CS 5470: Compiler Principles and Techniques

Administrative Details and Syllabus
Spring 2018

Important Information

Class Website: Canvas (available through CIS)
Lectures: Tuesdays and Thursdays 2-3:20p in 2250 WEB
Instructor: D. Erin Parker, 3144 MEB

Important Dates
Midterm exam: Thursday, March 8 (during class)
Final exam: Wednesday, May 2 (1-3p)

Final course grade: Project 50%, Final exam 25%, Midterm exam 15%, Problem sets 10%
Failing exam average → failing course grade

Prerequisites: CS 3100: Models of Computation, CS 4400: Computer Systems

Getting help: The class website has details on how to see TA(s) and the instructor outside of class, as well as how to post questions to the class forum and email to the course staff.

Course Information

Catalog Description. Lexical analysis, top-down and bottom-up parsing, symbol tables, internal forms and intermediate languages, runtime environments, code generation, code optimization, semantic specifications, error detection and recovery. Use of software tools for lexical analysis and parsing.

Better Description. Compilers translate “programs” written in one language (e.g., C programs, Java programs, \LaTeX documents, or hardware descriptions) into “programs” written in another language (e.g., machine code, Java byte code, PDF documents, or circuit layouts). This course considers the principles that underlie all compilers and focuses on the problem of translating programs written in a conventional higher-level language into semantically equivalent programs written in assembly language.

In this course, students learn how modern programming languages are implemented, how compilers interact with operating systems and machine architectures, and how to use compiler construction tools.
The prerequisites for this course are CS 3100 (Models of Computation) and CS 4400 (Computer Systems). Students who take CS 5470 should already be comfortable with the following concepts:

- Formal notations for describing languages, including regular expressions and context-free grammars.
- Common algorithms and data structures, including tree, set, and graph representations and the algorithms that manipulate them.
- The basics of computer architecture, including how machines actually work and how to program in assembly language.
- The basics of systems programming, including how a computer’s memory is organized at runtime.
- Reading, writing, and maintaining moderately large programs.

CS 5470 is a “capstone” course, bringing together what students have learned in core undergraduate courses. The primary goal of this course is to give students a better understanding of how material learned in software courses (algorithms and data structures, programming language semantics, and software engineering), computer architecture courses (assembly language and computer organization), theory courses (formal language descriptions), and system courses (interaction of programs with system services) fit together.

This is a 4.0 credit hour course that meets for lecture three hours a week. The fourth hour represents time spent on the substantial course project — building a compiler. The compiler is built by students in a number of stages, with “starter” code (written in Java) given for each stage. The source language of the compiler is a subset of Java, called MiniJava. The target assembly language of the compiler is MIPS.

Course Materials

Website. The class website is a Canvas course available through CIS. It is always under development, with updates to the class schedule, lecture notes, assignment specifications, and more, occurring regularly. It is critical that students become familiar with the class website right away and plan to visit it several times a week, at a minimum.

Textbook. Regular reading assignments are made from Modern Compiler Implementation in Java by Andrew W. Appel (2nd edition, 2002, ISBN: 052182060X). Students are highly encouraged to purchase used copies. This is an older textbook. However, the fundamental techniques of compiling explained therein, and especially the course project presented, are still relevant.

Lecture notes. The instructor often makes use of slides, sample problems, and other materials during lecture. These items are posted on the class website following the lecture; however, such posted items may not represent completely the material covered in class. Students who must miss class are strongly encouraged to check with a classmate.

Class tools and lab accounts. The CADE Lab (L224/226 WEB) has all of the software needed for this class. You are not required to purchase, install, or maintain any software to take this class — simply use the CADE Lab. If you do not have a CADE Lab account, or if you need your password reset, go to https://webhandin.eng.utah.edu/cade.
Students may use their own computers for completing assignments; however, broken tools or computers, or network connectivity issues are not sufficient basis for an assignment deadline extension. Plan ahead and use the lab computers if your own computer is not working.

Student Evaluation

Project. The programming project of CS 5470 is to construct a working compiler from a language specification. The appendix of the course text describes MiniJava, the language whose programs the compiler translates to MIPS assembly code. The language for implementing the compiler is Java. In implementing the compiler, students make use of software tools for constructing the scanner and parser. The project consists of six cumulative stages (scanner, parser, checker, IR-tree generator, assembly-code generator, and register allocation), with each stage depending of the successful completion of the previous stage.

It is the students’ responsibility to ensure the successful and timely submission of each assignment — start early and follow the instructions carefully. Corrupted or missing files are not grounds for extensions — double-check your submissions and save a digital copy of all of your work in your CADE account. For each hour a submission of the compiler project is late (except in the case of a documented medical emergency), one point will be deducted from the total score.

