15-213: Introduction to Computer Systems

Majd F. Sakr and Khaled A. Harras
Computer Science
Carnegie Mellon University

Fall 2009

1 Organization

Instructors:

Majd F. Sakr
msakr@qatar.cmu.edu
M1007, 454-8625
Tue, 3-5pm

Khaled A. Harras
kharras@qatar.cmu.edu
M1020, 454-8617
Thu, 2-4pm

TA:

Md. Shahriar Haque
mhaque1@qatar.cmu.edu
Academic Resource Center (ARC)
Mon ?-?pm, Wed ?-?pm
Please see the class Web page for up-to-date office hours.

Lecture:

Mon and Wed, 8:30–9:50am, M1031

Recitations:

Thu 10:30–11:20am, M1031, Shahriar Haque

Class Web Page:

`http://www.qatar.cmu.edu/~msakr/15213-f09/`

Class Message Board:

`http://autolab.qatar.cmu.edu/15213-f09/`

Note: This is the only message board your instructors will be monitoring. We will not be using the Andrew or Blackboard message boards for this class.

2 Objectives & Learning Outcomes

Our aim in CS 213 is to help you become a better programmer by teaching you the basic concepts underlying all computer systems. We want you to learn what really happens when your programs run, so that when things go wrong (as they always do) you will have the intellectual tools to solve the problem.

Why do you need to understand computer systems if you do all of your programming in high level languages? In most of computer science, we're pushed to make abstractions and stay within their frameworks. But, any abstraction ignores effects that can become critical. As an analogy, Newtonian mechanics ignores relativistic effects. The Newtonian abstraction is completely appropriate for bodies moving at less than 0.1c, but higher speeds require working at a greater level of detail.

Oversimplifying matters somewhat, our 21x sequence works as follows: 211 is based on a simplified model of program execution. 212 builds further layers of abstraction. 213 introduces greater detail about system behavior and operation. This greater detail is needed for optimizing program performance, for working within the finite memory and word size constraints of computers, and for systems-level programming.

The following "realities" are some of the major areas where the abstractions we teach in 211/212 break down:

1. *Int's are not integers, Float's are not reals.* Our finite representations of numbers have significant limitations, and because of these limitations we sometimes have to think in terms of bit-level representations.

2. *You've got to know assembly language.* Even if you never write programs in assembly, The behavior of a program cannot be understood sometimes purely based on the abstraction of a high-level language. Further, understanding the effects of bugs requires familiarity with the machine-level model.

3. *Memory matters.* Computer memory is not unbounded. It must be allocated and managed. Memory referencing errors are especially pernicious. An erroneous updating of one object can cause a change in some logically unrelated object. Also, the combination of caching and virtual memory provides the functionality of a uniform unbounded address space, but not the performance.

4. *There is more to performance than asymptotic complexity.* Constant factors also matter. There are systematic ways to evaluate and improve program performance

5. *Computers do more than execute instructions.* They also need to get data in and out and they interact with other systems over networks.

By the end of the course you will understand these "realities" in some detail. You will become a more effective programmer; you will be able to find and eliminate bugs efficiently and to tune program performance. As a result, you will be prepared to take any of the upper level systems classes at Carnegie Mellon (both CS and ECE) such as Compilers, Operating Systems, Networks, Computer Architecture and Embedded Systems. Even more important, you will have learned skills and knowledge that will help you throughout your career.

3 Textbook

The primary textbook for the course is

> Randal E. Bryant and David R. O'Hallaron, *Computer Systems: A Programmer's Perspective*, Prentice Hall, 2003.

In addition, we require you to have the following reference book on the C programming language:

> Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language, Second Edition*, Prentice Hall, 1988.

This the classic *K & R* book, the standard against which all reference manuals are compared. It is an essential part of every computer scientist's library.

4 Course Organization

Your participation in the course will involve five forms of activity:

1. Attending the lectures.
2. Preparing for and participating in the recitations.
3. Laboratory assignments.
4. Reading the text.
5. Exams

Attendance will be taken at the lectures and recitation, it will be worth 5% of your grade. You will be considered responsible for all material presented at the lectures and recitations.

Lectures will cover higher-level concepts. Recitations will be more applied, covering important "how-to's", especially in using tools that will help you do the labs. In addition, the recitations will help clarify lecture topics and describe exam coverage.

The textbook contains both *practice problems* within the chapter text and *homework problems* at the end of each chapter. The intention is that you work on the practice problems right as you are reading the book. The answers to these problems are at the end of each chapter. Our experience has been that trying out the concepts on simple examples helps make the ideas more concrete. In addition, the schedule (at the end of this document and on the class web page) shows specific homework problems with each lecture topic. The intention is that you try these out and discuss them in the next recitation. You will find that you will get much more out of recitation if you have done some advance preparation.

The only graded assignments in this class will be a set of seven labs. Some of these are fairly short, requiring just one week, while others are more ambitious, requiring several weeks.