ELEC2204 Computer Engineering

Coursework: Computer Simulation

Introduction

Simulators, such as MAI ploring the design of computer systems. They can implement the function f a computer in software, and may be used to make a computer run software ed for another machine. They may also allow internal signals to be visualised is is a relatively resource-intensive task, it allows much finer-grained debugging and analysis of a computer's operation than, for example, adding a stop-point to code and reading memory contents.

In this coursework, you are going to develop a simulator to represent the operation of a simple MIPS-like computer (including the processor and memory). *Simple* is the key word here: you need to define, implement and test the capabilities of each of your system's modules, and it should all connect together to work as a computer. It only needs to implement a very limited instruction set.

Specification

This coursework contributes 15% of the mark for ELEC2204, so should take you around 25 hours to complete. Your processor should (as a minimum):

- Implement the basic functionality of a **single-instruction, multi-cycle** computer *in software.*
- Run machine code, which should be read in from a separate text file.

- Fetch and execute instructions from memory, without caches/virtual memory.
- Follow a straightforward *fetch/decode/execute/write-back/memory access* cycle. This can be implemented using a state machine.

- Implement a limited instruction set and register topology, which you may define yourself (you should choose to implement a subset of MIPS instructions). However, you **must not** implement a multiply or divide instruction in your processor!
- Be written in C or C++ (with comments), with the source code handed in with your report.
- Be tested: you should first test each module, and then write a set of basic test programs that allow you to check that your processor is working as intended; you should also be able to capture the operation of the system, e.g. logging the instructions issued to the processor and the total number of instructions executed.

The composition of any initial test programs - for use in developing your system - are up to you.

However, included in your report should be details of a specific 'showcase' program: to calculate (and save to an array in memory) the squares of all the integers between 0 and 99. You should code this as efficiently as possible, to minimise the number of processor cycles required to carry out the computation.

You are free to decide on the instruction set and functionality (except that it must not implement multiply or divide operations). You *could* consider the most efficient method of implementing the showcase program, before deciding on the required functionality of your processor. This is, of course, not the normal way of doing things, as processors are normally general-purpose!

You are strongly encouraged to automate the translation of MIPS assembly into machine code to run on your machine.

Similarly, you may wish to automate the production of testing scripts to fully test your system.

Report

Your report should be formatted as a technical report (A4, single-column, minimum font size 10), and include:

- Design: up to 1 page to describe your implemented "processor", and up to 1 page to illustrate the registers, memory layout and instruction set.
- Functionality: up to 2 pages describing how the system and each module has been implemented.
- Basic Testing: up to 2 pages to describe your basic testing plan and your key results. You can include detailed evidence of testing in an appendix.
- Showcase Testing: up to 1 page to describe your implementation of the showcase program, initially described in C or pseudo-code. You should include a count of how many instructions were required to run it. You can include detailed evidence of testing in an appendix.
- Conclusions: up to half a page to summarise the work, and describe any future work that needs to be done.

You must submit your source-code and a report. Without the report, it will be impossible for us to interpret how you have implemented your system. Any submissions which do not include both the source code and a report will receive a mark of zero.

Hand-in and Marking

Hand-in is electronic-only via handin.ecs.soton.ac.uk, deadline is 09.00 on Monday 12 April 2021. You will need to submit a ZIP of your source code, along with a PDF of your report.

The mark breakdown is:

A robust and effective design/implementation:	40%
Basic test program suite and results:	25%
Showcase test program and results:	15%
Quality of technical report:	20%

Academic Integrity

You may discuss this coursework with your colleagues, but the code and report you hand in **must be entirely your own**. If you use any library code for any functions, this must be clearly declared. Both the submitted software and the reports will be checked for similarity.