COURSE DESCRIPTION

Dept., Number	CSci 525	Course Title	Compiler Construction
Semester hours	3	Course Coordinator	Stephen V. Rice, Assistant Professor

Current Catalog Description

Introduction to techniques used in current compilers for computer languages; the syntactic specification of programming languages and an introduction to syntax-directed compiling.

Textbook

Kenneth C. Louden. *Compiler Construction: Principles and Practice*, PWS Publishing Company, 1997.

References

Course Outcomes

The primary goal of this course is to introduce the techniques used in programming language compilation. A secondary goal is to improve the students' knowledge and skill in computer programming by developing a large computer program (more than 2,000 lines of source code). After successfully completing this course, students are able to:

1. represent language tokens using regular expressions and finite automata
2. write a lexical analyzer for a language and generate a lexical analyzer using Flex
3. develop a top-down recursive-descent parser, and generate a bottom-up parser using Yacc, which parse syntax expressed by a context-free grammar and produce a syntax-tree representation of the input
4. implement type checking for a block-structured language using a symbol table
5. generate target code for a stack-based runtime environment
6. understand how source programs are mapped to target platforms via the compilation process, and write better source programs based on this understanding

Relationship between Course Outcomes and Program Outcomes

The course outcomes contribute to the program outcomes as follows: (1) to (j), (2) to (i), (3) to (i) and (j), (4) and (5) to (i), (6) to (c).

Prerequisites by Topic

CSci 311, Models of Computation, *or* CSci 450, Organization of Programming Languages

Major Topics Covered in the Course

- Scanning / lexical analysis: hand-coded scanner; generated scanner using Flex; use of regular expressions and finite automata
- Parsing / syntax analysis: hand-coded recursive-descent parser; generated parser using Yacc; use of context-free grammars and syntax trees
- Semantic analysis: symbol table; type checking
- Code generation: instruction generation and memory layout; stack-based runtime environment

Assessment Plan for the Course

This is an elective course offered approximately every two years. An offering typically has four examinations and a series of programming assignments in which the student incrementally develops a compiler for a subset of the C programming language. The examinations and assignments are designed to assess course outcomes (1) to (6).

How Data in the Course are Used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)

Estimate Curriculum Category Content (Semester hours)

Area	Core	Advanced	Area	Core	Advanced
Algorithms			Software design		1
Data structures			Concepts of programming languages		2