Course

ECEG 431 (Crosslisted as ECEG 631)
Computer Systems

Catalog Description:

This course provides students the concepts, technologies, and skills needed for advanced study in computer engineering. It includes aspects of computer organization, computer architecture, operating systems, networking, and performance evaluation and the relationship between them.

Prerequisite

CSCI 206 or ECEG 247 or ECEG 347

Instructor

Stewart J. Thomas
Office & Contact: ACET 322 // 570-577-1515 // sjt015@bucknell.edu
Office Hours: Its easier to do things by appointment– https://calendar.google.com/calendar/u/0/appointments/schedules/AcZssZ1vYuVs91KW2CCKUYIiFsVQhW8HQJORxkcWLrK21fOXv54IMliKO-al4DYdFoGZlstI4M2z8FKL (this link is also in Moodle)

I'm typically in my office during the week if you want to stop by, or just reach out and make an appointment. The easiest way is to send me a google calendar invite. Typically, most students in ECEG 431 prefer individual appointments to review code, ask concepts, etc. over group office hours and I've found appointments easier and more efficient for everyone.
Default zoom link: https://bucknell.zoom.us/j/8906027786 if you want to use Zoom (unless a calendar invite has a specific zoom link)

Also, if you want to reach me for a quick question, you can:

- Emails tend to work most reliably. [A note: PLEASE do not send me screenshots or worse: phone pictures of a screen displaying text. Ugh. If you do, you will probably get a photograph of my response back.]
- Text me at 570-577-1515 This is my office phone number through Zoom Phone. It is very strange though in how it pings over to my mobile. Sometimes it works really well, and other times it will just not forward the text to me! I don't know why. (just include a note like "It's Mark Rober from ECEG 431. On project…" that way I won't see a random

phone number.

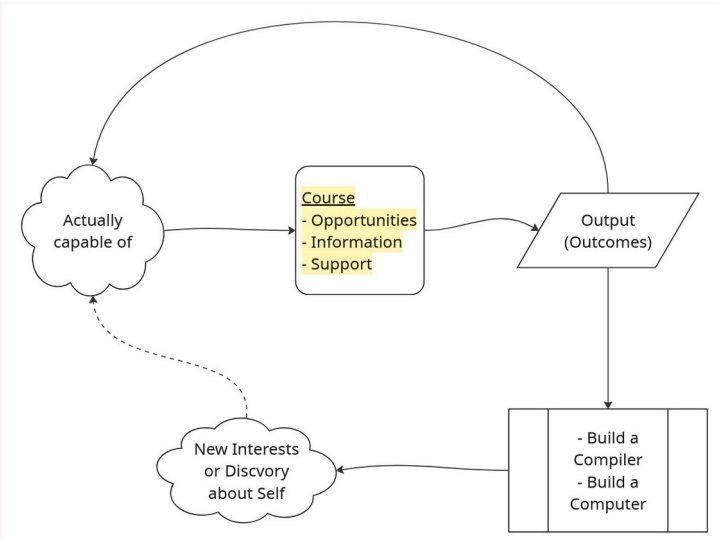
Lecture Meetings

```
ECEG 431 01 :     M,W,F  03:00PM – 03:50PM,  DANA 303
                  Tu     08:00AM – 09:50AM,  DANA 305
```

Course Description

By the end of this course, you will have built a general-purpose computer from scratch. Well, actually from transistors. Starting with the fundamental idea of transistors, logic gates and Boolean algebra, you will build computer hardware from logic gates and arrange them to form an architecture. You will program a basic single-task operating system and create a compiler that converts your OS programs to machine code that will run on your architecture. From here, you will develop other general-purpose applications that run on this architecture which you've built. Get ready!

You've probably seen a lot of these pieces before and have already spent time building up your own knowledge and abilities related to programming and digital circuits. This course will let you build on your own set of capabilities and support you on your way to building a working computer and compiler. The outcomes (mostly consisting of course outcomes as well as the individual projects to complete) are built to help you both get to your end goal, and also help both reinforce and expand the your own capabilities. By the end of this course, you will hopefully learn a bit about yourself (perhaps how you mange complex projects, time, plan projects) as well as your interests (or *dis*interests). A common outcome is students discovering they wish they had more programming courses, or wishing to be able to go much deeper into computer architecture. This course will help you ge to that point as well as help you prepare to learn these things on your own if you wish.



