ECE 585 – Computer Organization and Design
Fall 2019

Project 2 – MIPS CPU Design and Implementation (Pipeline)
Report Due: Friday, December 6th 2019, 11:59PM

IMPORTANT:	***You must sign and date below acknowledgment statement on the title page of your report.*** **Failing to do so, or any violation of this rule will result in an automatic failure for this course.**
Acknowledgment:	I acknowledge all works including figures, codes and writings belong to me and/or persons who are referenced. I understand if any similarity in the code, comments, customized program behavior, report writings and/or figures are found, both the helper (original work) and the requestor (duplicated/modified work) will be called for academic disciplinary action.

I. Introduction

The purpose of Project 2 is to design and implement your own custom 32-bit RISC processor, a stripped-down MIPS processor. The goal of this project is to provide you a more practical, hands-on approach to computer architecture design problems. For this project, you can either work on your own, or work in groups of two.

The processor you are required to design is a 32-bit version of the MIPS processor with a minimal instruction set from the MIPS ISA. You should implement the pipeline datapath version of the processor utilizing the VHDL hardware descriptive language. You may use any construct within the VHDL language, however, the design must be of your own. Copying of any form from any source is illegal and will not be accepted under any circumstances.

Your customized processor is required to support three instruction types: R-type, I-type and J-type as described in the textbook and lecture slides. Table I summarizes a minimum set of instructions for your required ISA. Your memories should be word-aligned/word-addressed where each word is 32 bits.

II. Implementation Details

The best way to accomplish this is to implement the datapath first without the memory and control interface, along with your design's ability to process 5-stage MIPS pipeline where each stage will be separated in equal amount of clock cycle time. 5-stage MIPS pipeline will have the following steps per stage:

1. IF: Instruction fetch from memory
2. ID: Instruction decode & register read
3. EX: Execute operation or calculate address

4. MEM: Access memory operand
5. WB: Write result back to register

You may choose your own clock cycle time per stage, but they need to be equally assigned per stage. If you work with abstraction in mind by initially testing each lower-level part completely, it will eliminate potential errors within your design later.

Table I shows the minimum subset of the ISA that your processor should support, and execute successfully. You are responsible for designing the datapath and control logic additions/modifications for the instructions in pipeline shown in Table I.

Table I. Required MIPS Instruction Set

Code Sequence	OpCode [31:26]	Function Field [5:0]	Instruction	Operation (example)
1	000000	100010	sub	sub $2, $1, $3
2	000000	100100	and	and $12, $2, $5
3	000000	100101	or	or $13, $6, $2
4	000000	100000	add1 (RCA)	add1 $14, $2, $2
5	000000	100000	add2 (CLA)	add2 $14, $13, $2
6	100011	-	lw	lw $15, 100($2)
7	001000	-	addi	addi $3, $5, 200
8	000010	-	j	j 2500

ALU implementation can be achieved in similar method as Project 1. Start with a 1-bit ALU cell and construct a 32-bit ALU unit. Note that you may have to modify the existing 1-bit ALU cell design to provide more functions due to the extended ISA in your CPU design. For your feature of the ALU, you must add extra control signal(s) to use your Ripple Carry Adder *(add1)* and Carry Lookahead Adder *(add2)* from Project 1.

Next you should implement the control logic. It is recommended that the control be assembled next using an FSM (Finite State Machine). Using/adding/modifying the control unit shown in Figure 1, develop an adequate control implementation using an FSM. In summary, your processor must support basic MIPS code.

The following guidelines will help you achieve your goal.

