

CS 2110 [Online] – Computer Organization and Programming Syllabus – Spring 2027

Instructor: Dr. Mark Moss {mmoss7@gatech.edu}

Instructional Staff	Sections	Lecture Times & Locations	Office Hours & Location(s)
Dr. Mark Moss	Online	Most Lectures & Assessments Completed Asynchronously Some In-Person Assessments Required	To Be Announced

Course Description

This course fundamentally focuses on how computing systems are structured at a fairly low level. For example, most computing systems can be envisioned as layers of complexity, where the components of a lower layer are combined to create more powerful components and capabilities for the level above. In this course, we examine some of these relationships between adjacent levels:

- How transistors can be used to implement such logic gates and key functions like NOT, AND, and OR
- How logic gates can be used to implement combinational logic components such as decoders, multiplexers and adders
- How logic gates can also be used to implement sequential logic components such as latches, flip flops, registers and memory
- How combinational and sequential logic components can be combined to implement key (micro)architecture components such as finite state machines, special- and general-purpose registers, arithmetic logic units, and buses & gates (e.g., with tri-state buffers)
- How (micro)architecture components can be used to implement a set of assembly language instructions to provide capabilities such as: executing arithmetic and logical operations; moving data between registers and memories; and, controlling the flow of program execution
- How combinations of assembly language instructions can be used to implement common higher-level language capabilities such as sequence, selection, iteration, calls & returns, managing variables, and creating composite & dynamic data structures

Since many of you are focused on computer science, you might wonder why a course like this is required. Let me offer an analogy: consider someone who is able to drive a car. What do they need to know? How the car should be started (e.g., ignition), how the gear shift can be used to select the desired direction of travel, how the steering wheel can be turned to control the car's orientation, and how the accelerator and brake can be used to manage the car's speed. Does the driver have to know exactly how the engine, transmission, or braking systems work? In almost all cases, no: knowledge of how the "lower level" supporting systems function is not needed. However, having a better understanding of how those systems work can make you a more knowledgeable and better driver – for example: improving your understanding of your vehicle's capabilities; knowing how to make certain repairs without assistance; and, being able to diagnose problems accurately based on the vehicle's behavior, are all skills that come with that deeper knowledge.

We understand that you might not need to use your knowledge of digital circuits, instruction set architectures, or assembly language in all of your future specific assignments. You might accept roles or tasks that have a minimum requirement of understanding how to write programs in a higher-level language. We firmly believe, even in these cases, that having this knowledge will give you a better perspective on how to implement systems using that higher-level language more effectively and efficiently. And if the role or task is related to one or more specific topic areas – for example, computer security, language development, embedded computing, systems programming, networking, or a myriad of related fields – then this knowledge will be very important, if not essential, for your success.

Course Objectives

1. Understand the structure and operation of a modern computer from the transistor-level upwards
2. Understand basic hardware concepts including bits, combinational & sequential logic, digital circuits, registers, and memory
3. Convert between various data representations/formats including binary, octal, decimal, hexadecimal
4. Convert between various arithmetic numeric systems including unsigned, signed, two's complement and IEEE Floating Point
5. Construct state machine diagrams and then implement them as finite state machine circuits
6. Understand the structure and operation of a conventional von Neumann model computer architecture
7. Understand the structure and function of machine language instructions and symbolic assembly language instructions
8. Understand basic software concepts of computer systems such as the runtime stack, memory heap, and simple I/O devices
9. Introduce the C language with emphasis on translation to the underlying assembly and machine language formats

Course Delivery

All content is delivered asynchronously, with pre-recorded videos, Honorlock-proctored weekly quizzes, and week-long projects. There will also be assessments - including a Final Exam - that must be taken in person. Please refer to Canvas - and to the Living Schedule - to get a better idea of the "pace" of the course (i.e., the dates and times of these events).

Prerequisites

Students are expected to have taken two (2) computer programming courses at least one of which has been taught in Java. Students taking this course typically would have taken CS1301, CS1331, and CS1332:

- CS1301 is a basic programming course taught in Python. Students would learn basic programming concepts such as variables and constants, datatypes, operators and expressions, assignment statements, selection structures, loops, functions, etc.
- CS1331 is a course in Object Oriented Programming taught in Java so students are exposed to C-Style syntax.
- CS1332 is a thorough course in data structures again taught in Java.

The student entering CS2110 should know how to write and debug high-level language programs and have a solid familiarity with computing language syntax.

Course Workload

The course has exams, quizzes, projects, and a final exam:

- 40% In-Person Exams (4 exams, 10% each)*
- 28% Weekly Quizzes (14 quizzes, 2% each, taken remotely via a proctoring service)
- 12% Projects (best 4 out of 5 projects, 3% each)
- 20% In-Person Final Exam*

***In-person attendance is required for certain events.** In those cases, missing the event – which includes being more than 15 minutes late per Georgia Tech policy – will result in a zero for the event score unless prior arrangements have been made with the Instructional Staff and Division of Student Life Office.

Books

There is **no required textbook** for this course, but other resources might be recommended and/or made available primarily via Georgia Tech's Online Library.

