CSC 541 - COMPILER PRINCIPLES AND TECHNIQUES

CREDIT HOURS: 3
PREREQUISITES: CSC 342; CSC 441 or 540
GRADE REMINDER: Must have a grade of C or better in each prerequisite course.

CATALOG DESCRIPTION

Language theory, grammars and recognizers, methods for lexical analysis, top-down and bottom-up parsing, code generation, run-time structures, optimization, error handling.

PURPOSE OF COURSE

To present fundamental concepts of languages and the formal mechanisms for describing and translating languages, to become familiar with compiler-writing tools, and to study all phases of compiling with primary emphasis on parsing and generating code for high-level programming languages.

EDUCATIONAL OBJECTIVES

Upon successful completion of the course, students should be able to:

1. Describe elements of formal language theory that are directly pertinent to lexical analysis and syntax analysis.
2. Describe top-down and bottom-up parsing methods fundamental to performing lexical analysis and syntax analysis on programs written in typical high-level programming languages.
3. Design a compiler to generate object code corresponding to a high-level programming language source program.
4. Describe the importance of the use of formal grammar techniques in the specification of a variety of process control applications.

CONTENT

Survey of Programming Language Features and Overview of Compiling ....................................................... 2

Formal Language Notation and Classification of Languages. ................................................................. 5

Grammars........................................................................................................................................ 5
  Regular grammars and lexical analysis
  Context free grammars and syntactic analysis
  Normal forms

Recognizers. ........................................................................................................................................ 5
  Finite state automata and finite state diagrams
  Pushdown automata

Parsing Techniques. ........................................................................................................................... 15
Top-down recursive descent parsing
Predictive parsing
Bottom-up parsing
Shift/reduce methods
Parse table generators
Use of compiler development tools

Code Generation, Syntax-Directed Translation.......................................................... 5
Run-Time Storage Management. .......................................................... 3
Error Handling: Detection and Recovery. .......................................................... 2
Code Optimization: Loop Optimization, Data Flow Analysis. .................................. 3

TOTAL 45

REFERENCES


