GUC
German University in Cairo
الجامعة الألمانية بالقاهرة

Faculty of Media Engineering and Technology
Dept. of Computer Science and Engineering
Dr. Cherif Salama
Eng. Amal Rizkallah
Eng. Jailan Salah
Eng. Lydia Mamdouh

CSEN 601: Computer System Architecture
Spring 2015

Project Description

Project Name: Pipelined MIPS datapath simulator

Grading weight: This project accounts for 25% of the course mark (15% for the implementation and 10% for the report). A bonus up to 5% **of the project mark** 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 cycle-accurate pipelined MIPS datapath simulator. Simulating the datapath includes simulating all of its storage components (register file, memories, and pipeline registers) and all of its control signals. This document details the supported instructions, the inputs to the simulator, and the expected outputs.

Implementation language: Any general purpose programming language (preferably an object-oriented language like C++, Java, or C#.NET). The resulting application can either be a console application or a graphical user interface (GUI) application as a *bonus* feature.

Team Size: 1 to 6 students (preferably 2 or more)

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 coming from the following sources: other teams, previous year projects, open-source software, tutors, etc…

Project Deadline: Wednesday April 22nd, 2015. Evaluations will take place immediately afterwards.

Instruction set architecture (ISA): The simulator must support the following MIPS instructions ***only***:

- Arithmetic: `add, addi, sub`
- Load/Store: `lw, lb, lbu, sw, sb, lui`
- Logic: `sll, srl, and, nor`
- Control flow: `beq, bne, j, jal, jr`
- Comparison: `slt, sltu`

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 pipelined datapath identical to the one presented in slide 21 of lecture 8 except where it needs to be extended to support the extra instructions required. The studied datapath uses a hardwired control unit (built as a simple combinational circuit). 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. The simulator should also keep track of the **register file contents** and **memory contents**. Finally, the simulator should keep track of the number of clock cycles spanned during the execution of the input program.

Please note the following:

1. We will pretend to have separate instruction and data memories for timing purposes and to avoid structural hazards; 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 design of the hardwired control unit you have used (a logic diagram of the control unit).
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 3 programs. The programs must cover all instructions supported and one of them at least must have a loop and one of them at least must have function call. You should tweak all programs so that they execute correctly despite the lack of hazard support.

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 pipeline registers and control signals are updated each clock cycle. The animated diagram should clearly show which instruction is in each of the stages of the pipeline. 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 can 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. Providing data and control hazards support (forwarding unit, hazard detection unit, and branch prediction). In this case, your simulator should show the values of the new control signals at each clock cycle. Your report should include a new list of programs (at least 3) featuring hazards. This feature will be counted as two bonus features as far as grading is concerned.
5. Supporting the simulation of the single cycle implementation of MIPS and comparing the time needed to execute the provided programs on this single cycle implementation with the time needed to execute them on the pipelined implementation. In this case the user should provide the delay for each of the components of the datapath so that the simulator can infer the clock cycle needed for the pipelined implementation and for the single cycle implementation.