Fall 2012 Syllabus for:
CSCI 2500, Computer Organization

Prof. Christopher D. Carothers
Department of Computer Science
Rensselaer Polytechnic Institute
110 8th Street Troy, New York 12180
e-mail: chrisc@cs.rpi.edu

course website: www.rpi.edu/~carotc/COURSES/CSCI-2500/FALL-2012/index.html
phone: (518) 276-2930 fax: (518) 276-4108 & (518) 276-2529
Office Hours: MRC 309a, Tuesday and Friday, 2 to 4 p.m. and by appointment.
Class Time and Location: DCC 318, Tuesdays and Fridays, Noon to 1:50 p.m.

August 28, 2012

1 Course Description and Textbook Information

Introduction to computer organization, assembler language, and operating systems with a heavy emphasis on systems and low-level programming. Topics include, but are not exclusively limited to:

- Organization/design of processors, memory and I/O.
- Numeric representation including binary integer and floating point number systems.
- Digital logic including Boolean algebra, gates, digital logic circuits, and memory.
- Assembly language including instruction formats, addressing modes, instruction types, flow of control, the assembly process, macros, linking, loading.
- Advanced architectures including RISC architectures and parallel architectures.
- Operating systems virtual memory, processes and interprocess communication.

Prerequisite CSCI 1200 (CS II).

1.1 Required Textbooks

- Computer Organization & Design: The Hardware/Software Interface, Revised, 4th Edition (2012), By Patterson and Hennessy. Make sure it comes with the companion CD. To make sure you have the correct edition, please see the photo of the textbook cover as shown in Figure 1.

- Soul of a New Machine, Tracy Kidder. Available on books.google.com, or barnesandnoble.com. To be used for the bonus assignment.

1.2 Optional Textbooks

2 Performance Expectations

As a professor, student adviser and course instructor, I get asked, “Is Comp Org Hard?”. The answer is that depends. For many of you, Computer Organization is your first really hands on programming oriented course. So, I have found over the years that students who truly enjoy the “Art” and “Science” of programming, perform very well in the class. However, students who struggled with the programming assignments in CS I or CS II, unfortunately have done even worse in this course. The reason is that the details for low-level assembly language programming grow by at least 10x. Additionally, if you are not fluent in the basic constructs of at least C++ programming such as classes, for-loops, while-loops, if-then-else statements coupled with the ability to take a relatively high-level problem description (i.e., write a program that reads two variable dimension NxN matrices from a file, multiplies them together and outputs the result in a matrix format) and quickly break it down mentally and construct a structured program to accomplish this task, you will find this course very difficult.

Another compounding factor is time management. If you are able to complete the Computer Organizations programming assignments relatively quickly, say 10 hours, as opposed to getting stuck on a bug for nearly 40 hours, successful Computer Organization students tend to stay rested, have more time for other classes as well as prepare for the bi-weekly Computer Organization quizzes. Unsuccessful students have the feeling they are overwhelmed in an ever growing avalanche of homework and quizzes.
If you start to get that overwhelmed feeling at all, ask for help and ask for early – do not wait until a few days before the assignment is due. Additionally, if you feel you are not as strong of a programmer and found CS I and CS II challenging, then I suggest you start all the programming oriented assignments the day they are assigned. If any part of the assignment is unclear, please see Dr. Carothers as soon as possible or e-mail him.

3 Graduate Teaching Assistants

We have two Graduate TAs assigned to our class. All office hours are typically held in the Amos Eaton, room 217, unless otherwise specified by the TA.

1. Deyang Gu (gud@rpi.edu): Office hours: Wed. and Friday 10 am. – 12 p.m.

2. Andrew Zonenberg (zonena@rpi.edu): Office hours: Mon. and Thurs. 4–6 p.m.

About TAs in this course: Much of the work of our TAs is the behind the scenes grading. As you can imagine, with over 100 students, 7 homeworks, 7 quizzes, 1 bonus assignment and 1 group project, that makes for nearly 1600 potential grades for this one class. So, their grading load is substantial and this is on top of a full load of research and graduate coursework! So, the TAs may not get time to fully delve into the subtle nuisances of different programming solutions etc. Additionally, an overwhelming majority of our TAs come from other institutions. The undergraduate courses may not precisely line-up with ours and the version of a Computer Organization course they took could be substantially different depending on the school. Consequently, our TAs are not necessarily masters of either x86 nor MIPS assembly language. If you have complex debug questions regarding programming assignments, please see or e-mail Dr. Carothers.