Course Learning Objectives

This course will give you opportunities to:

1) Design the hardware elements and architecture of a basic computer using gate-level hardware description language. [A1,6, D1,2]
2) Write software that converts programs written in a high-level language to a set of

instructions that run on a basic computer (i.e., compilation). [A1,7, D1,2]

3) Design, implement and debug assembly code for a specific architecture (that you have built). [A1,6, D1,2]
4) Design object-oriented software applications that run on a computer you have designed. [A1,6, D1,2]

Note: A1, A5, A6 refer to ABET student outcomes 1, 5 and 6. See: https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2025-2026/#GC3
D1, D2, D3 refer to Bucknell's ECE departmental outcomes. See: https://www.bucknell.edu/academics/college-engineering/majors-departments/electrical-computer-engineering/abet-accreditation

Approach

This is a hands-on course. Each project will build on the previous project. For each project, you will be given a design document and an Application Programing Interface (API) that specifies an input/out interface contract, an executable solution, a test script that describes *what* the module is supposed to do or behave and a detailed implementation plan that proposes *how* to build the project. The projects will be spread out over the semester.

For each project,

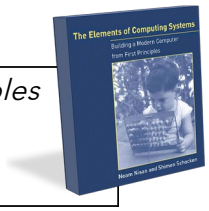
- you will be expected to read the associated chapter in the textbook
- submit reading questions and thoughts into a reading-chapter-check-in on Gradescope
- work and discuss the project in-class
- turn in your code to Gradscope for evaluation through the automatic grader
- walk me through your design once complete in a sit-down code review
- and then submit a short reflection (in Gradescope) on the project once you have completed it

It will all make sense once we get moving in the course! Gradescope will have the due dates to help keep you on track. A link to gradescope is in Moodle, or you can visit gradescope directly.

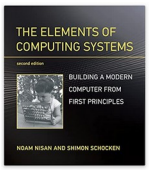
Course Materials

The book "Elements of Computing Systems" is required for this course. This book is a great resource for working on the projects and well-written. It gives you specific and helpful advice for working through the course. It is also inexpensive compared to most other textbooks. The first half of the textbook is available online at the companion website if you have trouble receiving your copy. I expect you to keep up with the readings.

The book's companion http://nand2tetris.org will be an invaluable resource to you as well. It contains the projects we will be doing as well as the suite of software you will need to download and use in the course.



The Elements of Computing Systems: Building a Modern Computer from First Principles
by Noam Nisan and Shimon Schocken
ISBN: 9780262313209 | Copyright 2005



The Elements of Computing Systems: Building a Modern Computer from First Principles, 2e
by Noam Nisan and Shimon Schocken
ISBN: 9780262539807 | Copyright 2021

Purchasing options: [see: https://www.nand2tetris.org/book]
Rent or buy from MIT press (e-rent: $25.00 / 4 months):
https://mitpress.ublish.com/book/elements-of-computing-systems-2#purchase
Rent or buy from amazon (buy new for $38.00 at time of syllabus):
https://www.amazon.com/Elements-Computing-Systems-second-Principles/dp/0262539802

Most links above are for the 2nd edition which is easier to find. Either edition is fine for this course as they both contain mostly the same information. The software you will need to download (whether onto your own device or onto computers on campus) is at:
https://www.nand2tetris.org/software

The details

Attendance Policy

Regular attendance in class is expected. If you have a reason that you must miss class (university excused absence, interview, society travel, illness), please talk to me. In general, you will be responsible for learning the missed material and meeting deadlines. However, I also understand that emergencies happen. ECEG 431 as a course is designed to have flexible deadlines but this has been known to exacerbate time-management problems for many students and negatively impact their learning. All deadlines (with extra penalty-free "late" extensions for projects) will be seen in gradescope and it is your responsibility to meet these.