- Each datapath element must be designed from scratch. That is, modifying the multi-cycle MIPS should be done utilizing your groups code only. Intellectual Property (IP) or other publicly-available code for the datapath and control cannot be used for this project

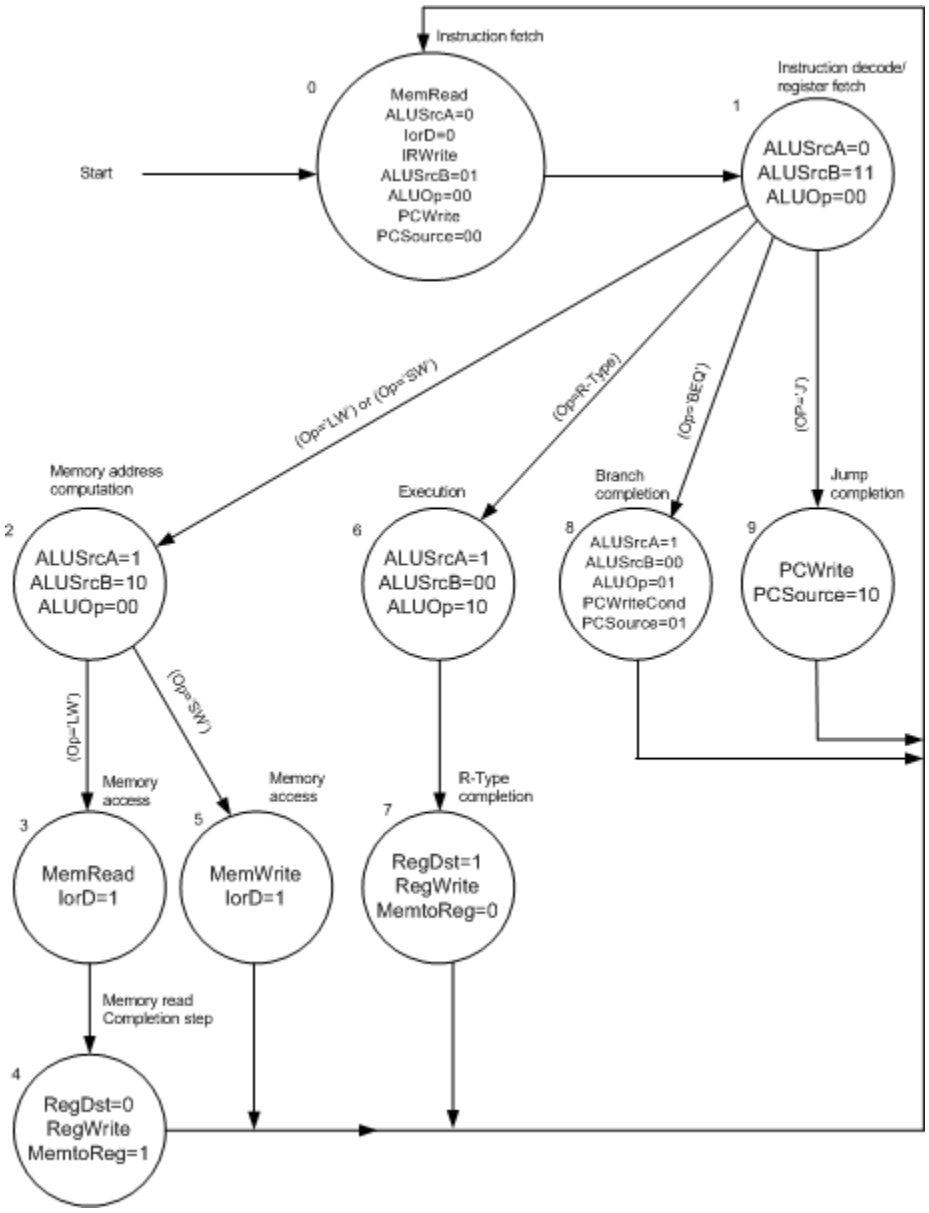

Figure 1. Finite State Machine for Control Logic

- You are not required to get area and delay numbers for your project. However, you should be able to determine the critical path for the cycle time by making some assumptions. This should be noted in your project report.

- You are required to show all 8 instructions complete executions in the simulation result. This means, you must show all clock cycle times of your simulation results from the first clock cycle time of Code 1 (sub $2, $1, $3) to the last clock cycle time for Code 8 (j 2500).

