Computer Engineering 175
Formal Language Theory and Compiler Construction

Winter 2012
Mondays, Wednesdays, and Fridays
1:00 pm – 2:05 pm

Instructor

Instructor:	Darren Atkinson
E-mail:	datkinson@scu.edu
Office hours:	Tuesdays 1:15–2:15 pm and Wednesdays 9:30–10:30 am
Office:	EC 245
Website:	http://www.cse.scu.edu/~atkinson/teaching/wi12/175/

Teaching Assistant

Teaching assistant:	Yiwen Zhang
E-mail:	yiwenzhang1@gmail.com
Lab hours:	Mondays and Tuesdays, 2:15–5:00 pm

Textbooks

Required: Aho, Lam, Sethi, and Ullman, *Compilers: Principles, Techniques, and Tools*, 2007

Grading

Midterm exam:	20% (2/15)
Final exam:	40% (3/23)
Project:	40%

Course Outline

1. Introduction; lexical analysis; syntax analysis
2. Ambiguous grammars; top-down parsing of expressions . . . . . Phase I due on 1/18
3. Parsing of statements; syntax-directed translation; semantic checking . . . . . Phase II due on 1/27
4. Scoping and symbol table management; type checking
5. Type systems and type equivalence; storage allocation methods . . . . . Phase III due on 2/8
6. Stack frame manipulation . . . . . Midterm exam on 2/15
7. Intel storage allocation; code generation for simple expressions . . . . . Phase IV due on 2/22
8. Code generation for complex expressions and statements . . . . . Phase V due on 3/2
9. Regular languages and automata; grammars and context-free languages
10. First and follow sets; table-driven parsers; decidability . . . . . Phase VI due on 3/16
11. Final exam on 3/23

Learning Outcomes

Students will . . .

1. Build a compiler for a nontrivial programming language.
2. Describe the phases of compilation.
3. Specify regular expressions for matching tokens in a language.
4. Show the equivalence between regular expressions, NFAs, and DFAs.
5. Specify and disambiguate context-free grammars.
6. Specify a type system for a language including type equivalence, and use it to correctly type check expressions in a language.
7. Apply fundamentals of storage allocation strategies toward run-time management of data.
8. Generate correct assembly code for simple expressions and statements in a programming language.
9. Demonstrate programming proficiency in an object-oriented language (e.g., C++), including an understanding of fundamental object-oriented concepts such as encapsulation, abstraction, inheritance, and polymorphism.

Policies

Course Policies

You must be registered for both the lecture and the lab. Since the project grade is more than 20% (1 unit out of 5 units) of your course grade, you simply receive the same grade for both the lecture and lab in this course.

Disability Accommodation Policy

To request academic accommodations for a disability, students must be registered with Disabilities Resources located in Benson, room 216. If you would like to register with Disabilities Resources, please visit their office in Benson 216 or call (408) 554-4109. You will need to register and provide professional documentation of a disability prior to receiving academic accommodations.

Academic Integrity Policy

The University is committed to academic excellence and integrity. Students are expected to do their own work and to cite any sources they use. A student who is guilty of a dishonest act in an examination, paper, or other work required for a course, or who assists others in such an act, may, at the discretion of the instructor, receive a grade of F for the course.

In addition, a student found guilty of a dishonest act may be subject to sanctions up to and including dismissal from the University as a result of the student judicial process as described in the *Community Handbook*.

A student who violates copyright laws, including those covering the copying of software programs, or who knowingly alters official academic records from this or any other institution is subject to similar disciplinary action.