COSC171 - Computer Systems

Spring 2023

Instructor: Professor Lillian Pentecost **Email:** lpentecost@amherst.edu

1 What is this course about?

'Computer Systems' encapsulates all the aspects of a computer that underlie the software that you run as a user. Together this semester, we will reveal and analyze *how* modern computers actually run a user's programs, as well as *why* computer systems are designed in particular ways.

This course will emphasize hands-on projects and peer discussion to develop our understanding. Our meetings together will usually be a mix of lecture, group activities, and time to work on projects and ask questions. So, along with developing hands-on programming and thinking skills, as well as conceptual understanding of the different layers and functions of computer systems, a learning goal in this course will be to give you practical experience and confidence in group collaboration on technical projects and verbal and written presentation of computer systems topics. As such, engagement with and reflection on group exercises, as well as verbal and written expressions, will sometimes be integrated into assessment of assignments and projects.

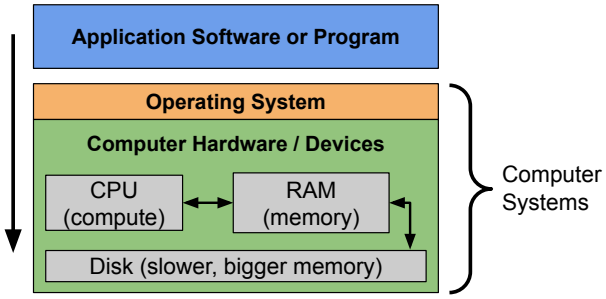

Figure 1: Below application software (e.g., programs you wrote in prior computer science classes), there are multiple *layers* that comprise a computing system. In this course, we take a peek at what these lower layers are, why they are important to understand, and how they support software. This figure represents our starting mental model of a computer system: the operating system sits between the software and the hardware and helps manage system components, and the underlying hardware is comprised of both compute units (where math and computation happens) and memory units of varying size and function (where information is stored).

In this course, we will often draw a distinction between a fundamental need in a computer system (e.g., a circuit must be designed to interpret a specific set of bits to mean 'add these two numbers') versus a specific *implementation* to satisfy such a need (i.e., the Intel x86 assembly language). Learning about specific implementations will help us appreciate and understand the workings of computer systems at deeper levels, as well as provide practical skills because these implementations and design choices are found in billions of computing devices around the world.

However, we can think critically together throughout this course about how and why particular implementations, systems, and programming languages were initially designed and have proliferated, as well as who has (and hasn't) contributed to the design and implementation of computer systems over time. For example, we will learn that certain hardware design choices can be traced back to the decisions of handfuls of Intel engineers and executives over the past several decades, and that those choices continue to directly impact hardware, operating systems, and software today, as well as programmers and end-users.

Adjusting Our Policies Together, Other Course Guidelines: What I outline in the remainder of this syllabus is just a starting point – we will all be relying on each other to discuss the material, develop understanding, and tackle projects. As such, all of you are welcome and encouraged to contribute additional classroom guidelines, give consistent feedback and input on what is working well and what is not (you will have an anonymous way to submit such feedback to me at any time). We will take a few minutes during class about once per month to check if there are any additional guidelines or policies to revisit.

Objectives: By the end of this course, you will deepen your understanding of the following questions:

- How does a computer run a program?
- How is a computer physically organized, and why?
- What is the role of the operating system?
- How are programs and their data stored?
- What is the memory hierarchy and how is it designed?
- What happens when a program needs more memory, or less?
- What systems choices and properties impact the performance of a piece of software?
- How does a single computer run multiple programs?
- How can a single program run on multiple computers?

Skills: Throughout this course, you will work to develop the following skills:

- Comfort with programming in C
- Comfort with basic programming in assembly (Intel x86)
- Comfort using the Linux command line and scripting
- Confidence using version control (git) for coding projects
- Understanding and interpretation of binary, hexadecimal, other number representations
- Confidence and productivity in collaborative group work
- Self-directed project organization and time management skills for coding projects
- Improved habits in debugging and commenting code
- Effective written and verbal communication related to computer systems concepts

Prerequisites: COSC 112 or equivalent programming / computer science familiarity.

2 How can you get the most out of this course?

Working through the conceptual puzzles and thorny debugging of computer systems will require constant **communication** and **questions**. This course is heavily **project-based and problem-solving-oriented** – we will consistently be learning a new idea or skill, and then immediately applying it in an activity or project; staying engaged and putting in consistent work on these projects will be key to your learning and success in this class. As such, you are highly encouraged to collaborate and conspire heavily with your peers as your primary source of developing understanding. As your instructor, I am also a handy resource for high-level and low-level questions (and comments, feedback) throughout the semester, as well as the course teaching assistants (TA) and options for individual tutoring. Below, I've listed some important resources and ways to get in touch:

Office Hours: These are times that I am available for open discussion and random questions, as well as topic review and hands-on project debugging. This is **dedicated time just for talking to you about computer systems**. I'll be available every Tuesday and Thursday from 4:30-6PM in my office (Science Center C217). If those times don't work for your schedule, or for detailed or personal questions and concerns, please schedule a time to meet one-on-one using this link. Please note that Fridays are my *research days*, and you should expect slower/delayed responses from me on Fridays and weekends.

