ECE 443/543

Final Project

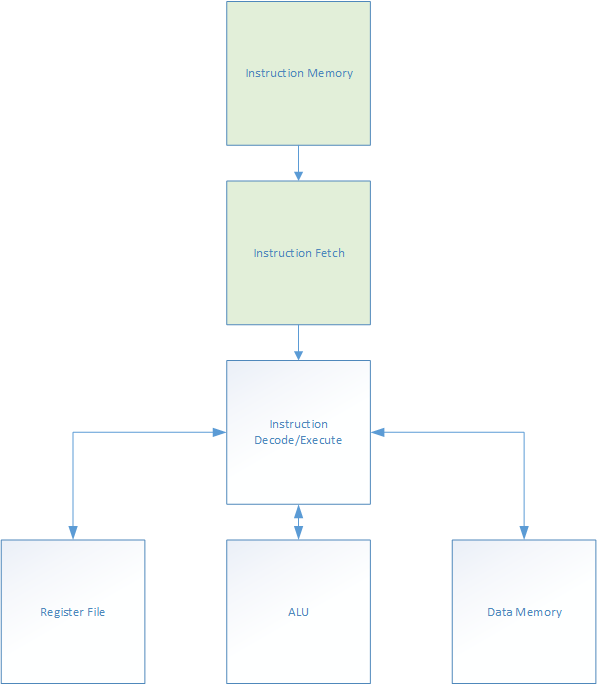
**General Guideline**

1. **Due on December 7, 5PM**
2. **For ECE 543 students, this is an individual project. For ECE 443 students, this is a two people project. For ECE 443 students, only one of your team members need to submit your work to Blackboard.**
3. **You will receive 20 points deduction for every 12 hours late submission**
4. **You need to submit your 1 VHDL Workspaces and electronic version of your report via Blackboard (see the Section of Deliverable for details). You have to zip them into one file and clearly name and organize them.**
5. **For your report, you need to have one cover page to identify name, ECE 443 or ECE 533 student, and your team member (if applicable).**
6. **You need to put the comments to your code**

**ALDEC will be used, not Quartus.**

**Objective**

For the final project, you will be implementing a simple processor. The processor will support 7 different instructions, two instruction formats, support 256 half words of RAM (256 addresses by 16 bits) and have 8 registers. You should be able to modify the ALU you implemented for Project #2 to use with this project. The following figure shows the high level overview of this processor.



*High level overview of the processor*

For this lab, you will be implementing a processor. The basic design is a single cycle processor which implements the Instruction Decode/Execute, ALU, Register File and Data Memory sections. Each instruction is 16 bits wide. There are two formats, R-Type (register) and I-Type (immediate). Their formats are shown below. R-Type instructions perform an operation using the registers specified in a and b portions. The result is stored in the register specified by c portion. I-Type instructions specify an immediate value. In this project, the ldi instruction is the load immediate instruction. It loads the immediate value into the register specified by d portion. The sh and lh instructions store and load halfwords into the memory address specified by the immediate value.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| R-Type | | | | |
| unused | opcode | c | a | b |
| 1 bit | 3 bits | 4 bits | 4 bits | 4 bits |

|  |  |  |  |
| --- | --- | --- | --- |
| I-Type | | | |
| unused | opcode | d | value |
| 1 bit | 3 bits | 4 bits | 8 bits |

The opcodes for the different instructions are specified below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S0 | S1 | S2 | Instruction Format | Operation | Mnemonic |
| 0 | 0 | 0 | R | Signed Addition | add |
| 0 | 0 | 1 | R | Signed Multiplication | mult |
| 0 | 1 | 0 | R | Passthrough A | pa |
| 0 | 1 | 1 | R | Passthrough B | pb |
| 1 | 0 | 0 | R | Signed Subtraction | sub |
| 1 | 0 | 1 | I | Load Immediate | ldi |
| 1 | 1 | 0 | I | Store halfword | sh |
| 1 | 1 | 1 | I | Load halfword | lh |

**Requirement:** Both theregister file and the RAM/memory can be implemented behaviorally using arrays. The instruction decode/execute is implemented using structural model.

Your processor should execute the following program.

ldi $r0, 10

ldi $r1, 5

ldi $r2, 0

ldi $r3, 0

ldi $r4, 0

ldi $r5, 0

ldi $r6, 0

ldi $r7, 0

add $r2, $r0, $r1

mult $r3, $r0, $r1

sub $r4, $r0, $r1

sh $r3, 0x0B

sh $r4, 0x0A

lh $r6, 0x0A

lh $r7, 0x0B

The following is the program compiled into hexadecimal.

500A

5105

5200

5300

5400

5500

5600

5700

0201

1301

4401

630B

640A

760A

770B

**Grading/Deliverables**

**Screenshots must be readable!**

* Implement the ALU (15 pts). Generate a testbench and show that it works for addition, subtraction, multiplication, passthrough A and passthrough B by including screenshots. In the report, you also need to explain how you design your ALU, such as, circuit, etc.
* Implement the 8 x 16 bit register file (15 pts). Generate a testbench and show that you can store and retrieve 16 bit values. Generate a testbench and show that it works using a readable screenshot. In the report, you also need to explain how you design your register file.
* Implement the instruction decoder (15 pts). Generate a testbench which shows it operating and include a readable screenshot. In the report, you also need to explain how you design your instruction decoder, such as circuit, etc.
* Implement the 256 address x 16 bit RAM (20 pts). Generate a testbench and show that you can store and retrieve values from it. Include a screenshot. In the report, you also need to explain how you design your RAM.
* Show the processor works (20 pts). Generate a testbench to run the above program. At the end of the program, what are the values in $r6 and $r7? Write these values in your lab report.
* Implement the instruction memory and instruction fetch cycle (15 pts). Generate a testbench and show that it works by using a screenshot.

The deliverables for this lab are the report and the code. Your code should be put into a zip file and sent to the instructor.