CS406 + CS316 : Compilers (theory + lab)

Spring 2021 (off-campus based session)

Lectures Tuesdays (10:30AM - 11:15AM), Wednesdays(8:30AM - 9:15AM), Thursdays(9:30AM - 10:15AM) (live, online sessions)

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

Instructor Nikhil Hegde (nikhilh@iitdh.ac.in)
Office Hours (doubt clearing sessions) : decided based on individual's / groups' availability online.

TAs Priya Hampannavar, Gayatri Rayar, and Shiwali Gupta
Office Hours (doubt clearing sessions) : decided based on individual's / groups' availability online.

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. 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 the course (CS406 and CS316) requirements will have demonstrated the ability to:

1. Describe and explain the terminology, representation and use of formal languages and grammars; (CS406)
2. Describe and explain the terminology and techniques of lexical analysis, parsing, semantic actions and code generation; (CS406)
3. Design and implement a compiler for a small language based on their knowledge of the previous two points (CS316).

more specifically, after taking this course (CS406 and CS316), 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. (CS306)
- 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, online, take-home), short quizzes, group activity, and a series of individual programming assignments (7 to 8). 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:

Course	Score	Assessment	Comments
CS406	30%	Exams	Mid-sem (15%) and End-sem (15%)
CS406	25%	Short Quiz	Multiple-choice based questions, one per class. Best 25 taken.
CS406	5%	Class participation (live and discussion forum)	Asking a technical question (1), Answering a technical question (4)
CS406	40%	Assignments (4)	Written and Programming-based
CS316	100%	Programming assignments (7)	The weight of an assignment depends on its complexity

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 (e.g. C instead of C++). Programming assignments must follow the below guidelines:

- All programs should compile and run correctly. It is your responsibility to make sure that your code works in the execution environment made available to you (details about the execution environment will be communicated later).
- Unless otherwise specified, all assignments are due at 11:59 PM on the deadline date.
- You may use C/C++/Java for implementing the programming assignments.

Programming assignments will be submitted via GitHub Classroom (https://classroom.github.com). Whether you are registering for CS316 only or not, you are required to have a GitHub account. These can be obtained for free at https://github.com. *You must send your GitHub username to the Instructor latest by Friday, January 8th, 2020. You must provide the details in a Google Form, the link for which will be emailed to you.*

Prerequisite Software To develop the programs on your local environment (laptop / desktop), you must have a compiler toolchain (GCC, ICC, LLVM, MS Visual Studio etc.). In addition, you must have Flex, Bison, or ANTLR (if you are developing Java programs). It is also strongly advised that you have `Git` software available on your local 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 we will have two exams for CS406. We will not have mid-sem and end-sem exams for CS316. The exams are open book, open Internet, 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, code generation	TBD
Endsem	Register allocation, instruction scheduling, peephole optimizations, loop optimizations, and program analysis	TBD

Exam topics and tentative dates 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 GitHub Teams for class discussion. If you have questions about the course or the project, we encourage you to post them on the discussion page in GitHub Teams. 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 GitHub Teams.
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 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.