Ain Shams University
Faculty of Engineering
CESS Program
CSE116: Computer Architecture

Project Description (Spring 2018)

Project Name: MIPS datapath and control unit simulator

Grading weight: This project accounts for 20% of the course mark and will be graded out of 20 marks (15 marks for the implementation and 5 marks for the report). A bonus up to 5 marks will be given for implementing at least **two** of the *bonus* features suggested below. Please do not be tempted to implement all the bonus features, as this will cost you too much time.

Project Overview: The goal of this project is to implement a low-level MIPS datapath and control unit **simulator**. Datapath simulation includes simulating all its components (registers, memories, ALUs, multiplexers, shifters, sign extenders, and the connections among these). This document details the supported instructions, the inputs to the simulator, and the expected outputs.

Implementation language: Any general purpose programming language. Java is of course recommended. The resulting application can either be a console application or a graphical user interface (GUI) application as a *bonus* feature.

Team Size: 1 to 5 students (preferably 2 or more)

Important Plagiarism notice: You have to write your own code and your own report 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: Thursday May 10th 2018, 11:59 pm. Projects and reports should be submitted via the LMS (only one student from each team should submit the project). *If needed*, a meeting with the course TA to evaluate your work will be scheduled in the following days.

Instruction set architecture (ISA): The simulator should support the following MIPS instructions:

- **Arithmetic:** `add, addi`
- **Load/Store:** `lw, sw, lb, lbu, sb`
- **Logic:** `sll, nor`
- **Control flow:** `beq, j, jal, jr`
- **Comparison:** `slt, slti`

Simulator inputs:

1. *Assembly program:* The user should be able to input a program (using the supported instructions only) to be simulated. He should also specify its starting address (where the program's first instruction should be loaded in the memory).
2. *Program data:* The user should also specify any data required by the program to be initially loaded in the memory. For each data item both its value and memory address should be specified.

Simulation and simulation outputs: The simulator should assume a single cycle datapath identical to the one presented in lecture 7 (slide 5) except where it needs to be extended to support the extra instructions required. Your program should simulate the program execution showing the values (possibly in decimal and/or hexadecimal) carried by **all the wires** of the datapath in **each** clock cycle. The simulator should also keep track of the **register file contents** and **memory contents**. Finally, while simulating the execution, your program should also keep track of the number of clock cycles spanned.

Please note the following:

1. We will pretend to have separate instruction and data memories for timing purposes; however, to keep our simulation simple, we will assume that there will be a single common memory underlying both (we will assume that both have the same address space and that they always have the same contents).
2. Register 0 always contains the value 0. You will need to make sure you initialize register 0 appropriately and make sure any attempt to modify it does not succeed.

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.
2. The datapath you have used including any necessary extensions to support all the required instructions.
3. The truth table of the control unit and its logic diagram.
4. Any assumptions you adopted that are not mentioned in this description (if any).
5. A user guide including a full simulation example step-by-step with snapshots.
6. A list of programs (and associated data if any) you simulated. You should at least provide 2 programs. The programs must cover all instructions supported and one of them at least must have a loop.
7. A summary of how the work was split among your team members (who did what exactly)
8. Optionally, you can include a section about your experience working on this project. This section will NOT affect your grade in any way.

***Bonus* features:**

1. Building the application as a GUI application
2. Building the application as an educational GUI application. In this case the GUI should include an animated diagram of the datapath illustrating how the wire values propagate through it each clock cycle. This bonus feature is highly recommended

and will be counted as two features as far as grading is concerned (since it includes and extends the previous bonus feature).

3. Implementing and integrating an assembler to allow the user to supply programs in a suitable assembly language. The assembly code can either be entered directly in your program or in a text file read by your program. The assembler should support labels, pseudoinstructions (like `move` and `blt`), and directives (like `.data`, `.word`, and `.text`).
4. Support the following instructions as well:
    - **Arithmetic:** `sub, mul`
    - **Load/Store:** `lh, lhu, sh, lui`
    - **Logic:** `srl, and, andi, or, ori`
    - **Control flow:** `bne`
    - **Comparison:** `sltu, sltui`
5. Including a larger set of programs (at least 6 meaningful programs) and their equivalent C programs.
6. Provide a simulator for the **pipelined** datapath of MIPS based on the diagram in lecture 8 slide 29. Note that this datapath does not detect or handle hazards at all (which means that the input program has to be hazard-free to produce correct results). Your simulator should simulate the program execution showing the contents (decimal and/or hexadecimal) of all the fields of the pipeline registers (PC, IF/ID, ID/EX, EX/MEM, and MEM/WB) including control signals at **each clock cycle**. Any control signals that are not part of the pipeline registers (e.g., the PCSrc) should also be displayed at each clock cycle. This feature will be counted as two bonus features as far as grading is concerned.

What and how to submit: Submit a compressed folder (zip or rar) on the LMS. The folder must contain: 1) All your code, 2) The files representing your test programs and data, and 3) Your report (PDF).