Health/Isolation Policy

If you are not feeling well or are expected to isolate/quarantine because of disease exposure, please email your instructor and team prior to class or lab and do NOT come to class. For all medical issues, please go see Student Health. It is not my job to judge your health. If your health (physical or mental) is causing continued issues with class attendance or course progression, unfortunately this will negatively impact your grade unless you have worked with student health and the College of Engineering Dean's Office for flexibility notices. To be completely transparent, I am mentally and emotionally exhausted from supporting students who are quite literally months apart in terms of course progression for the past few years. I will do my best to lay out clear deadlines and it is your responsibility to work with your instructor and team members to catch up and make up what you missed during an absence. The good news is that in this course there are multiple paths to earning a specific grade. Please plan accordingly, and do not put things off! This course will "snowball" and keep growing.

Homework policy

Collaboration on homework is heavily encouraged. This includes verbal discussion of problems and the use of scratch paper or white boards to discuss concepts and approaches to solving specific problems but DOES NOT include sharing specific code implementations. **The nature of the projects in this class means that everyone's projects will be very different.** I cannot stress this last part enough. You may not have seen enough code yet to know, but code solutions are as unique as essays. It is surprising, but very true.

You would probably expect two essays by different students on something such as *Current Politics and Social Cohesiveness of Former Soviet Nations* to be quite different in form, content, structure, conclusions and style. You may not realize it, but **code is just as expressive**

as essays, and it is very obvious when two people have moved from collaborative discussion to sharing code. It is for this reason that discussing algorithms or approaches or class/module structures is most useful and helpful compared to sharing specific implementations—*the context is key*. Someone else's specific solution may not work with your personal approach to the problem. Therefore, **unless specifically authorized in an assignment, do not copy someone else's work**. If you have any questions about collaboration, *please ask!*

The above paragraph also applies to aid you may find on the Internet or to using chatGPT (or other AI/LLM tools). I'm somewhat OK with chatGPT, if you are finding it to support your learning and not just "vibe-coding" with what it spits out. My main concern is the mental crutch it becomes. Think about this: What kind of engineer do you want to be after you graduate? One who relies on LLMs for everything? Or one who avoids them? Or one who uses them to support their work in other ways?

If you find LLMs to be something that *you require* and cannot work without, then stop using it. If you do use chatGPT, then please cite this within your code. Don't ask it for entire solutions, but work with the AI to develop a structure for your code, or maybe to do something specific like remove spaces from text. I've found it somewhat helpful when developing a program as it is similar to talking through design decisions with another person. However, it is up to you to understand and be able to fully explain any code you submit.

All that said: I have observed that students who use either whiteboard, or drawing apps on an iPad and immediately jump to drawing, tend to be able to explain their code very well, and also be able to talk to others about strategies. Taking time to think through and plan how functions or data objects interact is important and worth taking a few moments to consider. It will also help you collabarate and think through possible solutions with others around you!

Collaboration on an exam in any form is of course strictly forbidden. You may *not* share notes, homework, laboratory assignments, textbooks, or any other materials with any other individuals—even if they were your laboratory or homework partner. During the exams, you may not discuss any ECEG 431 materials until notified by the professor.

Remember, according to the Honor Code, it shall be the responsibility of the student, when in doubt, to ask the instructor what is or is not academic dishonesty.

Examinations

There will be one optional examination in this course on **Tuesday 7 October** (in lab) and in

lieu of a final exam, we will hold an after-action review during the final exam period on **Thursday, 18 December** from **3:30–6:30 pm** (preparing and participating in-person is worth a "check" --- It is of course optional, so plan either your travel schedules, or your work-load during the semester accordingly). We can easily add a second exam opportunity if the class wants. We can schedule this for some time near the end of the semester.

