15-213/18-213/15-513/18-613: Introduction to Computer Systems

15-213 Seth Copen Goldstein
18-213 Saugata Ghose
15-513 Brian Railing
18-613 Ian Lane

Carnegie Mellon University
Spring 2020

1 Organization

Class web page: www.cs.cmu.edu/~213
Piazza: piazza.com/cmu/spring2020/213513613
Autolab: https://autolab.andrew.cmu.edu/courses/15213-s20
Office Hours: Please see the class web page for instructor and TA office hours.
Lecture:

15-213 and 18-213 Lecture 1: Tue and Thu, 1:30–2:50pm, DH 2210

15-213 and 18-213 Lecture 2: Tue and Thu, 1:30–2:50pm, HOA 160

18-613 PGH students: Tue and Thu, 12:00–1:20pm, DH A302

18-613 SV students: Tue and Thu, 9:00–10:20pm, Building 23, Room 211

15-513 students: Videos of lectures will be available shortly after they are delivered. Links are included on the schedule web page.

Recitations: Please see the course web page for the list of recitations.

2 Objectives

Our aim in the course 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 lan-guages? 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 repre-sentations.
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, as activities you should be able to do after complet-ing the course:

1. Explain common bit-level representations of numeric values (unsigned, two’s complement, floating point) and the consequent mathematical properties of arithmetic and bitlevel operations on them.
2. Recognize the relation between programs expressed in C and in assembly code, including the imple-mentation 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 programmer’s interaction with the underlying system through the different APIs and abstractions, including system support for process and thread control, virtual memory, and network-ing.
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 compil-ers, 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.
8. Understand the sources of conflict that can arise when multiple threads of execution share resources, and demonstrate the ability to use synchronization constructs to mediate those conflicts.

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. You can get either hardcopy or electronic editions. Electronic access, starting at $33.99, is available through Vital Source. Copies are also on reserve in the Sorrells Engineering Library. Warning: Don't buy the paperback version of the book! It's not the same as the hardcover/electronic version. The practice and homework problems were rewritten and are a total mess.

In addition, we expect you to use reference material about the C programming language. Our suggested reference is:

> 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. Copies are on reserve in the Sorrells Engineering Library.

4 Course Organization

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

1. 15-213, 18-213, and 18-613: Attending the lectures and recitations
2. 15-513: Viewing the videotaped lectures and recitations
3. Doing laboratory assignments.
4. Reading the text.
5. Taking exams.

Attendance will not be taken at the lectures or recitation sections. 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 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 eight 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 (www.cs.cmu.edu/ 213) as the central repository for all information about the class.

For technical (lectures, exams, assignments) or logistics (accounts) questions, 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.

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

6.1 Working Alone on Assignments

You will work on all assignments by yourself.

6.2 Version Control

We will be using GitHub Education for you to work on labs, with pre-populated directories for labs 1, and 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 via Piazza posts, in office hours and during the recitations. In general, you should work as follows:

Add all of the provided source files in your lab assignment upon downloading them from Autolab and commit the initial version. Source files include any code (extensions '.c', '.h', '.pl', '.py', and '.sh'), as well as any Makefile and any program input file. It does not include any compiled libraries or reference programs.

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 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.

6.3 Handing in Assignments

All assignments are due at 11:00pm 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.

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 11:00pm ET / 8:00pm PM 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. Grace days and penalties are computed automatically by Autolab, with grace days being used up before penalties are applied. You cannot choose how to distribute your grace days among your assignments.

6.4 Dealing with Serious Persistent Personal Issues

We hope that everyone in the course 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 the course, 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 if you are unable to keep up with the course due to a serious personal issue.

6.5 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.

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).

6.6 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 eight 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 (50%): There will a midterm exam counting 20% and a final exam counting 30%.

The grading cutoff points are: 90 (A), 80 (B), 70 (C), 60 (D).

In-class quizzes (except 15-513): In the middle of many lectures, we will pause a few minutes to have students answer a 2–3 multiple-choice question quiz on the material covered in class. This will help the instructor gauge which topics need further discussion. Quiz questions will be answered by each student using Canvas. Student grades on these quizzes will be used solely as "bonus points," as discussed next.

