Ain Shams University - Faculty of Engineering
Computer and Systems Engineering Dept.
Dr. Cherif Salama
Eng. Diaa El-Din Mohamed

CSE 311: Hardware Organization (2)
Fall 2015

Project Description

Project Name: MIPS Processor

Grading weight: This project accounts for 20 marks (13 for the implementation and 7 for the report). A bonus up to another 10 marks will be given for implementing at least **two** of the *bonus* features suggested below. Please do not be tempted to implement more than two bonus features, as this will cost you too much time.

Project Overview: The goal of this project is to write and test a simplified Verilog description of the **single-cycle** MIPS datapath. The description should be as close as possible to the one shown in Slide 20 from Lecture 1, but modified as needed to support the required instructions. This document details the instruction set you should support, the description style, the inputs needed for the simulation, and the expected outputs.

Implementation language and tools: Verilog must be used to describe the entire MIPS datapath; however, you are free to use any Verilog simulator of your choice.

Team Size: 2 to 6 students

Important Plagiarism notice: You have to write your own code from scratch. Projects based on others code will receive a grade of **zero** in the entire project and report (even if the code is heavily re-factored/modified, etc…). Examples of such sources include (but is not limited to) code copied from the following sources: other teams, previous year projects, open-source software (or Internet in general), tutors, etc…

Project Deadline: Sunday, December 27th, 2015 at 11:59 pm. You should send your project and report to your TA (diaaeldin.osman@eng.asu.edu.eg). *If needed*, a meeting with the TA to evaluate your work will be scheduled in the following days.

Instruction set architecture (ISA): Your description must support the following set of instructions:

- **Arithmetic:** `add, addi`
- **Load/Store:** `lw, sw`
- **Logic:** `sll, and, andi, nor`
- **Control flow:** `beq, jal, jr`
- **Comparison:** `slt`

Verilog Description: At the top level, your description must consist of one **fully-structural** module describing the modified version of the MIPS datapath in Slide 20 from Lecture 1. This module **should not have any ports at all**. This top-level module will of course need to instantiate several other modules that should be described either structurally, behaviorally, or a mixture of both (depending on your own preference). In all cases, for the simulation to be realistic, you should assume reasonable delays for each hardware component you use and you should use a clock whose period is big enough to accommodate the delay of the critical path of your design.

Simulation inputs: The description should be tested using binary programs and binary data (when needed). One simple way to do so is to assume that both are stored in a single memory and to store its contents in binary or hexadecimal (based on your own preference) in a text file. This file should be loaded in the beginning of the simulation using an `initial` block into both memories. Please note that the binary program must start from the same location initially pointed to by the program counter (PC).

Simulation: The description should be simulated till the program finishes its execution. This requires the simulation to continue for a number of clock cycles equal to the actual number of instructions that will be executed. No other inputs (other than the program being executed and its data) should be given during the execution.

Simulation output: At the end, the simulation should display the final state of the datapath. The final state is composed of the final value of the PC, the final values of all 32 MIPS registers, and the final value of all the data memory locations involved in the program execution.

Project Report: In addition to your team member names, the report should include:

1. A brief description of your implementation including any bonus features included. This should also include any tools/languages used for implementation or simulation.
2. The datapath you have used including any necessary extensions to support all the required instructions.
3. A summary of how the work was split among your team members (who did what exactly)
4. A user guide including a full simulation example step-by-step with snapshots.
5. A list of programs (and associated data if any) you simulated. You should at least provide 3 programs. The programs must cover all instructions supported and one of them at least must have a loop. Each program should be provided in the report both in assembly **and** in binary/hexadecimal
6. For each program, state the number of clock cycles needed to simulate it to completion and describe its expected final state (obtained by manually tracing the program).
7. For each program, show the actual final state obtained by running the simulation (with snapshots). This state should be identical to the expected final state.
8. Optionally, you can include a section about your experience working on this project. This section will NOT affect your grade in any way.

Ain Shams University - Faculty of Engineering
Computer and Systems Engineering Dept.
Dr. Cherif Salama
Eng. Diaa El-Din Mohamed

***Bonus* features:**

1. Write and test a simplified Verilog description of the **pipelined** MIPS datapath. The description should handle all kinds of hazards correctly. This bonus counts as TWO bonus features. **Do not implement any other bonus features** if you implement this one.
2. Support the following instructions as well:
   - **Arithmetic:** `sub, mul`
   - **Load/Store:** `lh, lhu, lb, lbu, sh, sb, lui`
   - **Logic:** `srl, or, ori`
   - **Control flow:** `bne, j`
   - **Comparison:** `slti, sltu, sltui`
3. For each of the components of the datapath, provide both a fully structural and a behavioral description and verify that they give the same simulation results.
4. Instead of just displaying the final state of datapath after the simulation ends, find a way to show the state of the datapath each cycle during the simulation. Include at least one example of that in your report.
5. Implementing a simple assembler to allow the user to supply programs in assembly language. The assembler in this case would be a program to translate instructions (and data if any) into the appropriate binary or hexadecimal format and saving it to a file that can be used in your simulation. You are free to use any general purpose programming language (C, C++, Java, C#.NET, etc…) to implement this assembler. A user guide should be included in the report.
6. Provide a relatively large set of test programs of different levels of complexity (not less than 12 programs instead of only 3 programs).
7. Implement the project in VHDL too.

What and how to submit: Send a compressed folder by email to your TA and to myself. The folder must contain: 1) All your code (Verilog description, assembler, VHDL description, etc…), 2) The files representing your test programs and data, and 3) Your report (PDF).