- **Open Office Hours:** Tuesdays 4:30-6PM, Thursdays 4:30-6PM, Science Center C217
- **Scheduled Office Hours** by appointment using `https://calendly.com/professor-lillian`

Course Moodle: Regular announcements (including project due dates, slides from class, readings) will be distributed via Moodle. Check Moodle regularly for project instructions and up-to-date course information. You are also expected/encouraged to use the 'Questions' forum as a first-stop in asking for help from instructors and peers, as well as to monitor these fora and contribute answers and comments (which can also contribute towards your class engagement and participation expectations).

Email: If you have an urgent issue or if you have an extension request (see course policies in Section 4), try me via email (lpentecost@amherst.edu) **and include [COSC-171] in the subject line**.

Always-Available Anonymous Feedback Form: There is an always-open questionnaire for **anonymous** feedback, available via the Moodle page or via QR code at the end of each class. You can use this form at any time to provide anonymous feedback and ask questions on anything and everything course-related. I'll be checking responses roughly weekly and addressing feedback in class as needed.

TA Hours: Additionally, we will have regular evening help sessions available with a teaching assistant. Help sessions are great opportunities to bring questions and comments on class material, as well as a place to meet classmates and work on course projects and exercises. [Details to be announced on Moodle.]

Lecture & Discussion Meetings Of course, we will also meet for lecture and/or discussion on Tuesdays and Thursdays. You are expected to be present for **all class meetings**. These meetings will often be a mix of lecture (using slides and blackboard), working on exercises in small groups, and previewing project materials. You will often be encouraged to discuss with your peers and expected to present and participate in a community of learning in the classroom (see course policies in Section 4 for more details).

Hacker, Guess-And-Check, and Google-It Attitudes: Many times, the best way to understand project code and to test/debug your efforts will be to find something you don't understand, change it, and try to figure out why it does or doesn't work. Fiddling with things, breaking things, and looking things up on the web are very legitimate learning strategies for these concepts and projects. If you want some pointers and examples of strategic web searching of bugs or ways to explore the source code, that is a great topic for office hours and TA hours. I would also highly recommend bookmarking the 'missing semester' resources for handy tutorials and some time-saving skills that will be relevant to many tools we use together.

References and Readings: Readings will be assigned on Tuesdays and expected to be completed over the course of each week by the following Tuesday's class. You are not required or expected to purchase any books for this course. In addition to occasional news articles, some Wikipedia pages, and historical references, we will follow content in the following two texts:

- Dive Into Systems (fully openly available online, hard copies available on reserve in Science Library)
- Computer Systems: A Programmer's Perspective, 3rd Edition (important sections available on Moodle via eReserves, hard copies available on reserve in Science Library)

When completing reading, there will often be exercises available at the end of sections, or links to additional information or historical notes that may deepen your understanding or curiosity. These are all part of the 'reading', too! Taking your own notes, attempting exercises, and/or looking for related information online are all helpful strategies and can spur questions for you to post on Moodle or bring to office hours / TA hours. Some readings will be required to process and supplement lecture and discussion materials, while others are supplemental, but highly suggested, and may provide tips or context for projects. I'll try to mark the readings accordingly in the weekly Moodle schedule.

3 Other Logistics and Details

Grading:

- **Projects (40%):** We will work through eight projects over the course of the semester. In each project, your evaluation will depend on your strategy, your collaboration document, and your analysis/reflection on the project, in addition to the correctness and functionality of your implementation (i.e., the code you submit). Projects are your primary opportunities to wrestle with course content, demonstrate your understanding, and collaborate with your peers to analyze and implement key components and concepts in computer systems.
- **Class Engagement, Participation, Quizzes / Check-In Activities (30%):** Your engagement as a contributor and collaborator to the classroom community are key components of your learning and assessment in this course. This portion of your grade is primarily computed using your quiz / check-in activity completion and complemented by other forms of engagement and participation as part of our COSC-171 community (see course policies in Section 4 for more details).
- **Exams (30%):** There will be one midterm exam and a cumulative final exam. For both exams, you will be free to use any of your personal class notes and other assigned course materials. Nominally, the midterm is worth 10% of your overall grade and the final is worth 20%, though I will re-weight exam grades in your favor if you do significantly better on one exam or the other. (For example, if you do much better on the midterm than the final, the weight may be 14% midterm / 16% final, or if you do much better on the final than the midterm, the weight may be 5% midterm / 25% final.)

