

## COURSE DESCRIPTION

<b>Department and Course Number</b>	CMPS 451	<b>Course Coordinator</b>	Mark Radle
<b>Course Title</b>	Compiler Construction	<b>Total Credits</b>	3
<b>URL</b>	<a href="http://www.cacs.louisiana.edu/~mgr/451">www.cacs.louisiana.edu/~mgr/451</a>		

### **Current Bulletin Description:**

Compiler Construction. Introductions to compilers and language translation. Aspects of lexical, syntactic and semantic analysis including language theory and implementation. Use of finite state machines, regular expressions, top-down, bottom-up parsing techniques. Code generation and optimization, subroutine calls, symbol table management, LL and LR parser generators. Prereq: CMPS 450 with a grade of C.

### **Textbook:**

*Compilers: Principles, Techniques, and Tools*, Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman, Addison Wesley, 1986.

### **References:**

### **Course Goals:**

- To provide students with an understanding of the major phases of a compiler.
- To introduce students to the theory behind the various phases, including regular expressions, context-free grammars, and finite state automata.
- To provide students with an understanding of the design and implementation of a compiler.
- To have the students build a compiler, through type checking and intermediate code generation, for a small language.
- To provide students with an opportunity to work in a group on a large project.

### **Course Outcomes:**

- Students will have experience using current compiler generation tools.
- Students will be familiar with the different phases of compilation.
- Students will have experience defining and specifying the semantic rules of a programming language

### **Prerequisites by Topic:**

- In-depth knowledge of at least one structured programming language.
- Strong background in algorithms, data structures, and abstract data types, including stacks, binary trees, graphs.
- Understanding of grammar theories.
- Understanding of data types and control structures, their design and implementation.
- Understanding of the design and implementation of subprograms, parameter passing mechanisms, scope.

### Major Topics Covered in the Course:

Overview of compilation and the phases of a compiler	(1 hour)
Tools to aid in the automatic construction of compilers Lex, Yacc	(0.5 hour)
Lexical analysis Tokens, regular expressions, finite state automata (NFA, DFA) Algorithm converting from regular expression to NFA, algorithm converting from NFA to DFA Implementation of lexical analyzer from DFA	(10 hours)
Syntax analysis Context-free grammars, ambiguity, specifying operator precedence and associativity Parsing—top-down parsing, recursive-descent parsing, predictive parsing Left recursion elimination, left factoring	(9 hours)
Symbol table construction and issues Organization, operations Issues such as scope and overloading and their effect on symbol table design, implementation, and operations	(3 hours)
Syntax-directed translation Syntax-directed definitions, translation schemes, synthesized and inherited attributes, propagation of attribute values through syntax tree	(3 hours)
Construction of abstract syntax tree representation of program Binary tree representation of expressions and statements, construction of binary tree representation using a stack	(3 hours)
Semantic analysis, type checking Rules that cannot be described using a context-free grammar, Type checking rules for expressions and statements and issues such as type equivalence, overloading, coercion	(4.5 hours)
Intermediate code generation Three-address code for expressions and statements (including assignments, conditionals, loops, procedure calls), generation of temporary variables and tables	(3 hours)
Correctness Complexity of large projects, completeness of language implementation, evolution, design choices and tradeoffs, adherence to language standards.	( 2 hours )
Tests	(2 hours)
Final Exam	(2 hours)

### Laboratory projects (specify number of weeks on each) :

Students complete a compiler for a subset of Ada in phases (14 weeks total):

- Lexical analyzer.
- Syntax analyzer.
- Intermediate code generation.

Currently, the Purdue Compiler Construction Tool Set is used for the compiler generation.

### **Oral and Written Communications**

Every student is required to submit at least   1   written reports (not including exams, tests, quizzes, or commented programs) of typically   4   pages and to make        oral presentations of typically        minutes duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

### **Social and Ethical Issues**

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

N/A.

### **Theoretical Content**

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

N.A.

### **Problem Analysis**

Please describe the analysis experiences common to all course sections.

### **Solution Design**

Please describe the design experiences common to all course sections.