ELEC2204 Computer Engineering

Coursework: Computer Simulation

Introduction

Simulators are useful when designing computer systems. They can implement the functionality of various parts of a computer in software, and may be used to make a computer run software developed and compiled for another machine. It may also allow all internal signals to be visualised and recorded; while this 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 very simple 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 computer *in software.*
- Follow the Von Neumann architecture, fetching and executing from memory. It is not expected that you will have time to implement caches or virtual memory.
- Follow a straightforward *fetch/decode/execute/write-back/memory access* cycle.
- Implement a limited instruction set and register topology, which you may define yourself (you may choose to implement a subset of MIPS instructions). You **must not** implement a multiply 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 memory accesses.

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 two specific 'showcase' programs:

1. To calculate (and print out) the squares of all the integers between 0 and 99
2. To calculate (and print out) all the prime numbers between 1 and 1000

You *could* consider the most efficient method of implementing these programs, 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!

Your processor should run machine code, which should be read-in from a separate input file, but you are free to decide on the instruction set and functionality. You may wish to automate the translation of assembly into machine code. The most able students may even implement a basic C compiler!

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

Report

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

- Design: up to 1 page for a diagram of your implemented "processor", and up to 1 page to describe its 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 programs, and the key results. 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 20 April 2020. 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:	20%
Showcase test programs and results:	20%
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. Both the submitted software and the reports will be checked for similarity.