CS620T: Compilers - Principles and Implementation

Autumn 2026

Lectures CLT309 - Tuesdays (8:30AM - 09:55AM), Thursdays (10AM - 11:25AM)

Course web page https://hegden.github.io/cs620/

Instructor Nikhil Hegde (nikhilh 'at' iitdh 'dot' ac 'dot' in)

TAs Damarla Nithin (cs25mt017 'at' iitdh 'dot' ac 'dot' in)

Prerequisites A solid grounding in C/C++, A working knowledge of data structures. A grasp of the 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. Textbook 2 will be consulted as well. Lecture notes will supplement the textbooks. While I highly encourage you to purchase the textbook 1, it is not required. All lecture notes and supplements will be posted online.

Course outcomes A student who successfully fulfills this course (CS620T) requirements will have demonstrated the ability to:

1. Describe and explain the terminology of regular expressions and scanners, grammar and parsers, program representation;
2. Describe and explain the terminology and techniques of memory management, code generation, and optimization;
3. Design and implement analysis and transformation passes in LLVM.

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.
- Be able to identify each of these passes and modify them if needed in a production-grade compiler.
- Explain program analysis techniques that are used for code optimization, such as dataflow 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, take-home), short quizzes, group activity, and a series of individual programming assignments. Tests, quizzes, and class activities 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:

Weight	Assessment	Comments
65%	Exams	Mid-sem (25%) and End-sem (40%)
20%	Short Quiz	Multiple-choice based questions, roughly one per class. Best 20 taken.
8%	Assignments (2)	Written and/or Programming-based
7%	Open (source) problem	check LLVM repo issues

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 will be submitted via GitHub Classroom (https://classroom.github.com). So, you are required to have a GitHub account. These can be obtained for free at https://github.com.

Prerequisite Software To help you develop the programs on server environment, we will create an account for you and share the details of software installed. It is strongly advised that you have `ssh` and `Git` software available on your local (laptop) environment.

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 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.

Exam	Topic	Week
Midsem	Scanning, parsing, semantic routines, program representation, optimizations, analysis and transformations	23/9 - 24/9 (tentative)
Endsem	Register allocation, instruction scheduling, loop optimizations, polyhedral compilation, misc.topics	19/11 - 20/11 (tentative)

Exam topics and dates (Tentative) Note that this is a tentative assessment scheme. The academic office will communicate the exact scheme at a later date. If you need a change in either the test times or environment due to 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 mostly be using the `Discussion` feature of Google classroom for class discussion. If you have questions about the course or the project, we encourage you to post them on the discussion page in Google classroom. It's a shared discussion forum, where your question can be answered by the instructors, TAs, or your fellow students!

Students who are active participants, asking (or answering!) questions are eligible for class participation points that are accumulated monthly.

Email Questions about course material or programming assignments should be posted to the discussion page or raised during lecture or office hours. *The Professor or TAs 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 Course Webpage is the primary source of information for course-related material and important announcements. Homework assignment links will be distributed via `Discussion` page/email. Other course announcements, including changes in due dates, course topics, programming assignment details, etc., will be communicated in one or more of the following ways:

1. Updates to the course webpage.
2. Announcement posts on Google classroom.

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
Structure of a compiler (introduction, overview), scanner, parser, tool: flex,bison,ANTLR	1.5
Semantic routines, Program representation (AST, Intermediate Representation, Control Flow Graphs), SSA	1.5
Memory Management and Garbage Collection hline Local and Global Optimizations	1 1.5
Program Analysis and Transformation	1.5
Loop Optimizations and Polyhedral compilation	1.5
Instruction selection, Code Generation	2
Register allocation	1.5
Parallelism (instruction level parallelism, optimizing for parallelism)	1.5

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 the Discussion forum (but do not copy-paste your 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.*

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.