Course Rules

1. **Reasonable accommodations will be made for students with verifiable disabilities.** In order to take advantage of available accommodations, students must register with [Disability Services](#).
2. **Final grades will use the scale:** [100:90] = A, (90:80] = B, (80:70] = C, (70:60] = D, (60:0] = F.
3. Overall course scores will not be rounded up by default. Also, there will be very few (if any) extra credit opportunities provided in this course, so you are encouraged to make the most of all assignments that are already designated in the course schedule.
4. **Please work with the Division of Student Life Office** if you encounter any major personal problems (e.g., family, illness, etc.)
5. All assignments must be turned in on time. **Late submission scores are capped using the late assignment window** as follows:
 - On time: 100% of the maximum score
 - After deadline, before 24 hours after deadline: 85% of the maximum score
 - After 24 hours, before 48 hours after deadline: 70% of the maximum score
 - After 48 hours: 0% of the maximum score
6. **You are also responsible for ensuring that what you turned in is what you meant to turn in.** If you have any problems submitting via Canvas, email the Instructional Staff what you would have submitted BEFORE the end of the late assignment window. No submissions will be accepted after the submission window (later than 48 hours) has closed.
7. **In-person events must be taken at the scheduled date, time, and location.** Missing an in-person event – including the Final Exam – will result in a zero (0) score for that event. You must contact the Instructional Staff in advance if you have an Institute-approved excuse to reschedule any in-person event.

8. **The deadline for a regrade request is one week after an assignment grade has been posted.** Regrades are only for correct answers accidentally marked as partially or completely wrong. Requests to simply increase partial credit will not be honored. Also, making a regrade request permits the Instructional Staff to review the entire submission for possible score changes.
9. You will be signed up for Piazza and the link will be made available via Canvas. The Piazza forum is a place for you to post questions about course concepts, assignments, projects, etc. **If you believe that your Piazza post contains any information that might provide answers towards a graded event – which might then constitute an Academic Honor Code violation if viewed by other students – then please ensure that your post is marked “Private”** so that it is only viewed by the Instructional Staff. Also, questions initially marked “Private” might be converted to “Public” (and thus visible to other students) per the discretion of the Instructional Staff.
10. **The Canvas and Piazza announcements must be read every day.** Course changes will be announced via these channels, and being unaware of these changes will not be a sufficient excuse if your score on an assignment is adversely impacted.
11. Recordings of lecture, lecture materials and other artifacts are copyrighted by your instructors. **No non-educational reproduction of any materials, including lecture records and recordings, is permitted.**

Academic Misconduct

The goal of all assignments in CS 2110 is for you to learn. Learning requires thought and hard work. Copying answers thus prevents learning. More importantly, it gives an unfair advantage over students who do the work and follow the rules.

- **As a Georgia Tech student, you have read and agreed to the [Georgia Tech Academic Honor Code](#).** The Academic Honor Code defines Academic Misconduct as “*any act that does or could improperly distort Student grades or other Student academic records.*”
- You must submit an assignment or project as your own work. **No collaboration on answers is permitted. Absolutely no code or answers may be copied from others. Such copying is Academic Misconduct.**
- Using code from GitHub, via Googling, from Stack Overflow, etc., is Academic Misconduct (Honor Code: Academic Misconduct includes “*submission of material that is wholly or substantially identical to that created or published by another person or persons*”).
- Publishing your assignments on public repositories, GitHub, etc., that is accessible to other students is unauthorized collaboration and thus Academic Misconduct.
- Suspected Academic Misconduct will be reported to the Division of Student Life Office of Student Integrity. It will be prosecuted to the full extent of Institute policies.
- We use accepted forensic techniques to determine whether there is copying of a coding assignment.
- **If you are not sure about any aspect of this policy, please ask your lecturer.**

Using AI-based Systems

Anything you did not write – or properly document – in your assignment will be treated as an academic misconduct case. From this perspective, **our default policy on utilizing AI-based systems is that submitting an assignment with code or text from an AI-based system (e.g., ChatGPT, Copilot) will be considered academic misconduct.** However, some assignments might include guidance that permits some level AI-based system to be utilized without penalty. In those cases, the level of AI usage that is permitted will be clearly defined for that specific assignment.

Still, **AI-based systems are permitted for helping you to improve your personal understanding of the course concepts as opposed to generating and submitting deliverables.** If you are unsure where the line is between collaborating with AI to learn, and copying AI to complete your work, then we recommend the following heuristics:

- **Heuristic 1:** Never hit “Copy” within your conversation with an AI assistant. You can copy your own work into your own conversation, but do not copy anything from the conversation back into your assignment. Instead, use your interaction with the AI assistant as a learning experience, then let your assignment reflect your improved understanding.
- **Heuristic 2:** Do not have your assignment and the AI agent open at the same time. Like the previous heuristic, use your conversation with the AI as a learning experience, then close the interaction down, open your assignment, and let your assignment reflect your revised knowledge.

CS 2110 [Online] – Computer Organization and Programming (Moss)

- **Heuristic 3:** Do not integrate or activate AI-based tools directly into your composition environment, just as you should not let a classmate write content or code directly into your submission. Also, avoid using tools that directly add content to your submission.

Deviating from these heuristics does not automatically qualify as academic misconduct; however, following these heuristics (in letter and spirit) essentially guarantees your collaboration will not cross the line into misconduct.

Coding Guidelines (C/C++ code)

1. You must turn in ALL files specified in the “Deliverables” section of the assignment instructions. We reserve the right to impose a penalty on submissions that do not follow the given submission directions.
2. You must provide a ***Makefile*** that compiles and links your code by default. **If you are given a Makefile with the project, we expect your code to compile with make. (don’t worry, we’ll explain what all this means later)**
3. Your code must compile with **gcc** on Ubuntu 22.04 LTS. If your code does not compile, then you will receive a zero/0.
4. You will be penalized if your code produces warnings when compiled with the given Makefile, or the following flags if no Makefile is provided: **gcc -Wall -Wextra -Wstrict-prototypes -pedantic -O2**
5. Code should be well commented and use a clean, consistent (readable) style (i.e., proper indenting, etc.). We may impose style requirements and deduct for non-conforming solutions. This is **not** the [obfuscated C code competition](#)!

Statement for not mentioned policies and issues: Any policies and issues not explicitly mentioned in this syllabus will follow policies and procedures according to the Georgia Institute of Technology: http://policylibrary.gatech.edu