- The datapath handles 32-bit numbers.
- The base datapath is shown in Figure 2 for your reference. However, it is also available in your textbook and lecture slides. You will modify this datapath in order to extend the instruction subset.
- The control logic should probably be very similar to the multi-cycle control logic and FSM described in Figure 1. However, you need to modify this state diagram for the additional instructions.

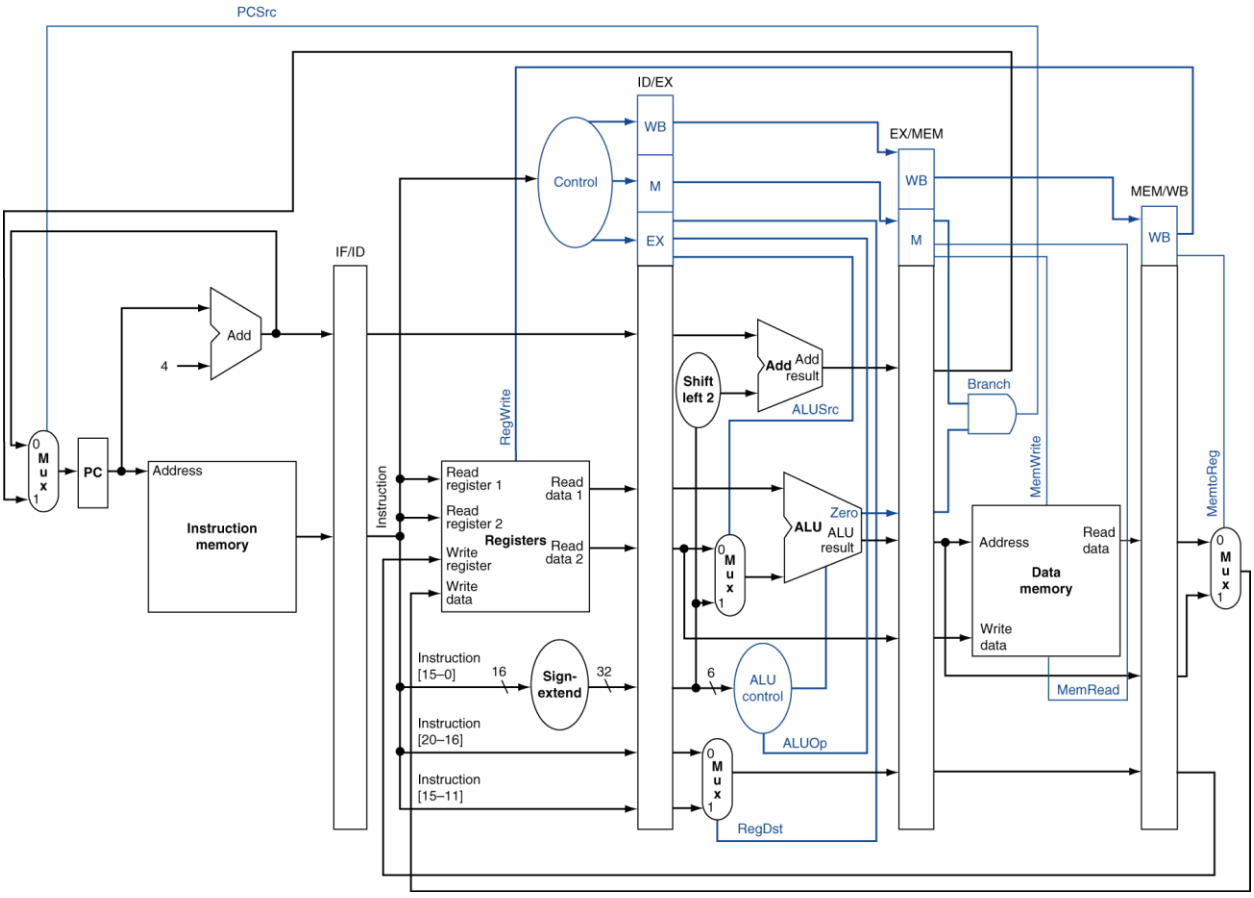

Figure 2. Datapath with Pipelined Control

III. Project Report

1. You are required to provide your original VHDL codes, as well as your testbench codes that you have used to verify your VHDL codes. Both codes should contain your original comments.
2. You are required to provide simulation results by capturing the screen of the simulation output. Your simulation result must contain all 8 instructions from Table I in the code sequence, along with input data used and corresponding output data. You must clearly state in all captured figures which input data used and corresponding output data.

3. You must provide results, discussions, screenshots, source codes, testbench codes.
4. Your report should include the following sections:
    a. Title Page with Acknowledgment and your Signature
    b. Abstract of your report
    c. Introduction *(please remember, introduction and abstract aren't the same)*
    d. System Design
        i. Description of how you come up with your design
        ii. Description of how you designed the datapath
        iii. Description of how you designed the control logic
        iv. Overview block diagram of your design (system flowchart)
        v. Any other description that you'd need to describe your design
    e. Simulation Results and discussion
        i. Descriptions of your test/example cases to prove your design works correctly
        ii. Descriptions of each screenshots
        iii. Discussion on any improvements or additional features made to your design
        iv. Discussion of how you tried to optimize your design
        v. Discussion of any issues that you faced, any improvement could be made, what and why does not work correctly in your design
    f. Conclusion
    g. List of references
        i. Write one short paragraph about each of the references and its relevance to completing your project
    h. Appendix
        i. Entire Source Code of your project with comments
        ii. Entire Testbench Codes with comments that used to verify your design

IV. Project Requirements