Bonus points: 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. In particular, except for 15-513 students, the in-class quizzes provide a record of lecture attendance and attention, and hence are a good means for a student just below a cutoff to have his/her grade raised.

6.7 Academic Integrity

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

This course, as one of a set of related systems courses in CS, ECE, and INI have adopted a uniform policy on Academic Integrity Violations (AIVs). It is available at:

www.cs.cmu.edu/ 213/academicintegrity.html.

It provides very specific guidelines on what forms of collaborations are permitted, and what forms are not. Continue referring to it during the semester as you encounter specific choices you must make in doing your assignments. The following text is based on this web document, but it also includes more specific information about the possible penalties when an infraction occurs.

Our policy is based on the following beliefs, gained through many years of experience:

> Understanding the operation and implementation of computer systems is best learned by hands-on activities: writing, debugging, measuring, and exercising programs that expose the relevant system principles.
>
> Developing programs from scratch, or with limited starting code, requires using design principles and logical thinking that are much deeper than can be gained by copying and modifying an existing implementation. Making use of unauthorized sources diminishes the educational value of an assignment.
>
> Although teamwork and collaboration are important real-world skills, it is important to first gain the core competencies that enable individuals to serve as effective team members. These courses are designed to teach and assess these core competencies. Unauthorized collaboration diminishes the educational experience and the reliability of our assessments.

Based on these principles, we provide the following guidelines on what forms of resource use, resource sharing, and collaboration are permitted in these course.

Exams

Each exam must be the sole work of the student taking it. 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.

Labs and other Assignments: Information Sources

As a general principle, you may not obtain any information about an assignment from an unauthorized source. The following provide clarifications as to which sources are authorized, and which are not. These rules hold throughout the term.

Copying:

You may use material that we explicitly provide you for the assignment. No attribution is re-quired.

You may use other course material, including lectures, Piazza posts by the instructors, and material from the course website. For any such use involving code, you must provide clear attribution, indicating the source, and where the included material begins and ends.

You may use any material from the CS:APP book, any other course textbook, or the CS:APP web site. For any such use involving code, you must provide clear attribution, indicating the source, and where the included material begins and ends.

You may not obtain code or other solution information from an unauthorized external source, including web pages, code repositories, blog posts, etc.

Searching:

You may search for or refer to general information, including the use of systems, networks, compilers, debuggers, profilers, and program libraries.

You may not search the Web for solutions or for any advice on how to solve an assignment.

Reusing:

You may reuse elements of general knowledge from prior courses. For example, you may use existing code for a linked list or to process commandline arguments. For any such use involving code, you must provide clear attribution, indicating the source, and where the included material begins and ends.

If you have worked on one of the labs from CS:APP, either at CMU, at some other school, or on your own, you should arrange a meeting with one of the instructors at the beginning of the term to devise a policy on which parts of your solutions you may use. Reuse without explicit permission of an instructor, even if it's your own code, is forbidden.

Using other's code or documents:

You may not look at someone else's code (or other documents.) 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.

You may not make use of any information about the labs posted online, except for the authorized sources listed above.

Assistance:

You may get assistance on an assignment from the instructors, course staff, and university tutors. You may only get high-level, strategic advice from others, including current and former stu-dents, and people external to the university. Forbidden forms of advice include: anything more detailed than a brief verbal description or block diagram, any kind of code or pseudo-code, explicit directions on how to assemble allowed blocks of code, and code-level debugging assistance.

Labs and assignments: Sharing and Collaborating

As a general principle, you may not provide detailed help with an assignment to students this semester or in future semesters for any of the above-listed courses (unless you are serving as a teaching assistant or instructor for the course.) The following are clarifications about which forms of aid are authorized and which are not. Note that they apply to all of the above listed courses and from now and into the indefinite future.

Sharing:

You may not supply a copy of a file or document to an individual student or via a public channel, such as a blog post.

