15-513: Introduction to Computer Systems

Brian Railing

Sol Boucher
Carnegie Mellon University
Summer 2018

1 Organization

Class Web page: `http://www.cs.cmu.edu/afs/cs/academic/class/15513-m18/www/`

Electronic copies of class handouts and lecture slides as well as links to lecture videos can be found on the class Web page. They will be posted after each class.

Piazza address: `http://www.piazza.com`

Please post a private message on Piazza whenever you have questions about the course. Don't send mail to individual staff members except to schedule one-on-one meetings. We will not be using Blackboard or any other message board.

Instructors:

Brian Railing
bpr@cs.cmu.edu
GHC 6005, 412-268-3143

Lecture: Tue, Wed, Thu, Fri, 12:00–1:20pm, GHC 4215

Recitations: Recitation material is integrated into the summer lecture schedule.

Office Hours: Please see the class web page for instructor and TA office hours.

2 Objectives

Our aim in 15-213/18-213/15-513 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.

The following "realities" are some of the major areas where the abstractions you've learned in previous classes 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. As a result, you will be prepared to take any of the upper-level systems classes at Carnegie Mellon (both CS and ECE). Even more important, you will have learned skills and knowledge that will help you throughout your career.

In detail, we set forth the following learning objectives:

1. Explain common bit-level representations of numeric values (unsigned, twos complement, floating point) and the consequent mathematical properties of arithmetic and bit-level operations on them.

2. Recognize the relation between programs expressed in C and in assembly code, including the implementation of expressions, control, procedures, and data structures.

3. Demonstrate ability to understand basic intention of a program through its binary representation and apply these skills to debugging programs.

4. Investigate the programmers interaction with the underlying system through the different APIs and abstractions, including system support for process and thread control, virtual memory, and networking.

5. Analyze the consequences of imperfect system usage, such as poor memory and CPU performance, crashes, and security vulnerabilities.

6. Apply tools, both standard and self-developed, that will aid program development, including compilers, code analyzers, debuggers, consistency checkers, and profilers.

7. Apply these analytic and tool-use abilities to create reliable and efficient programs exercising the different components of a modern computing system.

3 Textbook

The primary textbook for the course is

> Randal E. Bryant and David R. O'Hallaron, *Computer Systems: A Programmer's Perspective, Third Edition (CS:APP3e)*, Pearson, 2016.

Please make sure you have the Third Edition, which is significantly different from the Second Edition published in 2011. 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 is the classic *K & R* book, the standard against which all reference manuals are compared. This book should be in the library of anyone who programs in C.

4 Course Organization

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

1. 15-213 and 18-213: Attending the lectures and participating in class
2. Doing laboratory assignments.
3. Reading the text.
4. Taking exams.

Students in 15–513 are not permitted to attend the 15–213 lectures. Online materials are provided for students' review purposes.

Classroom time will cover both higher-level concepts as well as applying this material in practice, particularly toward the labs.

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 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. Try out the practice problems associated with the readings for each class and ask questions about them at 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.

5 Getting Help

We will use the class website (`http://www.cs.cmu.edu/afs/cs/academic/class/15513-m18/www/`) as the central repository for all information about the class.

For technical questions (lectures, exams, assignments), post a question on Piazza. By default, any question you post will be private to you and the instructors. We will put posts on Piazza and in the FAQ web page answering some common questions. Be sure to check these before contacting an instructor.

The lab assignments are offered through a hosted autograding service, developed by Dave O'Hallaron and a group of CMU undergrads, called *Autolab*. See the Autolab web page at `http://autolabproject.com` for more information.

If you want to talk to a staff member in person, the posted office hours are the best opportunity, as the they represent times when we guarantee that we will be in the location identified. If a meeting is needed outside of the office hours, please use email to arrange a time.

6 Policies

Working Alone on Assignments

You will work on all assignments by yourself.

Version Control

We will using GitHub Education for you to work on labs, with pre-populated with directories for labs 1, 4–7. The GIT repositories are private and will be deleted after the end of the semester. You will have a chance to download their contents before they will be deleted. We will explain the proper usage of the server and help with setting up the server in office hours and the recitations. Follow the following procedure:

- Add all the *source files* (.c, .h, Makefile, input files) in your lab assignment upon downloading them from Autolab and commit the initial version.
- Commit early and often. Make it a habit to commit at least every hour you work actively on the assignment, and commit in small increments. Commit at the end of your work day.
- Make sure you commit your final version right before/after you submit via Autolab.

It is good software engineering practice to use version control, and learning it before starting Lab 1 is a a good idea. We will be watching commit statistics on the server and may be reaching out to students who disregard our version control policy.

Handing in Assignments

All assignments are due at 11:59pm (one minute before midnight) on the specified due date Eastern Time. All handins are electronic using the Autolab system. You may handin in as often you like, with your most

recent handin counting for credit.

