

COURSE DESCRIPTION

Department and Course Number: CSCI 525

Course Title: Compiler Construction

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

Total Credits: 3 hours

Coordinator: P. Tobin Maginnis, Associate Professor of Computer and Information Science.

Textbook: Holub, *Compiler Design in C*, Prentice-Hall, 1994.

Other required materials: Study guides and miscellaneous class handouts.

References: None.

Course Goals: Juniors, seniors, and graduate students are provided with the conceptual tools to analyze the compilation process, understand the design principles and tradeoffs in compiler construction, and use compiler generation tools for custom language construction.

Prerequisites by Topic:

1. Programming expertise in a high-level language such as Java, C, or Pascal (CSCI 211).
2. Familiarity with UNIX or Windows.
3. Preferred to have previous study of programming language organization (CSCI 450).

Major Topics Covered in the Course:

1. Introduction and overview (1 hour).
2. Formal grammars and expression analysis (7 hours).
3. A simple expression analyzer implementation (7 hours).
4. Description of a compiler (7 hours).
5. Runtime environments (5 hours).
6. Intermediate code generation (5 hours).
7. Code generation (3 hours).
8. Code optimization (2 hours).
9. Tests (5 hours).

Operating Systems and Languages: Unix, Linux, C

Laboratory projects:

There is a semester-long project to develop a compiler for the real-time language SKED.

Estimate of ABET/CAC Category Content:

	CORE	ADVANCED		CORE	ADVANCED
Data Structures	_____	_____ 1 _____	Computer Organization and Architecture	_____	_____ 1 _____
Algorithms	_____	_____ 1 _____	Concepts of Programming Languages	_____	_____
Software Design	_____	_____		_____	_____

Oral and Written Communications:

Not a significant focus of this course.

Social and Ethical Issues:

Not a significant focus of this course.

Theoretical Content (Foundations):

1. Formal languages (10 hours)
2. State machines (5 hours)

Problem Analysis:

Application of formal languages to implement state machines translate statements and expressions into executable code.

Solution Design:

1. Decide how to best define language syntax with formal rule set.
2. Construct a formal grammar that fits the language.
3. Design and implement run-time environment and insert into formal grammar.