Providing Access:

You may not have any of your solution files in unprotected directories or in unprotected code repositories, either by putting files in an unprotected location or by allowing protections to lapse. 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.

Coaching, Assisting, and Collaborating:

You may not provide electronic, verbal, or written descriptions of code or other solution infor-mation.

You may clarify ambiguities or vague points in class handouts or textbooks.

You may help others use the computer systems, networks, compilers, debuggers, profilers, code libraries, and other system facilities.

You may discuss and provide general, strategic advice about an assignment. Providing anything more detailed than a brief description or a block diagram is not allowed. Providing any kind of code or pseudo-code is not allowed. Providing explicit directions on how to assemble allowed blocks of code is forbidden.

You may provide suggestions of potential bugs based on high-level symptoms. Code-based debugging assistance is forbidden.

Enforcement

We will aggressively employ cheat checkers and other means to detect unauthorized use of code from this term, previous terms, and available online. All infractions will lead to formal reporting of an AIV to the university and to the program. The standard penalty will be to be given a failing grade for the course. Lesser penalties may occur, depending on the circumstances, but as a general principle, the penalty will always be worse than if you had not turned in the assignment at all.

The above stated rules apply even after you 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. We can and will pursue AIVs against students after they have completed the course.

In risking an AIV, you jeopardize your participation in this course, your time at CMU, and your career beyond. The temptation to cheat can be very strong when deadlines approach, and you are unable to make satisfactory progress Doing so is far worse than failing to complete the assignment.

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 lecture or recitation, 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 this class, 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 assign-ments, suggested homework problems, lab start and end dates, and the lecturer for each class. The reading assignments are all from the CS:APP3e book.

11 Educational Research

For the undergraduate classes (15-213 and 18-213), Prof. Railing will be conducting research on improving student learning in computer systems courses through active learning. This research will explore two ap-proaches: first by utilizing several teaching techniques in the smaller, recitation setting, and second with the introduction of in-class quizzes. You will not be asked to do anything above and beyond the normal learning activities and assignments that are part of this course. You are free not to participate in this research, and your participation will have no influence on your grade for this course or your academic career at CMU. Participants will not receive any compensation.

The data collected as part of this research will include student grades. All analyses of data from participants coursework will be conducted after the course is over and final grades are submitted. The Eberly Center

may provide support on this research project regarding data analysis and interpretation. To minimize the risk of breach of confidentiality, the Eberly Center will never have access to data from this course containing your personal identifiers. All data will be analyzed in de-identified form and presented in the aggregate, without any personal identifiers. Please contact Prof. Railing (bpr@cs.cmu.edu), or Chad Hershock (hershock@cmu.edu) if you have questions or concerns about your participation.

12 Education Objectives for 18-213 (Relationship of Course to Program Outcomes)

The ECE department is accredited by ABET to ensure the quality of your education. ABET defines eleven Educational Objectives that are fulfilled by the sum total of all the courses you take. The following list describes which objectives are fulfilled by this course and in what manner they are fulfilled. The objectives are lettered from “a” through “k” in the standard ABET parlance. Those objectives not fulfilled by the course have been omitted from the following list.

(a) an ability to apply knowledge of mathematics, science, and engineering: Students will use mathe-matical and engineering concepts throughout the course to solve a series of programming assignments and on the exams.

(b) an ability to design and conduct experiments, as well as to analyze and interpret data: Students will need to design and conduct debugging sessions, including selecting test inputs and interpreting the results, in order to find and fix correctness and performance bugs in their programs.

(e) an ability to identify, formulate, and solve engineering problems: Students will need to iden-tify, formulate, and solve software engineering problems in a series of programming assignments of increasing open-endedness.

(f) an understanding of professional and ethical responsibility: Students will get practice in adher-ing to a code of professional ethics and responsibility dictated by the course, in the face of many opportunities to violate the code.

(g) an ability to communicate effectively: Students practice their communication skills during recitation sessions.

(k) an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice: Students will be taught computer programming techniques and skills via lectures, recita-tions, programming assigments, and exams. They will learn to use modern programming tools such as the gdb debugger, git version control, and x86 disassemblers.