CS 485 Compilers (Sp25)

Syllabus

Welcome to CS 485: Compilers! This course teaches students how to build an optimizing compiler for a Java-like object-oriented programming language. Topics include program semantics, typechecking, intermediate representations like single static assignment (SSA) form, code generation, and classic compiler optimizations including register allocation via graph coloring, peephole optimizations, and dataflow analyses. Students will also be introduced to functional programming, and its advantages for transformation-based programs like compilers that have strict correctness requirements.

Prerequisites

As a special topics course, this class officially has no prerequisites, and you're signing up for it at your own risk. If this course were to be offered as a regular course, the prerequisite would be CS 280 with a grade of C or better, or the equivalent at another institution. Taking CS350 in the same semester as this course, or prior to it, is recommended but not required, provided students have some prior knowledge of assembly. Students will also benefit from taking CS 341 concurrently or before this course, because there are some relevant theory topics to optimization. Finally, this course includes a non-trivial programming project. Students who have taken a software engineering course (e.g., CS 490 with me) or otherwise have software engineering experience will probably have an easier time.

The first homework assignment is due right at the drop date, and is intended to let you check that you have the basic skills that will be needed for this course.

Topics

- Functional programming
- Scoping
- Typechecking
- Operational Semantics
- Abstract Interpretation
- Intermediate Representations
- Code Generation

- Multi-language Projects
- Dead Code Elimination and Other Dataflow Analyses
- Local Optimizations
- Peephole Optimizations
- Global Optimizations
- Register Allocation
- Automatic Memory Management
- Exceptions
- Debuggers
- Linking, Loading, and Shared Libraries

Grading and Assignments

Your grade is composed of the following sub-scores (in no particular order):

- 60%: Programming Projects (5% for PA1 (Rosetta Stone), 17.5% for PA2 (COOL Typechecker), 17.5% for PA3 (COOL Code Generator), 20% for PA4 (COOL Optimizing Compiler))
- 25%: Exams (10% for the mid-term, 15% for the final)
- 15%: Participation & Professionalism

This class will be curved: when grading, I prefer to use the whole range available rather than scores in a tight range. That is, if an assignment is worth 10 points, I will give grades at all the points between 0 and 10. I will project your raw scores onto the final distribution twice during the semester:

- after the mid-term exam
- shortly before the final exam

You will be notified of your current projected class grade via email at each of these points.

Textbook

This course uses *Engineering A Compiler, 2nd edition* as the primary textbook. NJIT's library has copies available online. This textbook will be supplemented by other readings.

Participation & Professionalism

Your participation & professionalism grade is based on your interactions with the instructors and TAs: in-class, on the course discussion forum, in office hours, etc.

REMOTE PARTICIPATION

Generally this class does not support remote participation: teaching is much more effective, in my experience, when everyone is physically present. However, I understand that sometimes you are sick, traveling, or otherwise unable to come to class. I will arrange for remote participation in any particular lecture as long as you request it at least one hour in advance (if you're sick or in some other emergency) or 24 hours in advance (if you're traveling or otherwise planning to be unable to come to class). Notify the instructor via email if you need to participate in a particular class remotely.

ASKING QUESTIONS

There is a course Discord server which you can use to ask (and answer) questions about any of the course topics or for help with the homework. Participating on Discord is optional, but if you do participate in a productive manner (especially by answering other student's questions!), it will have a positive impact on your participation score.

OFFICE HOURS AND VIEWING TEST CASES

In addition to offering you a chance to ask questions about the lecture material, the instructor's office hours have a unique feature that can help you with the compiler's programming assignments (i.e., PA2-4). During the instructor's office hours (only—TA office hours do not count), once per assignment, each team (or student, if working individually) may ask to view the content of one test case from the grading server (you'll need to ask by test case number). Note that "per assignment" means that you can use this power only three times per semester, by default: once each for PA2, PA3, and PA4. You can only use this power in-person, and you cannot record the result electronically (e.g., by taking a photo). The idea here is to allow you to get "unstuck" if there is a particularly tricky test case that you're struggling with on the autograder. For that reason, you can only use this power once you have made a submission to the autograder that received a nonzero score on the relevant assignment.

You can gain more uses of this power per assignment by choosing a language from a language bucket other than Bucket 1; see the languages page.

Programming Projects

This course has four programming projects:

- **PA1: the Rosetta Stone**: Students will implement the same program in four languages, one from each of the following buckets:
  - *Bucket 1: Languages You Already Know*: Java, C, C++, or Python3
  - *Bucket 2: Languages With An Unusual Type System*: Kotlin, Rust, or Scala
  - *Bucket 3: Functional Languages*: OCaml or Haskell

    - *Bucket 4: Project Language*: Classroom Object-Oriented Language (COOL)
- **PA2: the COOL Typechecker**: the typechecker takes an AST (produced by a provided COOL parser) as input and produce either a type error or a special AST-related output format that serializes data structures produced during typechecking.
- **PA3: the COOL Code Generator**: the code generator takes the typechecker's output and produces x86-64 assembly. This assignment does not require any optimization, only correctness.
- **PA4: the COOL Optimizing Compiler**: the optimizing compiler is a modified version of PA3 that also optimizes the x86-64 code that it produces. This project is judged on the speed of the generated assembly code (in addition to correctness). Students are required to 1) implement some specific optimizations, including dead code elimination, and 2) match the performance of the COOL reference compiler's generated assembly. We will provide an anonymized leaderboard of student submissions (and reference implementations).

