CS 406: Compilers

Spring 2020

Lectures Mondays 4PM to 4:55PM, Wednesdays 8:30AM to 9:25AM and 2:30PM to 3:25PM, Room 211

Course web page https://hegden.github.io/cs406/

Piazza discussion page https://piazza.com/iitdh.ac.in/spring2020/cs406/

Instructor Nikhil Hegde (nikhilh@iitdh.ac.in) Office Hours: TBD

Prerequisites A solid grounding in C/C++, Java or some other high level programming language. A working knowledge of data structures. A grasp of the basic fundamentals of Automata Theory and Computer Organization and architecture.

Textbooks / References

1. Alfred V. Aho, Monica S. Lam, Ravi Sethi and Jeffrey D.Ullman: Compilers: Principles, Techniques, and Tools, 2/E, AddisonWesley 2007.
2. Fisher and LeBlanc: Crafting a Compiler in C.
3. Andrew Appel: Modern Compiler Implementation in C/ML/Java, Cambridge University Press, 2004
4. Dick Grune, Henri E. Bal, Cerial J.H. Jacobs and Koen G. Langendoen: Modern Compiler Design, John Wiley & Sons, Inc. 2000.
5. Michael L. Scott: Programming Language Pragmatics, Morgan Kaufman Publishers, 2006.

We will primarily be referring to textbook 1 and textbook 2 will be consulted often. Lecture notes will supplement the textbooks. While I highly encourage you to purchase the textbook, it is not required. All lecture notes and supplements will be posted online.

Course outcomes A student who successfully fulfills the course requirements will have demonstrated the ability to:

1. Describe and explain the terminology, representation and use of formal languages and grammars;
2. Describe and explain the terminology and techniques of lexical analysis, parsing, semantic actions and code generation;
3. Design and implement a compiler for a small language based on their knowledge of the previous two points.

more specifically, after taking this course, you will be able to:

- Explain the various passes of a compiler (scanners, parsers, semantic actions and code generation, register allocation and basic optimizations) and how they relate to the overall compilation process.
- Explain and implement the algorithms for each of these processes.
- Be able to implement each of these passes and integrate them into a full compiler.
- Explain program analysis techniques that are used for code optimization, such as data?ow analysis, def-use analysis, liveness analysis, etc.
- Describe basic code transformations and their application to program optimization.

Course assessment The achievement of course objectives will be assessed through a combination of tests (2 paper-based exams (mid-sem and finals)) and a series of programming assignments (5 to 6). The exams will test your understanding of the concepts covered in the class, while the programming assignments will test your ability to put those concepts into practice.

Course grading Grades will be assigned as follows:

Score	Assessment	Comments
65%	Exams	Mid-sem (25%) and End-sem (40%)
32%	Programming assignments (5-6)	Each assignment (except assignment 0) contributes equally from 6.4%-5.33%
8%	Class participation	Bonus

Note that the grading rubric adds up to 105%. This is because a part of class participation points (5) function as bonus that can raise your grade for noteworthy contributions during class or in online discussions (see below re: Course Discussion). Given your final numerical score, your course grade will be determined according to the following scale:

If your numerical score is at least:	Your course grade will be at least:
90	AA
80	AB
70	BB
60	BC
50	CC
45	CD
40	DD

Programming assignments We will have biweekly programming assignments. Importantly, these assignments build on top of previous assignments. The assignments may involve writing a substantial amount of code depending upon the language you choose for implementation (C instead of C++). Programming assignments must follow the below guidelines:

- All programs should compile and run correctly on lab machines in Room 24/Room 120. It is your responsibility to make sure that your code works in that environment.
- Unless otherwise specified, all assignments are due at 11:59 PM on the deadline date.

Programming assignments will be submitted via GitHub Classroom (https://classroom.github.com). As such, you are required to have a GitHub account. These can be obtained for free at https://github.com. *You must also send your GitHub username to the Instructor latest by Wednesday, January 8th 2020. You must provide the details in a Google Form, the link for which will be posted on Moodle and Piazza.*

Late submission policy except for medical and family emergencies (accompanied by verification), there will be no individual extensions granted for programming assignments. Late submissions will be scaled according to lateness, docking 10% from your score per day late, up to a maximum of 50%. Submissions more than 5 days late will be assigned a score of 0.

Exams we will have two paper-based exams. The exams are open book and open notes. You may bring the course textbook as well as any written/printed notes/programs from lectures or otherwise. *You may not use electronic devices in paper-based exams.*

Exam	Topic	Week
Midsem	Scanning, parsing, semantic routines, code generation	24/2/20 to 28/2/20
Endsem	Register allocation, instruction scheduling, peephole optimizations, loop optimizations, and program analysis	20/4/20 to 25/4/20

Exam topics and tentative dates The exact dates and the venue for the exams will be communicated by the academic office. If you need a change in either the test times or environment due to an excused University absence or an approved accommodation from Dean AP's office, it is your responsibility to contact the Professor two weeks prior to the exam so alternate arrangements can be made.

Course discussion this term we will be using Piazza for class discussion. If you have questions about the course or the project, we encourage you to post them on Piazza. It's a shared discussion forum, where your question can be answered by the instructors, TAs, or your fellow students! Find our class's Piazza page at: https://piazza.com/iitdh.ac.in/spring2020/cs406/

Students who are active participants on Piazza, asking (or answering!) questions are eligible for class participation bonus points.

Email Questions about course material or programming assignments should be posted to Piazza or raised during lecture or office hours. *The Professor will not answer programming questions via email.* This is to allow other students who might have similar questions to benefit from our answers. Of course, if you have questions of a personal or confidential nature, we welcome your email.

Course announcements Homework assignment links will be distributed via Moodle/Piazza announcements and grades will be posted on Moodle. Other course announcements, including changes in due dates, course topics, programming assignment details, etc., will be communicated in three ways:

1. Updates to the course webpage.
2. Announcement posts on Moodle and Piazza.
3. Email announcements.

Course topics below is the list of topics that will be covered in this course, and a rough estimate of how long we will spend on each.

Topic	Number of weeks	Reading
Structure of a compiler (introduction, overview)	1	Chapter 1
Scanning	1	Chaper 3
Parsing	1.5	Chapters 4 and 5
Semantic routines (symbol tables, AST)	1	Chapters 5 and 6
Code generation (3 address code, basic optimizations)	1	Chapters 2 and 8
Instruction scheduling and Register allocation	1	Chapter 8
Optimizations (Code motion, strength reduction)	1	Chapter 9
Control and Dataflow analysis	2.5	Chapters 8, 9 and lecture notes
Parallelism (instruction level parallelism, optimizing for parallelism)	1	Chapter 10, lecture notes

Academic honesty unless explicitly allowed, you are expected to complete all assignments by yourself or as a team when teams are allowed. However, you are allowed to discuss general issues with other students (programming techniques, clearing up confusion about requirements, etc.). You may discuss particular algorithmic issues on Piazza (but do not copy code!). *We will be using software designed to catch plagiarism in programming assignments, and all students found sharing solutions will be reported to the Dean of students.*

Punishments for academic dishonesty are severe, including receiving an FR in the course or being expelled from the University. By departmental rules, all instances of cheating will be reported to the Dean. On the first instance of cheating, students will receive a 0 on the assignment; the second instance of cheating will result in a failure of the course.