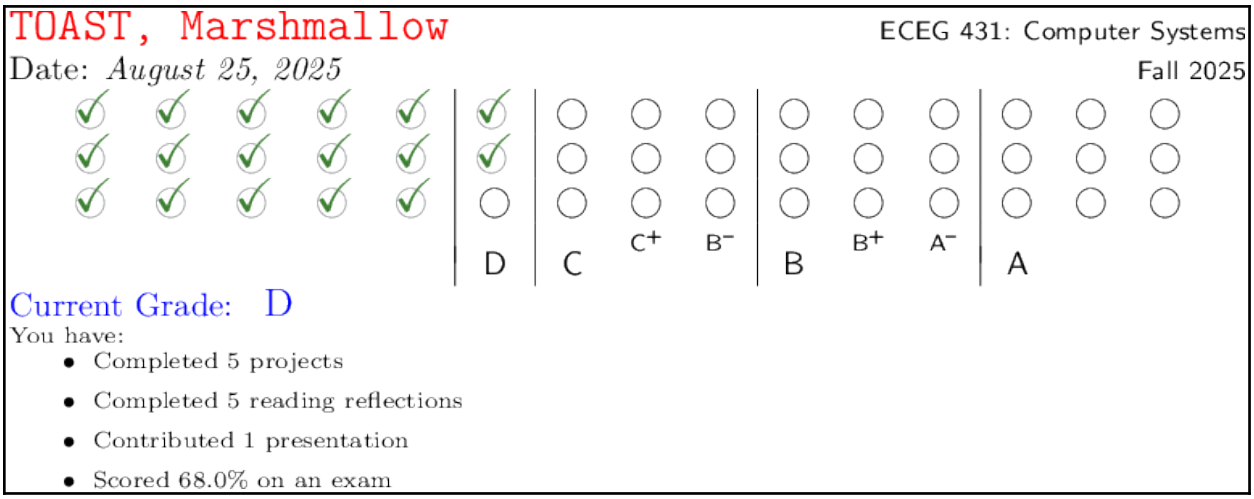
On-Line Grade Book

We will use Gradescope to turn in all assignments and I will work to put all your graded items into Moodle. I will also provide semi-regular grade updates to you via your Bucknell email address. This may change to a mix of emailing, and an online gradebook hosted at Bucknell. I will link to this from Moodle if we do begin using a system besides email. I will do my best to accurately record the grades throughout the semester. However: It is your responsibility to verify the grades I have recorded are accurate. If you find an inaccuracy in the grade book let me know as soon as you can.

Class Cancellation

Notifications of class cancellations will be made through Moodle with as much advance notice as possible. It will be both posted on Moodle and sent to your Bucknell email address. If you don't check your Bucknell email account regularly or have it set-up to be forwarded to your preferred email account, you may not get the message. Please check Moodle and your Bucknell email before coming to class.

Grading



This course uses a somewhat unique grading system. It changes a bit from year-to-year. Your main goal is to complete the 12 projects. Doing so (along with the associated book readings)

would guarantee you an A- in the course. However, there are other options to increase your grade:

For each project you complete, you earn two checks in the grade card. Each reading reflection earns a check. An exam (only one exam is slated for now, but we could add another) can earn one check for being above a 68% and an additional check for being above an 85%. A "scholarly contribution" can also earn you a check.

For this class, "scholarly contribution" simply means contributing back to our learning community. This mostly means short informational presentations to the class, but may include giving helpful demonstrations, presenting helpful code snippets, etc. This class tends to spark interest in something, so go off and learn something cool and come back and tell us.

In the past, we've also had an After Action Review that would earn a check, or a end-of-semester ePortfolio that would earn a check.

Grading Policy

This grading policy mostly relies on a contract of completing the projects. What therefore constitutes completing the projects? To complete a project, you will need to:
- Demonstrate the working project to your professor (typically via gradescope),
- Briefly sit down and explain your code – what works well, strategies you used, problems you ran into. This is intended to model a "code review." If you are at or above 87% on gradescope, and can explain our code then you are complete.
- You will then get a Complete/Incomplete measure on the project. If incomplete, keep working! You will need to be able to explain to me your code to demonstrate you know what you wrote.
- Once complete and checked off with me, write a brief reflection analyzing your process and thinking about what worked well, what was difficult, what you learned (either about the project or yourself). This is the "formality" to close out the project and move on.
Hand that in through gradescope, and give yourself the two checks. (This year, I am planning on offering a single check for a project that is partly completed. Meaning you are above 65% on gradescope, although the specific number may change!)