Handing in Late Assignments

The penalty for late assignments is 15% per day. Each student will receive a budget of five *grace days* for the course. These grace days are provided to allow you to cope with most emergencies that prevent completing a lab on time, including computer problems, a cold, getting stuck at the airport, etc. Here is how grace days work:

- Each assignment has a maximum number of grace days that can be applied, ranging from 0 to 2. The grace day limits are indicated on the Assignments web page and in the assignment writeups.
- Grace days are applied automatically until you run out.
- If your last handin is one day late, and you have at least one remaining grace day, then you will receive full credit for the lab and automatically spend one grace day. For example, if an assignment is due at 11:59pm on Thursday and your last handin is noon on Friday, then you will receive full credit and spend one grace day.
- Once you have spent your grace days, or exhausted the limit for the assignment in question, then you will receive a penalty of 15% for each subsequent late day. For example, suppose you have only one grace day left. If an assignment is due at 11:59pm on Thursday and your last handin is noon on Saturday, then you will spend your one remaining grace day and be penalized 15%. If your last handin is noon on Sunday, then you will spend one grace day and be penalized 30%.
- Handins will not be accepted after the *end date* of the lab, which is typically three days after the due date.

Grace days are a tool to allow you to manage your time in the face of personal issues and to help smooth out burstiness in assignment due dates across classes. They are for when you are sick, when a short-term emergency situation arises, when you have too many deadlines all at once, etc. Except for serious persistent personal issues (see below), you should not anticipate additional deadline leniency. **We strongly recommend that you conserve your grace days, saving them for the more difficult assignments at the end of the term.**

Dealing with Serious Persistent Personal Issues

We hope that everyone in 15-213/18-213/15-513 will remain happy and healthy. But, if you have a serious persistent personal issue, such as being hospitalized for an extended period or needing to leave the country for a family matter, please talk to your academic advisor as soon as possible. Such issues consistently affect one's ability to succeed in all classes, rather than just 15-213/18-213/15-513, and the academic advisors are equipped to coordinate plans for dealing with them. We will cooperate with such plans, but we cannot construct them independently of the academic advisors. Please contact your course instructor and academic advisor if you are unable to keep up with the course due to a serious personal issue.

Requesting a Regrade for an Assignment or an Exam

After each exam and lab assignment is graded, your score will be posted on the Autolab gradebook. We will make the utmost effort to be fair and consistent in our grading. But, we are human. If you believe that you did not receive appropriate credit for an assignment or an exam, you may request a regrade as follows:

- **Exam regrade request:** All exam regrades must be completed using the exam server.
- **Lab regrade request:** Post a regrade request as a private message on Piazza. Provide a detailed explanation of why you believe your grade did not conform to the posted grading standard, and indicate your and your grader's andrew ID.
- **Verbal and email requests will NOT be accepted.**
- All regrade requests must be received within **seven days** of the grades becoming available.

Your request will be processed off-line, and we will respond to your request as quickly as possible (typically within a week). This regrade policy is designed to correct legitimate mistakes in grading, while discouraging frivolous regrade requests (for the sake of being fair and consistent across the entire class).

Final Grade Assignment

Each student will receive a numeric score for the course, based on a weighted average of the following:

- **Assignments (50%):** There are a total of seven assignments (labs), which will count a combined total of 50% of your score. Assignments have different weightings, based on our perception of the relative effort required. See the Assignments web page for the assignment weightings.
- **Exams (45%):** There will a midterm exam counting 10% and a final exam counting 35%.
- **Prepatory (5%):** The AIV and Lab0 assignments to ensure that you are prepared for this course.

The grading cutoff points are: 90 (A), 80 (B), 70 (C), 60 (D). We will selectively consider raising individual grades for students just below the cutoffs based on factors such as attendance, class participation, improvement throughout the course, final exam performance, and special circumstances.

Cheating (and Plagiarism)

Please read this carefully, especially if this is your first semester at CMU!

Each exam and lab assignment must be the sole work of the student turning it in. Assignments will be closely monitored by automatic cheat checkers, including comparing turned-in code to the work of students from the same and previous semesters and solutions found on the Web. Students may asked to explain any suspicious similarities. These cheat checkers are very effective, having been refined over years of research, and they are not fooled by attempts to mask copying of code. Please don't try your luck.

Anyone caught cheating will receive an academic integrity violation (AIV) by the university. This will become part of the student's permanent record. The default penalty for cheating is to be removed from the course with a failing grade. In some circumstances, we may consider a lesser penalty, but it will always be worse than having not turned in the assignment at all.

No collaboration of any form is allowed on exams. Students may not discuss any aspect of any exam question with someone who has not yet taken the exam.

