

For full and partial credits, you must show your work and explain all steps.

For this lab, you are required to work in a team of two students to design and simulate a 16-bit single cycle RISC processor. The aim of the lab is to understand the basic principles of how a RISC processor works and how the instructions are used to control different parts of a data path through a control unit.

Design Steps

1. Design the Instruction Sets format:
 - a. Support R-type, I-Type, and J-type instructions
 - b. The minimum number of instructions: 15 instructions
 - c. Your team can decide the instructions based on the test program (see the test steps).
2. Design Datapath components and define clocking methodology (rising or falling edge-trigger).
 - a. Memory, ALU, Multiplexers, Register File, etc.
3. Assemble Datapath
 - a. Memory: The memory is 512 bytes to store data and instructions as binary machine codes for convenience.
 - b. ALU functions: add, sub, and, or, slt, etc. (do not use instructions for multiply or divide)
 - c. Number of registers (register file): your team can determine the number of registers (8 or 16)
4. Analyze the implementation of each instruction: Control signals
 - a. Refer to the class slides and determine your control signals to implement.
5. Assemble Control Unit
 - a. Control Unit inputs: opcode
 - b. Control Unit outputs: control signals
6. Write your simulator (program) with either VHDL (default) or software tools (C/C++, Java, Python) if I give you permission for that.
 - a. Write the code for the entire Datapath, including forwarding mechanisms
 - b. Write the code for the Control Unit

Timeline and Grades

1. Progress Reports and Demos (**Starts: October 15 and updated every week**): 40%
 - a. Your team should send a progress report of your work by email as you progress.
 - b. Teams should demonstrate their progress during the lab each week, as follows:
 - i. Week 1 (October 22): Complete the design sheet
 - ii. Week 2 (October 29): Test individual units, complete the Datapath, test the composite Datapath.
 - iii. Week 3 (November 5): Design and build the control unit. Integrate the control unit with the Datapath.

iv. Week 4 (November 12): Test the complete design and run a demo for the provided pseudocode.

2. Final Demonstration (During the Lab of Nov 12): 10%

- a. Simulator demonstration
 - i. Run your simulation program with input data.
 - ii. Show output data.

3. Final Project Report (Due date: Nov.12): 50%

- a. Your team must prepare and submit the *final project report* by email (see the *Final Report Format* at the class homepage). Late final report will not be accepted.
- b. Submit your *contribution report* to Peter with all team members' signatures (see the *Contribution Report Format* from the class homepage.)

Testing Steps

1. Run your simulator with the following test program.

```

$v0 = 0040hex; // you can redefine $v0-3, $t0, and $a0-1 with
$v1 = 1010hex; // your register numbers such as $1, $2, etc.
$v2 = 000Fhex;
$v3 = 00F0hex;
$t0 = 0000hex;
$a0 = 0010hex;
$a1 = 0005hex;
while ($a1 > 0) do {
    $a1 = $a1 -1;
    $t0 = Mem[$a0];
    if ($t0 > 0100hex) then {
        $v0 = $v0 ÷ 8;
        $v1 = $v1 | $v0;          //or
        Mem[$a0] = FF00hex;
    }
    else {
        $v2 = $v2 × 4;
        $v3 = $v3 ⊕ $v2;          //xor
        Mem[$a0] = 00FFhex;
    }
    $a0 = $a0 + 2;
}
return;

```

Initial data (hexadecimal) in Memory

Mem[address]	Data
Mem[\$a0]	0101 _{hex}

Mem[\$a0+2]	0110 _{hex}
Mem[\$a0+4]	0011 _{hex}
Mem[\$a0+6]	00F0 _{hex}
Mem[\$a0+8]	00FF _{hex}

2. Procedure to run your simulator:
 - a. Clear contents of memory and register file initially
 - b. Translate the pseudo-code into MIPS assembly language and binary machine code (refer to the attached design sheet).
 - c. Put the binary machine code into the memory and set the PC value with the address of the 1st instruction; if the address is 010hex (the maximum size of the memory is 512 bytes; 200hex), the PC value should be 0010hex (the register size is 16 bits) and will be increased by 2, 2 bytes, during the fetch stage.
 - d. Print initial contents, at the clock cycle 0, of your memory and register file (refer to the *Final Report* format at the class homepage).
 - e. Execute the code instructions step by step.
 - f. Print the contents of the memory and register file after each loop, one output per loop (refer to the attached *lab 4 template*).

Bonus Parts:

1. Implement three new instructions that are not part of the MIPS instruction as part of your design 10%
2. Implement the quick sort algorithm using your MIPS design. Show the outcome of sorting two random integer arrays of sizes 5, 10, and 20. 20%
3. (Only if you use VHDL) Make the code synthesizable and run it on an FPGA board. 20%
4. (Only if you use C/C++, Java, Python) build a graphical user interface for your MIPS design with the basic features of the MARS4 MIPS simulator 20%
5. Extend the design to a pipelined processor that deals with hazards using software-inserted NOPs (No forwarding units needed) 40%

Deliverables: Design sheet and lab report based on the attached templates.

Submission: You need to submit your report and code to the assignment list on Canvas. You **must** structure your submission as follows:

- One PDF file for your report (your code must also be included in the report as an appendix)
- One ZIP file containing your code

Include each file in your submission. Canvas allows you to include multiple files in your submission.