Problem sets. Problem sets, in the form of Canvas quizzes, are assigned approximately every two weeks throughout the semester. No problem set submissions are accepted late, except in the case of a documented medical emergency. The lowest problem set score will be dropped.

Midterm and final exam. The midterm exam is to be given during class on Thursday, March 8. The final exam is cumulative and will take place Wednesday, May 2, 1-3p in 2250 WEB. No exam may be taken at a different time for any reason other than a documented medical emergency.

Final course grade. If the average score for the midterm and final exam is 65% or lower, the final course grade is no higher than a D+. Otherwise, the final course grade is based on Project 50% (breakdown into stages TBA), Final exam 25%, Midterm exam 15%, Problem sets 10%.

Regrades. Students who wish to appeal a score on an assignment or a test must do so within one week of receiving the score and use the Regrade request form posted on the class website.

Letter grades. The following table is used to associate numerical scores with the corresponding letter grade:

<table>
<thead>
<tr>
<th>分数范围</th>
<th>字母等级</th>
</tr>
</thead>
<tbody>
<tr>
<td>93 \leq X \leq 100</td>
<td>A</td>
</tr>
<tr>
<td>90 \leq X &lt; 93</td>
<td>A-</td>
</tr>
<tr>
<td>87 \leq X &lt; 90</td>
<td>B+</td>
</tr>
<tr>
<td>83 \leq X &lt; 87</td>
<td>B</td>
</tr>
<tr>
<td>80 \leq X &lt; 83</td>
<td>B-</td>
</tr>
<tr>
<td>77 \leq X &lt; 80</td>
<td>C+</td>
</tr>
<tr>
<td>73 \leq X &lt; 77</td>
<td>C</td>
</tr>
<tr>
<td>70 \leq X &lt; 73</td>
<td>C-</td>
</tr>
<tr>
<td>67 \leq X &lt; 70</td>
<td>D+</td>
</tr>
<tr>
<td>63 \leq X &lt; 67</td>
<td>D</td>
</tr>
<tr>
<td>60 \leq X &lt; 63</td>
<td>D-</td>
</tr>
<tr>
<td>X \leq 60</td>
<td>E</td>
</tr>
</tbody>
</table>

Getting Help

To get help understanding course material, students may see the Teaching Assistant(s) during TA Consulting Hours, see the instructor during Office Hours, post a question to the Discussions, or send email to the course staff at teach-cs5470@lists.utah.edu. See the How to get help in CS 5470 module on the class website for details.
Course Guidelines

Working in Groups. Students may work alone or in groups of two to complete the programming project. The groups (or lack of) must remain the same throughout the semester. Students working alone may discuss only hi-level solution strategies with classmates. The same is true for students working in groups, when in discussions with classmates other than their partners. Students working in groups must work cooperatively during each stage of the programming project and turn in a single project solution written jointly by both students. (See the CS 5470 Academic Misconduct Policy for more information.)

Student code. All students are expected to maintain professional behavior, according to the University of Utah Student Code at www.regulations.utah.edu/academics/guides/students/studentRights.html. Students should read the Code carefully and know that they are responsible for the content. According to Faculty Rules and Regulations, it is the faculty responsibility to enforce responsible classroom behaviors, beginning with verbal warnings and progressing to dismissal from class and a failing grade. Students have the right to appeal such action to the Student Behavior Committee.

Laptop policy. Students are expected to engage with the instructor and classmates during class meetings. Students are permitted to use a laptop or mobile device to take notes. Use of a laptop or mobile device for any other purpose is not permitted, and students who do so will be asked to leave the classroom.

Syllabus

The following are the key topics planned for study, the approximate number of lectures devoted to each, and the corresponding chapters in the course text.

Introduction – 2 lectures – Chapter 1, Appendix
  Administrative details, Overview of the translation process, The MiniJava language

Syntactic Analysis – 6 lectures – Chapters 2-4
  Lexical analysis, Using scanner and parser generators, Context-free grammars and parsing,
  Top-down parsing: Recursive-descent (or predictive), Bottom-up parsing: LR,
  ASTs: Abstract syntax (or parse) trees

Semantic Analysis – 2 lectures – Chapter 5
  Symbol tables, Type checking

Run-time Organization – 3 lectures – Chapter 6
  Activation records and run-time stacks, Functions: linking and parameter passing

Code Generation – 6 lectures – Chapters 7-9
  Translating ASTs into intermediate representation trees, Basic blocks and traces,
  Instruction selection

Liveness Analysis and Register Allocation – 3 lectures – Chapters 10-11

Advanced Topics – 5 lectures – Chapters 17-18, 21
  Dataflow Analysis, Loop Optimizations, The Memory Hierarchy