The following are guidelines on what non-exam collaboration is authorized and what is not:

What is Cheating?

Unauthorized use of information. These provisions hold for you this semester.

- *Copying*: Obtaining code or other solution information, either by copying, retyping, or looking at a file or document from this semester, a previous semester, or an external source. You are only allowed to use code that we provide you or that is provided on the CS:APP web site.
- *Searching*: Searching the Web for solutions or for any advice on the lab.
- *Reusing:* Code reuse is complex. You are allowed to reuse general knowledge pieces from prior courses. For example, you can reference code for a linked list or to process commandline arguments. You may not submit prior work on the labs, even if they are your own, if you received credit for that course either at CMU or at another institution. You also may not reuse code if you dropped or withdrew or failed in a prior semester, in those cases we expect you to start afresh and improve your outcome in the course.
- *Looking at other's code:* Although mentioned above, it bears repeating. Looking at someone else's code (or exam problem) is cheating. This includes one person looking at code and describing it to another. There is no notion of looking "too much," since no looking is allowed at all.

Unauthorized Supplying of Information. Thes provisions hold for this semester and into the future.

- *Sharing*: Supplying a copy of a file or document to a current and future student in 213/513.
- *Providing access*: Keeping solution files in unprotected directories or in unprotected code repositories, now and in the future. Be sure to store your work in protected directories, and log off when you leave an open cluster, to prevent others from copying your work. If you make use of a code repository, such as Github, make sure your work is kept private, even after you have left CMU.
- *Describing*: Verbally describing code to a current or future student.
- *Coaching*: Providing detailed directions to a current or future student.

What is NOT Cheating?

- Clarifying ambiguities or vague points in class handouts or textbooks.
- Helping others use the computer systems, networks, compilers, debuggers, profilers, or other system facilities.

- Helping others with high-level design issues only. Algorithm implementations and other such details are not "high-level design issues." If you need code (or pseudo-code) to describe the issue, then it is not high-level.
- Helping others with high-level (not code-based) debugging.
- Getting help from the course instructors and teaching assistants.
- Using code from the CS:APP website or from the class web pages.

No Statute of Limitations. The above stated rules apply even after your have completed the course. You may not share code you have written for this course with future students. That means you cannot leave your code in unprotected repositories or post it on any web page. You may not provide coaching to future students. The university policies on academic integrity include the possibility of receiving an AIV even after a student has completed a course, potentially changing a grade retroactively and even revoking a degree. **Do not jeopardize your career at CMU or beyond by cheating or plagiarizing!**

Version Control and Cheating. In case you are accused of using un-allowed code in your hand-in (e.g., you are accused of having based your code off an implementation found on the web) we will use your commit log at the GIT server to establish the time line of your work. If we cannot establish reasonable progress of your work along a reasonable time line we will conclude that you indeed have cheated. It is in your interest to produce a convincing GIT commit log that shows you have done the work yourself, hence, commit early and often.

7 Mobile devices and other distractions

Research on learning shows that unexpected noises and movement automatically divert and capture people's attention, which means you are affecting everyone's learning experience if your cell phone, pager, laptop, etc. makes noise or is visually distracting during class. For this reason, we allow you to take notes on your laptop, but insist that you turn the sound off so that you do not disrupt other students' learning. If you are doing anything other than taking notes on your laptop, please sit in the back row so that other students are not distracted by your screen.

8 No recording of class meetings

Recordings of any 213 class, in part or whole, including any audio and/or video recordings, regardless of the media or format, and regardless of the intended or actual use, are not permitted without explicit prior written consent of all instructors. The class will be notified in advance should any such recording be approved. Students have no right to record classes under any University policy. If a student believes that he/she is disabled and needs to record or tape classroom activities, he/she should contact the Office of Equal Opportunity Services, Disability Resources to request an appropriate accommodation.

The penalty for violating this policy is an R in the course. If you are not comfortable with this, please drop the course now.

This policy is intended primarily to protect the privacy of the students. For example, no student should run the risk of potential employers finding a question, incorrect answer, or even look of confusion on the Web. The classroom is a learning environment, not an exhibition. Rather than attempt to control the uncontrollable or distinguish between neutral and detrimental uses, all recording is prohibited. Experience has shown that, excluding special cases such as use by students with disabilities or distance learners, undergraduate students do not improve their performance through the use of recordings.

9 Facilities: Intel Computer Systems Cluster

Intel Corp. has generously donated a cluster of Linux-based 64-bit multicore Nehalem servers, specifically for 15-213/18-213/15-513, that we will use for all labs and assignments. The class web page has details.

10 Class Schedule

Please see the schedule maintained on the class web page for information about lectures, reading assignments, suggested homework problems, lab start and end dates, and the lecturer for each class. The reading assignments are all from the CS:APP3e book.