4 Schedule of Topics

- Introduction to Unix and C: Assignment 1.
- History, Performance and Why Parallelism?: P&H/Chapter 1 and class hand-out, Assignment 2.
- Assembly Language Programming MIPS and x86: P&H/Chapter 2 and hand-outs, Assignments 3 and 4.
- Digital Logic: P&H/Appendix B, Assignment 5.
- Building a Processor: P&H/Chapter 4, hand-out, start of group project.
- Pipelining & Multiprocessors: P&H/Chapters 4 and 7 plus lecture notes, Assignment 7.
- Memory Hierarchy: P&H, Chapter 5, finish-up group project.

5 Schedule of Homeworks and Quizzes and NO CLASS days

- Assignment 1 due on Tuesday, September 4th → Quiz 1 on Friday, September 7th.
- NO CLASS, Tuesday, September 18th, Rosh Hashanah.
- Assignment 2 due on Wednesday, September 19th → Quiz 2 on Friday, September 21th.
• Assignment 3 due on Tuesday, October 2nd → Quiz 3 on Friday, October 5th.
• NO CLASS, Tuesday October 9th, follows Monday class schedule.
• Assignment 4 due on Tuesday, October 16th → Quiz 4 on Friday, October 19th.
• Assignment 5 due on Tuesday, October 30th → Quiz 5 on Friday, November 2nd.
• Assignment 6 due on Tuesday, November 12th → Quiz 6 on Friday, November 15th.
• Friday, November 23rd, NO Class, part of Thanksgiving Holiday.
• Group Project due date, Friday November 30th.
• Assignment 7 due on Tuesday, December 4th → Quiz 7 on Friday, December 7th.

6 Grading and Other Class Policies

• 42%: 7 homeworks, 6 pts each – due every other Tuesday.
• 49%: 7 quizzes, 7 pts each, given in class every other Friday.
• 9%: 1 project.
• Bonus – you can replace your lowest homework grade.

Attendance Policy: Attendance at lectures is not required, but be aware that I may include material not necessarily covered in the text or on the web page. You are responsible for all announcements made in lecture (e.g., any change in due dates).

Late Assignments Policy: Late assignments will not be graded. You will get a zero for that assignment, except under extenuating circumstances, such as illness, family death etc. If you are ill, please be prepared to provide a note from the health center or your own family physician.

Grade Modifiers Policy: Grade modifiers will be used in this class. Nominally, for example, you expect to earn a B- if your score is greater than 79.5 and less than 83.0, B if your score is greater than 83 and less than 86, B+ if your score is greater than 86 and less than 89.5. The same modifier points occur for the A, C and D ranges except that there is no A+ nor is a D- allowed under the RPI Grade Modifier Policy.

Assignment Grading Criteria: Programming assignments are graded as follows: 15% for proper comments (e.g., each function should indicate what it does) and 85% for a correct working implementation. We typically divide the correctness points over key functions working. For example, reading – worth 10 points, writing – worth 10 points as file correctly, and then doing the calculation correctly – worth 65 points. Note that programs that either don’t compile or generate a “core dump” typically get no more than 20 points of the 85. Thus, your max score for a “properly commented” program that fails in some fundamental way is only 35 points even if you spent 100 hours of time on it. Non-programming assignments/homeworks are graded on a per-problems basis. Typically 5 problems will be given and each is worth 20 points.

7 Academic Integrity

While I strongly encourage you to form study groups and work together in learning this material, the course project, homeworks and programming assignments are to be done individually unless otherwise noted by the assignment/project specification. What this means is that you should do whatever is necessary to ensure your work remains your work. For example, in doing programming assignments
you might want to prepend variable names with your initials. If during in the grading process, it is determined that students shared or duplicated work, those students will automatically take a zero for the offense plus a 5 point total average deduction. For a second offense, the student or students involved will fail this course and a report will be sent to the Dean of Students office which could result in additional disciplinary action.

8 Learning Outcomes

By the end of this course, you will be able to:

1. Apply the concepts of the C programming language to the construction of moderately complex software implementation problems.

2. Apply the concepts of assembly language to correct and efficient translation of a given C programming language into the course required assembly language(s).

3. Apply the concepts of integer and floating point formats to convert from the base-10 integer or scientific format into the correct machine readable binary format.

4. Apply the concepts of Boolean Algebra to simplify given Boolean equations

5. Apply the concepts of K-Maps to the problem of Boolean expression simplification.

6. Apply the concepts of Performance to the analysis of computer performance problems.

7. Apply the concepts of a multicycle datapath and control by showing in written form the processing steps that different classes of instructions require as they move through the datapath and control hardware structures.

8. Apply the concepts of a pipelined datapath and control by showing in written form the processing steps that different classes of instructions require as they move through the datapath and control hardware structures.

9. Apply the concepts of caching and memory hierarchy to solve a problem which requires you to design the “best” cache system given particular design constraints.

10. Apply the concepts of parallel programming to the construction/implementation of a correct and efficiently executing multithreaded program.