

CS 5470: Compiler Principles and Techniques

Administrative Details and Syllabus Spring 2010

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, Postscript 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 will 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 run time.
- Reading, writing, and maintaining moderately large Java 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.

Instructor. D. Erin Parker. *Office:* 3190J MEB. *Email:* parker@cs.utah.edu. *Office Hours:* See <http://www.cs.utah.edu/~parker/hours.html>.

Class Meetings. MWF 10:45-11:35a in 120 WEB.

Communication. The class web page is <http://www.eng.utah.edu/~cs5470/>. It will contain the lecture schedule, problem set assignments, specifications for the compiler project, and more.

For questions outside of class and office hours, students are encouraged to use email.

The course staff will use the class mailing list (cs5470@eng.utah.edu) to send urgent messages to everyone in the class, such as corrections to written problem sets or changes in due dates. Students are not able to send mail to the class list. Students must subscribe to the class mailing list by visiting <https://sympa.eng.utah.edu/sympa/info/cs5470>.

Students who would like to ask a question of the course staff should use the staff mailing list (teach-cs5470@eng.utah.edu). The instructor or teaching assistant will respond to each question directly.

Text. The required course text is *Modern Compiler Implementation in Java* by Andrew W. Appel, 2nd edition, Cambridge University Press, 2002 (ISBN: 0-521-82060-X). An eBook version of the text is available at <http://www.cambridge.org/us/catalogue/catalogue.asp?isbn=0511039301>.

Lectures. This class will meet for lecture three times a week for fifty minutes. The instructor will make use of slides during lecture, and the slides covered during each lecture will be posted on the class web page following the lecture. Students are encouraged to take notes in class and should not expect to rely solely on posted slides to recall the material covered in each lecture.

Reading. The lecture schedule is posted on the class web page. Following each class meeting, the schedule will be updated to reflect the material actually covered by the lecture that day and to indicate the reading assignment for the next lecture. By the end of the semester, the lecture schedule will be a record of all the material covered in this class.

Assignments and Grading. Assignments consist of a number of problem sets and a programming project in six stages. Problem set solutions must be handed in at the start of class (10:45a) on the due date. *Late problem set solutions will not be accepted (except for medical reasons)*. Project solutions must be electronically submitted by 11:59p on the due date. *For each hour a programming project solution is late (except for medical reasons), one point will be deducted from the total score*. The programming project is to construct a working compiler from a language specification. The appendix of the course text describes MiniJava, the language whose programs the compiler will translate to MIPS assembly code. The language for implementing the compiler is Java. In implementing the compiler, students will 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 allocator), with each stage depending on the successful completion of the previous stage. Solutions for each project stage will not be posted, and students may not consult any other solution, such as those from years past or those from classmates. *Students who fall behind early in the programming project will have difficulty passing this class*. More detailed information on the programming project will follow.

One in-class midterm exam is scheduled for Wednesday, March 10. The final exam for this class will be on Tuesday, May 4, 10:30a–12:30p.

A student's course grade will be based on the programming project (55%), problem sets (15%), the midterm exam (10%), and the final exam (20%). Some consideration will be given to class participation. *Students who do not pass exams may not pass the course*.

The following scale is used to assign letter grades.

100-93	A	89-87	B+	79-77	C+	69-67	D+	59-0	E
92-90	A-	86-83	B	76-73	C	66-63	D		
		82-80	B-	72-70	C-	62-60	D-		

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.

Problem sets must be completed individually.

Students with Disabilities. The University of Utah seeks to provide equal access to its programs, services, and activities for people with disabilities. If you need accommodations in this class, reasonable prior notice needs to be given to the Center for Disability Services, 162 Olpin Union Building, 581-5020 (V/TDD). CDS will work with you and the instructor to make arrangements for accommodations.

College of Engineering Guidelines. For information on withdrawing from courses, appealing grades, and more, see <http://www.coe.utah.edu/SemesterGuidelines.pdf>.

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 (3 lectures) Chapter 1, Appendix

Administrative details

Overview of the translation process

The MiniJava language

Syntactic Analysis (5 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 (3 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 (10 lectures) Chapters 7-9

Translating ASTs into intermediate representation trees

Basic blocks and traces

Instruction selection

Liveness Analysis and Register Allocation (5 lectures) Chapters 10-11

Advanced Topics (9 lectures) Chapters 17-18, 21

Dataflow Analysis

Loop Optimizations

The Memory Hierarchy

TBD