I am looking for well-written code that you are able to explain back to me and meets the project criteria. Perfection is not the goal. Do a project as best you can, explain it to me, reflect on what went well, and keep on going. In this class, your code will build on what you've previously written. So reflecting on what worked well, or what needs to change will

help you develop your project.

Projects will be demo'ed during meeting times (ideally lab meetings). Please sign up for a time slot to demo and explain your system. Otherwise, lab time will be used to work on projects and ask questions/seek help. I'm here to help you through this process! There may be presentations during lab as well.

I will limit you to demo'ing two projects per week. This is to avoid the intense bombardment of demonstrations all at once. Keep up with the projects. We will start individually, but around Project 6 will begin to work in teams if you want.

Access and Accommodation statement

Any student who may need an accommodation based on the impact of a disability, should contact the Office of Accessibility Resources (OAR) at 570-577-1188 or OAR@bucknell.edu. The office will help coordinate reasonable accommodations for those students with documented disabilities. Please visit https:www.bucknell.edu/Accessibility for more information about the OAR.

Diversity and Inclusion statement

The Department of Electrical and Computer Engineering is committed to our students, staff, and faculty. You are a valued, respected, and essential member of our community regardless of race, ethnicity, or nationality, gender, gender expression or sexual orientation, religion or belief system, economic status, or ability. We are together a community of learners who strive to offer a safe environment for learning, growth, inquiry, and the respectful sharing of ideas for all. By virtue of joining this community, all members commit to welcome others in the same manner.

We have a moral and professional obligation to always treat each other with respect and dignity, even when we disagree. We will not tolerate mistreatment or disrespect of persons for any reason. We all have to express this obligation through two actions:

* Actively contribute to creating an inclusive, collaborative learning environment.
* Take action to correct injustice when others do not.

Engineers investigate and solve problems that affect people and we must hold paramount the unquestioned support and value of all persons. The difficult work of engineering can only be done when we belong to and feel safe as members of a supportive community.

Your suggestions to help the ECE Department meet this commitment are encouraged and

appreciated. If something occurs in class that makes you feel uncomfortable, please talk to me about it. If you are not comfortable doing that please talk to your other instructors, the department chair Stu Thompson (mst008@bucknell.edu), and Associate Dean Terri Norton (trn005@bucknell.edu). You may also file a bias incident report (https://www.bucknell.edu/life-bucknell/health-wellness-safety/bias-incident-policy). This report may be filed anonymously if you so choose. The ECE Department commits to supporting students expressing concerns and/or reporting bias to empower them in any follow-up actions and to ensure that they are protected from repercussions of any kind.

Student Mental Health Statement and Resources

In this classroom and on Bucknell's campus we support mental health efforts. Any student who is struggling and believes this may impact their performance in the course is encouraged to contact their Associate Academic Dean or the Dean of Students at 570-577-1601 for support. Furthermore, please approach me if you are comfortable in doing so. This will enable me to provide resources and support. If immediate mental health assistance is needed, call the Counseling & Student Development Center at 570-577-1604 (24/7).

You are also encouraged to register for a Togetherall account to access peer to peer, mental health support, complete self-paced courses on a wide range of topics, or check out the catalog of articles and activities focused on investing in mental health and wellness.

Masking policy

Apparently COVID-19 hasn't disappeared. Boo. Here's Bucknell's policy:

https://www.bucknell.edu/life-bucknell/covid-19-information/students-families

Bucknell University Expectations for Academic Engagement

Courses at Bucknell that receive one unit of academic credit have a minimum expectation of 12 hours per week of student academic engagement (on average). Student academic engagement includes both the hours of direct faculty instruction (or its equivalent) and the hours spent on out of class student work.

Honor Code

The Bucknell University Honor Code states that

1. *I will not lie, cheat, or steal in my academic endeavors.*
2. *I will forthrightly oppose each and every instance of academic dishonesty.*

3. *I will let my conscience guide my decision to communicate directly with any person or persons I believe to have been dishonest in academic work.*
4. *I will let my conscience guide my decision on reporting breaches of academic integrity to the appropriate faculty or deans.*

In choosing to remain in this course, we are also choosing to uphold this honor code. Just as students have faith the faculty act justly and fairly, so I have faith that you will act accordingly.