Each assignment includes at least one "checkpoint" to encourage you to start early.

All assignments except PA1 may be completed in pairs (but working alone is also permitted).

Exams

There are two exams in this course:

- a mid-term, which is held in class about halfway through the semester (worth 10% of your course grade)
- a final exam, which is held during the university-scheduled final exam slot (worth 15% of your course grade)

Both exams will cover a range of topics discussed in lecture and/or in the mandatory readings, from any time during the semester up to the point when the exam is held. The exam will be comprehensive, covering many of the topics we discuss; I may ask about anything we covered in class or that you were supposed to read. The exam will be conducted in person. Contact the course staff privately via email if you are not able to attend for any reason (e.g., you are sick or need special accommodations) and we will arrange an alternative. See the exams page for more information.

Collaboration Policy

Collaboration is generally encouraged in this course, as is consulting online resources. You are permitted to copy small amounts of code from any source except someone else's copy of an assignment, *as long as you cite your source*. "someone else's copy of an assignment" also

includes students not currently enrolled in the course - e.g., students who took this class in previous semesters or took classes that used similar projects at other institutions. To make this more clear, here are some examples of acceptable and unacceptable collaboration on a programming assignment in this course:

Acceptable collaborations:

- Discuss problems/solutions/anything with any number of other students (as long as you don't look at each other's code).
- Copy a short (about 10 lines or fewer - use your judgment) snippet from stackoverflow.com or a similar source, as long as you include a comment with the source URL.
- Copy code written by one of your teammates during a group project for another part of the group project.
- Copy code from the output of a generative AI tool such as ChatGPT that you prompted yourself, if you include a link to a record of your interaction with the model (e.g., ChatGPT's "share" feature) as a code comment.

Unacceptable collaborations:

- Copy code directly from another student on an individual project.
- Copy code from another group on a group project.
- Copy a significant portion (more than about 10 lines of code or a single method - use your judgment) of your assignment from the internet, even if you cite your source.
- Copy a short snippet from the internet without citing your source.
- Copy code from the output of a generative AI tool (such as ChatGPT) without citing your source
- Copy code from the output of a generative AI tool prompted by someone other than you (or your teammates, for a group project)

These rules are intended to mimic what is acceptable in industry when working as a software engineer: using the resources available to you, such as your teammates and the wider internet, is always allowed. But, it would be illegal to copy code from a competing company working on a similar product.

Generative AI Policy

You are permitted to use generative AI tools on any assignment in this course, as long as you include a record of your interaction with the generative AI tool in your submission. For example, if you use ChatGPT to help you with an assignment, you are required to include the output of the

"share" tool in OpenAI's interface as a code comment in your submission. You may choose how you include your prompts and responses in your submission (e.g., it's also permissible to include the full text of both your prompts and the output of the tool), but the following must be accessible to the course staff while grading:

- the name and version of the model being used
- the full text of your prompts
- the full text of the model's responses

Moreover, there is a specific style of prompt that is **forbidden** in this course when interacting with generative AI tools: you may not include text directly copied from the assignment descriptions in your prompts. Doing so is a violation of the collaboration policy and will be treated as cheating.

Consequences of Violating the Collaboration Policy

(From the University)

"Academic Integrity is the cornerstone of higher education and is central to the ideals of this course and the university. Cheating is strictly prohibited and devalues the degree that you are working on. As a member of the NJIT community, it is your responsibility to protect your educational investment by knowing and following the academic code of integrity policy that is found at: http://www5.njit.edu/policies/sites/policies/files/academic-integrity-code.pdf.

Please note that it is my professional obligation and responsibility to report any academic misconduct to the Dean of Students Office. Any student found in violation of the code by cheating, plagiarizing or using any online software inappropriately will result in disciplinary action. This may include a failing grade of F, and/or suspension or dismissal from the university. If you have any questions about the code of Academic Integrity, please contact the Dean of Students Office at dos@njit.edu"

Late Policy

PA1, PA2, PA3, and all checkpoint assignments may be submitted late, with an escalating "Fibonacci" penalty for each day beyond the due date. More specifically:

Days Late	Penalty
1	2%
2	3%

Days Late	Penalty
3	5%
4	8%
5	13%
6	21%
`n`	`fib(n + 2) %`

“Days Late” is always computed AoE, so if the assignment is due on a Monday and you submit it while it is still Thursday anywhere on Earth (e.g., at 5am Friday in Newark), you will be assessed a 5% penalty (3 days late).

PA4 may not be submitted late, because it is due on the same day as the final exam. After that point, the course is over and I will be computing final grades. No exceptions to this policy will be made.

Research

Your class work might be used for research purposes. For example, we may use anonymized student assignments to design algorithms or build tools to help programmers. Any student who wishes to opt out can contact the instructor or TA to do so after final grades have been issued. This has no impact on your grade in any manner.

Students interested in considering undergraduate research should make an appointment with the professor to talk about it. I am happy to discuss independent study projects, paid research work over the summer, research work for credit, graduate school, or any other research related topic. To make an appointment with the professor, send a calendar invitation at a time when my calendar shows that I’m free during “regular business hours” (roughly 9-5, Monday through Friday). Include enough information in the invite so that I know why you want to talk to me.

Acknowledgments

This course is heavily indebted to Wes Weimer’s courses that I took as an undergraduate: UVa’s CS 4610 and CS 4501. Thanks to Wes for sharing his materials!