Important Dates (subject to modification):

Project 0 Due	Friday, February 10, 10PM
Project 1 Due	Friday, February 17, 10PM
Project 2 Due	Friday, February 24, 10PM
Project 3 Due	Wednesday, March 8, 10PM
Midterm Exam	in class Thursday, March 23
Project 4 Due	Wednesday, April 5, 10PM
Project 5 Due	Wednesday, April 12, 10PM
Project 6 Due	Wednesday, April 26, 10PM
Project 7 Due	Wednesday, May 3, 10PM
Final Exam	TBA

4 Course Policies:

Attendance: You are expected to be present at all class meetings. If you have a scheduled activity or an illness that will conflict with a class meeting, please let me know via email with as much advanced notice as possible and I can try to arrange to provide a class recording or other make-up materials.

Engagement: During class meeting times, you are expected to arrive ready to interact with your classmates and stay active in your learning. This could mean taking notes, taking moments to think and reflect, asking questions, debating with your peers, and more. To me, engagement in class also means muting/silencing any distractions (like text and email alerts) and being attentive and respectful to one another at all times.

Participation: In addition to being engaged and active during class meetings, there are other ways you can contribute to our classroom community. For example, this might look like responding to your peers' questions on the Moodle forum, posing your own questions for discussion, or frequent participation in open and/or scheduled office hours. If you have any questions about how to get engaged and stay engaged, feel free to reach out to me to discuss.

Quizzes and Check-Ins: Once a week or so (not necessarily announced ahead of time) we will use part of class to do an informal quiz or check-in activity. This may be a written response to a question posed during lecture, an exercise to be worked through in groups, or a quick survey on the readings; these are graded primarily based on *completion* to help reflect your attendance and engagement.

Project Collaboration: Collaboration and close work with your peers on projects is allowed and encouraged, but please take ownership of your own copy of the code and your own understanding. For example, you may discuss details of how to address a bug or how to structure your program, but you should type your own individual implementation, compile and run it on your own, and include plenty of comments according to your understanding. **Always include a brief *project collaboration statement* (less than one page) mentioning who you worked with in your project submission, what went well, and what could have gone better, as well as citing online resources or results from websites like stackexchange that helped.** If you find a solution to an issue or a suggestion for your project on the internet, that is totally fine, but please document what you are using, how you are using it, and include detailed comments with any borrowed code to demonstrate you understand it thoroughly.

Project Deadlines: Projects will be shared and graded via GitLab (see Course Tools document on Moodle for details). **Always share whatever you have for the project by the deadline, even if it is incomplete**. Your well-commented source code, complemented by notes on your confusions and debugging efforts, can often sufficiently express that you understood the assignment even if it is not complete, and I'm not here to overly penalize you for syntax issues. *Also, many of our projects can be extended or further improved – focus on getting a working example submitted by the deadline, but feel encouraged to continue to contribute to project code past the deadline and/or attempt Bonus portions at any time for extra credit; drop me a note if you have any questions on this.*

Project Extensions: Life happens. If you need a project extension for any reason, send me an email with as much heads-up as you can (at least 24 hours when possible). I will typically readily grant you 1 extra day to submit projects if you request it; I'll just ask you to share what you have as of the deadline for reference, and only the version that you submit the following day will be graded. Extensions of more than 24 hours can be discussed on a case-by-case basis and will come with reduced credit. Try not to make a habit of it – if you request more than 1 project extension, I'll have you come in for a one-on-one appointment with me to discuss your experience and some time management strategies for future projects or submit a time management plan for your future projects.

Early Bird Special: Really, don't leave these projects to the last minute. In fact, if you submit a project *early*, you can get 1% point extra credit for each 24 hours early you submit, up to 5%.

Academic Honesty: You are expected to follow the project collaboration guidelines, including *always* acknowledging any peers who aided in your understanding or implementation, as well as documenting and demonstrating your understanding of any borrowed/found code or suggestions from the internet, textbook, or other sources. The projects you submit, as well as your midterm/final exams, will be evaluated individually, and as such should demonstrate your individual understanding of the material. Where appropriate, this should include comments in your code and descriptions of your thought process. **You may not make the solutions to any exercise, project, or exam available to anyone else at any time**. If you have doubts about whether you may collaborate and how, please ask via the anonymous feedback form on Moodle or reach out individually. In all cases, you are bound to the Amherst College Honor Code, and violations of these expectations could result in disciplinary actions.

Building trust and having confidence in your individual ability to demonstrate your understanding are really important components of an inclusive classroom community, as well as critical ideals towards developing into a thoughtful and effective computer scientist.

Accessibility: As the instructor of this course, I endeavor to provide an inclusive learning environment. However, if you experience barriers to learning in this course, let's connect to discuss ways to best support your access (send me an email for personal requests, or use the anonymous feedback form on our Moodle page as needed). If you have disability-related circumstances and are seeking academic accommodations (e.g. extra-time testing, reduced distraction test area, short breaks as needed, note taking assistance, etc.), Accessibility Services is eager to assist us with identifying reasonable accommodations for the course. For exam-related accommodations, please coordinate with Accessibility Services well ahead of time (at least two weeks before exam) to ensure we can work with them to support your needs. They can be contacted at accessibility@amherst.edu.

**Some course materials are adapted from the "Dive Into Systems" community and from Professor Scott Kaplan; please feel free to view prior versions of the course at this link.*