1. You are required to design and implement this project **individually or in groups of two**. *Your group member must be registered to ECE 585.*
2. Only one report is required per group. However, each member of the group must write at maximum of 1 page of their contributions to the project. This must be signed by all group members.

3. Due date is **Friday, December 6th 2019 11:59PM**. *No late submission will be accepted*. You'll need to submit the following package in a single ZIP file to the Blackboard.
    a. Your Project Report
    b. Project Contributions Report (only required for groups of two)
        i. Maximum of 1 page of their contributions to the project
        ii. This report must be signed by both group members
    c. Your VHDL Source Codes with your own comments
    d. Your VHDL Testbench Codes with your own comments
4. Under no circumstances, copying of codes from anyone else outside of yours or your group will not be accepted, and will result automatic 0 grade for your Project 2 (including your partner if working in groups). Thus, do not impact others with improper choices of your code.
5. It is strongly recommended designing your code as soon as possible since this project is large and starting early is prudent.

V. Extra Points (Only applies if all requirements completed) +20 Points Max

Extra points work should be only attempted if all requirements defined in Section II are completed and will not be graded if submitted without the completion of Section II requirements. You can improve your processor (and your project grade) by:

- Adding below additional instructions within the MIPS ISA or instructions that would fit your design. These instructions must be in sequence with the examples given in Table I – **Up to 10 additional points *(2 points per instruction)***

Code Sequence	OpCode [31:26]	Function Field [5:0]	Instruction	Operation (example)
9	101011	-	sw	sw $t1, 100($t2)
10	000000	101010	slt	slt $t1, $s2, $s9
11	000000	100111	nor	nor $s1, $t2, $s3
12	001100	-	andi	andi $t2, $s2, 100
13	001101	-	ori	ori $t1, $s2, 100

- Implementation of Forward Unit and Hazard Detection Unit. In this case, you must show how your implementation overcomes hazards with examples of MIPS codes. – **Up to 5 additional points**
- Implementation of Exceptions including realization of EPC and Cause registers, as well as the control modification. For this implementation, you must provide examples of codes to prove your implementation. – **Up to 5 additional points**