CSCI 2321 (Principles of Computer Design), Spring 2015

Review for Final Exam

1 Format of the exam

The final will be a take-home exam. It will be about twice the length of the midterm so in principle should take no more than 2.5 hours, but you may spend as long on it as you care to as long as you turn it in by the deadline (5pm Thursday May 7). 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 distributed in class or via Google drive (such as sample solutions, but *for this year only*), and your own graded and ungraded work. You may not use other books or a calculator or computer, *except* to view the above material. My preference is that you do the exam in conditions as similar as possible to those for an in-class exam — one (possibly long) session in a quiet place with few distractions — but that's not a strict requirement if it's not feasible.

The exam will be comprehensive but will focus on material since the midterm (so, there will be proportionally more questions on material on which you have not yet been tested). Most questions will be similar in form to those in the homework assignments, the quizzes, and the midterm. Some questions will likely resemble either shorter versions of homework problems; others will be short-answer or multiple-choice.

2 Topics to review

You are responsible for all material covered in class or in the assigned reading, with the exception of the material in chapters 5 and 6. 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 any minute essays 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.
- Binary, decimal, and hexadecimal number systems; two's complement notation.
- Computer arithmetic (on integers): addition and subtraction.
- Floating-point representation and some of its limitations.

- 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").
- A little about exceptions (interrupts) and what needs to be added to the single-cycle implementation to support them; a little about how they interact with pipelining.