CSCI 2321 (Principles of Computer Design), Spring 2014
Review for Final Exam

1 Format of the exam

Scheduling of the exam is being decided. It is proposed to be a take-home. It will be about twice the length of the midterm (or a bit less) and so should take a little less than two hours, but since it's a take-home you may take as long as you like. Like the quizzes and midterm, it's open book / open notes, meaning that you can use your textbook, anything provided via the course Web site or as in-class handouts, and your own graded work. You may not use other books or a calculator or computer, *except* to view the above material.

The exam will be comprehensive but will focus on material since the midterm (about half of the questions/points will likely be about material from the second half of the course). Most questions will be similar in form to those in the quizzes, minute essays, and homework assignments, and somewhere in between with regard to difficulty (more difficult than the quiz questions, easier or at least shorter than some of the homework).

2 Topics to review

You are responsible for all material covered in class or in the assigned reading. The following is a summary of topics I think are most important, though it's not necessarily exhaustive. It would probably also be helpful to review sample solutions for the quizzes, assignments, and minute essays (the ones that have well-defined answers).

- *Review* Terminology from chapter 1.
- *Review* Defining and measuring performance; relationship among execution time, clock rate, cycle time, and cycles per instruction.
- *Review* Idea of "instruction set architecture" (as the interface between hardware and software).
- *Review* MIPS instructions described in chapter 2 — usage and binary (machine-language) representation.
- *Review* Compilers, assemblers, linkers, and loaders.
- *Review* MIPS conventions for procedure calls.
- *Review* Binary, decimal, and hexadecimal number systems; two's complement notation.
- Computer arithmetic (on integers): addition, subtraction, multiplication and division (basic ideas only, not details of algorithms).
- Floating-point representation and arithmetic (basic ideas only, not details).
- Combinational-logic blocks versus state elements.

- AND and OR gates and inverters, and how to use them to implement Boolean functions.
- Finite state machines, at the level discussed in appendix B.
- Design of an ALU.
- Design of a datapath for a single-cycle implementation of our selected subset of MIPS instructions: what elements are needed, how to connect them, what control signals are needed.
- Generating control signals for our single-cycle implementation — what elements we need (two combinational-logic blocks), their inputs and outputs, how outputs depend on inputs (expressed via truth tables and/or Boolean functions).
- How the completed single-cycle implementation (datapath and control) executes example instructions.
- Why a single-cycle implementation isn't really practical.
- A little about pipelining — how it can help, the basic idea (assembly-line analogy), what makes it tricky (the "hazards"), basic idea of how it's implemented (divide single-cycle design into stages and add "pipeline registers").