

**CS 5641: Compiler Design (4)****Catalog Description:**

A selection from the following topics: finite-state grammars, lexical analysis, and implementation of symbol tables. Context-free languages and parsing techniques. Syntax-directed translation. Run-time storage allocation. Intermediate languages. Code generation methods. Local and global optimization techniques.

**Textbook:** Alfred Aho, [Monica Lam](#), Ravi Sethi, Jeffrey Ullman, *Compilers: Principles, Techniques and Tools* (2<sup>nd</sup> edition), Addison-Wesley, 2006.

**References:****Course Goals:**

This course presents a comprehensive introduction to the process of creating a compiler for a high-level programming language. One of the goals of this course is to give students the theoretical background and hands-on implementation experience necessary to understand and employ tools used in compiler design such as flex and bison. Students are expected to understand and be able to implement both scanner and parser tools so as to better understand how to effectively use tools that automatically generate scanners and parsers. A second goal is to give students significant understanding of and experience with semantic analysis techniques such as abstract syntax tree generation, type checking, etc., in order to produce intermediate code. A third goal is to give students the requisite understanding of and some experience with code optimization and code generation. A fourth goal is to give students significant understanding of a variety of programming language capabilities to increase their proficiency with these programming languages and to allow them to better exploit the capabilities of these languages. A fifth goal is for students to increase their overall programming proficiency and their ability to work in teams as a result of the implementation of their team compiler project. The overall goal of the course is to give students significant knowledge and experience with compilers and programming languages, both in terms of being able to develop them and in terms of being better able to employ them as part of developing a software system.

**Prerequisites by Course & Topic**

[CS 2511: Software Analysis and Design](#) – software engineering, object-oriented design, software testing and debugging, team implementation

[CS 3512: Computer Science Theory – proof techniques, formal languages, mathematical logic, complexity measures of algorithms](#)

CS 2521: Computer Organization and Architecture – machine language, assembly language, basics of computer organization, data structure and representation (machine level)

**Major Topics Covered in the Course**

- Lexical Analysis
- Parsing
- Semantic Analysis
- Programming Language Components
- Code Generation and Optimization

**Class/Laboratory Schedule:** Lecture: 3 hours per week, Laboratory: 1

**Course Outcomes**

1. Proficiency with lexical analysis methods and tools.
2. Proficiency with parsing methods and tools.
3. Proficiency with basic issues of semantic analysis.
4. Familiarity with methods for code generation.
5. Further proficiency with software design in high-level languages.

rmaclin 5/16/11 1:38 PM

**Deleted:** June 2007

rmaclin 5/16/11 1:39 PM

**Formatted:** Superscript

rmaclin 5/16/11 1:38 PM

**Deleted:** 1986

rmaclin 5/16/11 1:32 PM

**Deleted:** yacc

rmaclin 5/16/11 1:36 PM

**Deleted:** CS 3512: Computer Science Theory – proof techniques, formal languages, mathematical logic, complexity measures of algorithms,

rmaclin 5/16/11 1:36 PM

**Deleted:** (via 3512 prerequisite 2511)

rmaclin 5/16/11 1:36 PM

**Deleted:**

rmaclin 5/16/11 1:36 PM

**Deleted:** a. Understand finite state machines, regular expressions, and their relationship. -  
a. Understand context free grammars, languages, and derivation. -  
a. Understand basic issues of syntax-directed translation. -  
a. Familiarity with the basic issues of memory management, stack and heap maintenance. -  
a. Design and develop a small compiler as part of a coding team. - ... [1]

rmaclin 5/16/11 1:37 PM

**Deleted:** a. Understand context free grammars, and derivation. -  
a. Understand basic issues of syntax-directed translation. -  
a. Familiarity with the basic issues of memory management, stack and heap maintenance. -  
a. Design and develop a small compiler as part of a coding team. - ... [2]

rmaclin 5/16/11 1:37 PM

**Deleted:** a. Understand basic issues of syntax-translation. -  
a. Familiarity with the basic issues of memory management, stack and heap maintenance. -  
a. Design and develop a small compiler as part of a coding team. - ... [3]

rmaclin 5/16/11 1:37 PM

**Deleted:** a. Familiarity with the basic issues of management, stack and heap maintenance. -  
a. Design and develop a small compiler as part of a coding team. - ... [4]

rmaclin 5/16/11 1:42 PM

**Deleted:** a. Design and develop a small of a coding team. - ... [5]

rmaclin 5/16/11 1:38 PM

Deleted: June 2007

**Relationship to Program Outcomes**

This course contributes to the following Program Outcomes:

2. Students can design, develop, and analyze significant software systems. Students are required to build a small compiler or interpreter as their semester project, giving them significant experience with development of a large piece of code and make them aware of the effects of design decisions.
3. Students understand the fundamentals of computer systems organization. In the latter sections of the course, both in code generation and optimization there is significant discussion of how these processes are affected by the type of machine the code is being generated for and how optimizations affect efficiency.
5. Students understand the application of programming languages in computer systems. One large focus of the course is to give students insight into how programming languages work with regards to how they are compiled and interpreted, arming students with significant knowledge of how the systems they will develop code within work.
6. Students can apply computer science principles and practices to a variety of problems. Students learn the basics of lexical analysis and parsing in addition to other aspects of compilers and interpreters which they practice on a significant project and which can also be applied to a variety of problems.

**Assessment of Performance Criteria**

CS 5641 is expected to assess the following program criterion:

6. Students can apply computer science principles and practices to a variety of problems.
  - a. Demonstrate knowledge of the core methods for that area
  - b. Demonstrate understanding of key issues of that area
  - c. Demonstrate proficiency in applying knowledge from that area to relevant problem

**Assessment Plan for Course:**

This course is assessed every third year by the instructor and a course assessment document covering all of the course outcomes and their effect on the program outcomes is prepared.



**Coordinator/Prepared by:** R. Maclin

rmaclin 5/16/11 1:53 PM

Formatted: Indent: Left: 0.44"

rmaclin 5/16/11 1:37 PM

Deleted: Estimate CSAB Category Content - CORE ... [6]

**Page 106: [1] Deleted** **rmaclin** **5/16/11 1:36 PM**

- a. Understand finite state machines, regular expressions, and their relationship.
- b. Understand automatic translation of regular expressions to FSMs.
- c. Proficiency with a scanner creation tool such as lex or flex.

**Page 106: [2] Deleted** **rmaclin** **5/16/11 1:37 PM**

- a. Understand context free grammars, languages, and derivation.
- b. Familiarity with issues in language precedence and associativity and grammars.
- c. Understand basic top-down parsing methods including recursive descent, predictive parsing, and LL(1) methods..
- d. Understand basic bottom-up parsing methods including SLR, LR(1) parsing, and LALR parsing.
- e. Proficiency with a parser creation tool such as yacc or bison.

**Page 106: [3] Deleted** **rmaclin** **5/16/11 1:37 PM**

- a. Understand basic issues of syntax-directed translation.
- b. Familiarity with construction and use of parse trees.
- c. Proficiency with constructing and using abstract syntax trees.
- d. Familiarity with issues of scoping and symbol table implementation.
- e. Familiarity with issues of parameter passing.
- f. Proficiency with type checking methods and implementation.

**Page 106: [4] Deleted** **rmaclin** **5/16/11 1:37 PM**

- a. Familiarity with the basic issues of memory management, stack and heap maintenance.
  - b. Familiarity with intermediate code generation.
  - c. Familiarity with the basic issues of code optimization.
  - d. Familiarity with the translation of intermediate code to machine code.
5. Ability to work successfully on a team.
- a. Meet with team members to design project goals, delegate responsibilities, and complete project.
  - b. Communicate with team members.
  - c. Integrate code generated by different team members.

6

**Page 106: [5] Deleted** **rmaclin** **5/16/11 1:42 PM**

- a. Design and develop a small compiler as part of a coding team.
- b. Develop a scanner using a lexical analysis tool.
- c. Develop a parser using a parser tool.

**Page 107: [6] Deleted** **rmaclin** **5/16/11 1:37 PM**

**Estimate CSAB Category Content**

	CORE	ADVANCED		CORE	ADVANCED
Data Structures			Computer Organization and Architecture		
Algorithms			Concept of Programming Languages		3
Software Design		1			