

**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, Ravi Sethi, Jeffrey Ullman, *Compilers: Principles, Techniques and Tools*, Addison-Wesley, 1986.

**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 lex and yacc. 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 3512: Computer Science Theory – proof techniques, formal languages, mathematical logic, complexity measures of algorithms, (via 3512 prerequisite 2511) software engineering, object-oriented design, software testing and debugging, team implementation

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

### **Relationship to Program Outcomes**

CS 5641 requires a student to have completed discrete math, systems analysis and design, and computer organization and architecture before taking the course. This course contributes to meeting the following program outcomes:

*2. Students can design, develop, and analyze significant software systems.*

Students greatly increase their knowledge of software design by implementing a large scale project that requires significant application of design and problem analysis knowledge. Students also gain a much better understanding of compilers and how these tools affect the design of software. Course outcomes 1-6 map to this program outcome.

*3. Students understand the fundamentals of computer organization and architecture, data structures and related algorithms, and programming languages..*

Students learn algorithms associated with scanning and parsing and utilize supporting data structures to implement these algorithms. Students significantly add to their understanding of programming languages by learning the details of language implementation. Students also gain further experience with the interaction between high-level programming languages and machine language. Course outcomes 1-6 map to this program outcome.

*4. Students can apply computer science principles and practices to a variety of problems..*

Students have the opportunity to design and build a compiler. Students also learn to use standard scanning and parsing tools which can be applied in many problem-solving contexts. Course outcomes 1-6 map to this program outcome.

### **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.

**Estimate CSAB Category Content**

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

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