> *"Students are responsible with for knowing what constitutes academic misconduct and for asking their professors for clarification whenever they have questions."*
>
> *--- from Bucknell Honor Code, FAQS for students*

Often, my exams will be take-home, or even unproctored. In these cases, I will ask that you sign a statement on the exam stating that the Honor Code was upheld. If you have any questions regarding what constitutes academic dishonesty, it is your responsibility to ask. More information on policies and procedures relating to the Honor Code and academic dishonesty can be found at https://www.bucknell.edu/AcademicResponsibility .

ECE Lab Policy User Agreement

To gain access to the ECE labs, you will need to read and electronically sign the ECE lab User Agreement. To use NI CDS (Multisim/Ultiboard), you will need to sign the Student Install Agreement.

Please sign the policy agreement before coming to lab.

ECE Lab User Agreement
https://forms.gle/P4ohhFsqSPtJ25vi8

NI CDS Student Install Agreement
https://forms.gle/dQVbFzm1UiY3vCUn7

Schedule at a glance

While a regular schedule of assigning labs on Mondays and then working and demo'ing in labs would be ideal, that schedule doesn't work perfectly with how the projects are structured. The current schedule compresses a few the projects that go quicker and gives a bit more time to the projects that tend to be confusing. However, this gives an overview of what the projects are going to do:

1. Getting started: Demonstration of the final computer built in the course; The abstraction/implementation paradigm and its role in systems design; Overview of the Hardware Description Language (HDL) used in the course; Designing a set of elementary logic gates from primitive Nand gates; Implementing the gates in HDL.
2. Boolean arithmetic: Using the previously built logic gates to design and implement a family of binary adders, culminating in the construction of a simple ALU (Arithmetic Logic Unit).
3. Sequential logic: Design and implementation of a memory hierarchy, from elementary flip-flop gates to single-bit cells to registers to RAM units of arbitrary sizes.
4. Machine language: Introducing an instruction set, in both binary and symbolic versions; Discussing some trade-offs related to its design; Writing some low-level programs in its assembly language.
5. Computer architecture: Integrating all the chips built in weeks 1-3 into a computer platform capable of running programs written in the instruction set introduced in week 4.
6. Assembler: Basic language translation techniques (parsing, symbol table, macro-assembly); Building an assembler for the assembly language presented in week 4.
7. Virtual machine I: Pushdown automata, stack machines, the role of virtual machines in modern software architectures like Java and .NET; Introduction of a typical VM language, focusing on its arithmetic commands. Building a program that translates from this VM language into the assembly language presented in week 4.
8. Virtual machine II: Completing the design of the stack machine and its associated VM language, focusing on flow-of-control and subroutine call-and-return commands; Completing the implementation of the VM translator.
9. High Level Language: Introducing a simple high-level object-based language with a Java-like syntax; Discussing various trade-offs related to the language's design and implementation; Writing a simple interactive game and running it on the

computer built in weeks 1-8 (using the compiler and OS that will be built in weeks 10-13).

10. Compiler I: Context-free grammars and recursive ascent parsing algorithms; Building a syntax analyzer (tokenizer and parser) for the high-level language presented in week 9; The syntax analyzer will generate XML code reflecting the structure of the translated program.

11. Compiler II: Code generation, low-level handling of arrays and objects; Morphing the design of the syntax analyzer into a full-scale compiler; This is done by replacing the routines that write passive XML with routines that generate executable VM code for the stack machine presented in weeks 7-8.

12. Operating system I: Discussion of OS/hardware and OS/software design trade-offs, and time/space efficiency considerations. Design and implementation of classic arithmetic and geometric algorithms (needed for implementing the OS's *Math* and *Graphics* libraries).

13. Operating system II: More classic mathematical, memory management, string processing, and I/O handling algorithms, needed for completing the OS implementation.

14. More fun to go: Discussing how the computer system (hardware and software) built in the course can be improved along two dimensions: optimization, and functional extensions; Proposed directions for further explorations.