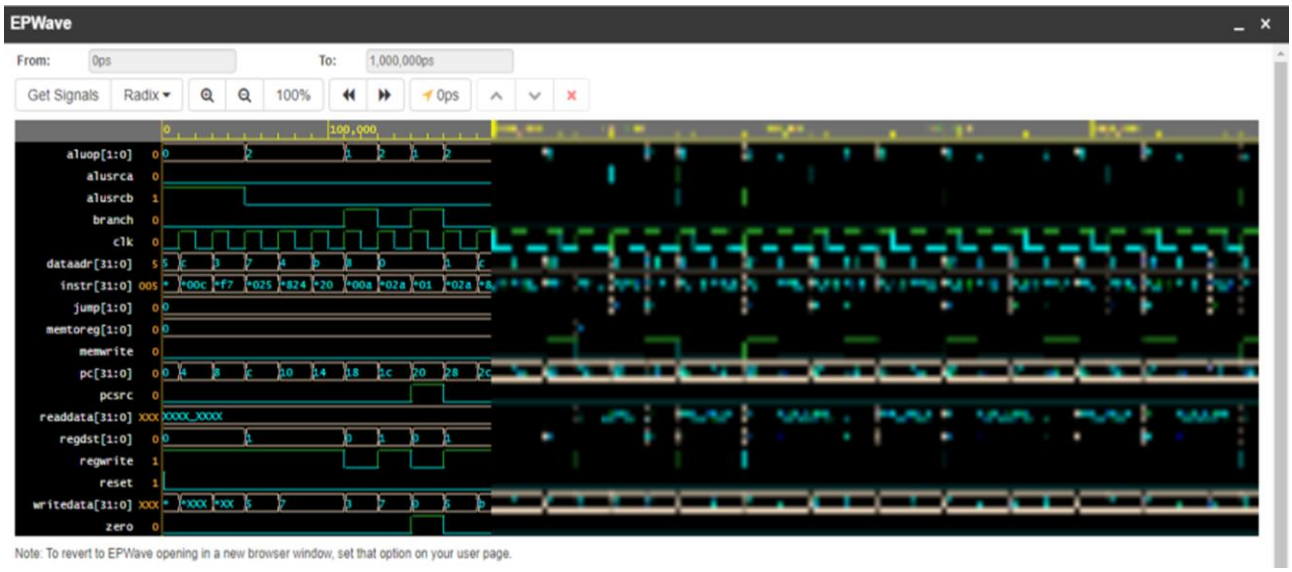

Figure 2: Example Waveform

f) **[5 Points]** Make some observations on the waveform that you generated. These questions assume that you have used imem module as is without modifying any instruction there.

i) In an R-type instruction what does writedata correspond to?
ii) Why is writedata undefined for some of the early instructions in the program?
iii) Why is readdata most of the time undefined?
iv) In an R-type instruction what does dataadr correspond to?
v) Why does dataadr become undefined at some point?

g) **[2 Points]** Make some observations on the architecture (see Figure 1).

i) Do you need to make any changes to support **srlv** instruction?
ii) Which module would you modify to support **sll** instruction? How would you modify it?

Part 2.

Section	MIPS instructions
1	Base: Original12 + "jr" New: "sw+"
2	Base: Original12 + "jr" New: "subi"
3	Base: Original12 + "jr" New: "jalm"
4	Base: Original12 + "jr" New: "bge"
5	Base: Original12 + "jr" New: "ble"
6	Base: Original12 + "jr" New: "jm"

Table 3: Section-wise distribution of the instructions

Instructions in quotes (e.g. "subi", except for "jr") 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.

subi: this I-type instruction subtracts, using a sign-extended immediate value. Example: 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, **ble**: 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

sw+: 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.} Example: sw+ $t0, 4($t1)

a) **[2 Points]** Register Transfer Level -Language- (RTL) expressions for the new instruction that you are adding (see Table 3), including the fetch and the updating of the PC.
b) **[10 Points]** 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. **Make your changes on "Final Datapath.png" file.**
c) **[8 Points]** Make a new row in the main decoder table for each new instruction being added, and if necessary, add new columns for any new control signals that are needed (input or output). You can use any opcode value you want unless it conflicts with the existing opcodes in the Table 1.

Be sure to completely fill in the table—all values must be specified. **Make your changes on the Main Decoder Table at the "JR Changes" document.** If any changes are needed in the ALU decoder table (Table 2), give this table in its new form (with new rows, columns, etc). The base table should be in black, with changes marked in red. {Note: if you need new ALUOp bits to encode new values, you should also give a new version of Table 2, showing the new encodings}

Part 3. Lab Work (50 points)

For the lab work, you only need to submit resulting code file in the txt format.

Part 3.

a) [**50 Points]** Now make necessary changes to your "Complete MIPS model.txt" code so that it supports the instruction assigned to your section (refer to Table 3, you already implemented "jr" in the form of a prelim work, you ought to implement the section-specific instruction for this part). You will code up the changes that you proposed in the Part 2. Note that the Original12 and "jr" instruction should still work. That is, at this stage your MIPS architecture should be able to support Original12, jr and the assigned instruction.

b) You can prepare a test MIPS program yourself to see if your architecture works correctly. In the lab day, we will provide you with our test program and grade you based on that.

Aim to complete lab part at home before coming to the online lab.

Part 6. Submit Your Code 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. Only a NOTEPAD FILE (txt file) is accepted. 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 !

Part 7. Lab Policies (Reminder)

1. As indicated in Lab1: Attendance is mandatory and the preliminary work is graded only if you submit your lab work with the observation/permission of your TA.
2. You can do the lab only in your section. Missing your section time and doing in another day is not allowed.
3. The questions asked by the TA will have an effect on your lab score.
4. 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.

Your lab attendance is tracked by the Zoom system. Please also see lab policies no. 1 item.