COMPUTER SCIENCE 481
COMPILERS & COMPILER WRITING

I. Introduction

A. Catalog Description

The study of formal languages and automata theory and their application to the process of translating a source program written in a high-level computer language (source language) to an intermediate language. The study of the process and techniques of taking an intermediate language and employing syntax-directed translation together with optimization to produce an efficient low-level language program equivalent to the source program. This course is based in part on PL, Programming Languages, of ACM 91. It gives a formal presentation of programming language translation and compiler writing. The emphasis is on both the theoretical and some of the practical problems posed in implementing a compiler.

B. Learning Objectives

By the end of this course the student will be able to construct a compiler or interpreter for a subset of the C++ language or some other significant programming language.

C. Prerequisites

CSci 281 and Math 211 or CSci 370 (CSci 370 may be taken concurrently). A grade of C- or better is required in the prerequisite courses.

II. Required Topics

A. Background. Formal grammars, languages and their syntax. BNF description of programming languages.

B. Scanners. Finite state automata and regular expressions. Implementation of lexical scanners and symbol tables.

C. Parsers. Theory and examples of context-free languages and push-down automata Context-free parsing techniques such as recursive descent: LL(k), precedence, and bottom-up: LALR, LR(k), LR(k). Syntax error detection.

D. Translation. Syntax directed translation. Intermediate forms. Code generation. Optimization. Semantic error detection.

III. Bibliography

Aho, Sethi & Ullman: Compilers: Principles, Techniques and Tools

Barret & Couch: Compiler Construction: Theory & Practice

Fischer & LeBlanc: Crafting A Compiler