Compiler Design – Spring 2002

Professor : Muhammed F. Mudawwar

Office : Room 733 Falaki Academic Center, x5305

Office Hours : UMW 10 - 12 or by appointment

Email : mudawwar@aucegypt.edu

Web: http://www.cs.aucegypt.edu/~mudawwar/csci447/

Textbook: Kenneth Louden, *Compiler Construction: Principles and Practice*, PWS publishing company, 1997

Reference: Aho, Sethi, and Ullman, *Compilers: Principles, Techniques, and Tools*, Addison Wesley, 1988.

Objectives

This course presents a practical approach to the subject of compiler construction. It is intended not only to cover the components of a compiler, but also how they actually fit together. The use of compiler tools, such as Lex and Yacc, are emphasized to automate the generation of compiler components, wherever applicable.

Subjects

- Introduction to Compiling, the translation process, major data structures in a compiler, programs related to compilers.
- Scanning theory, regular expressions, finite automata, from regular expressions to finite automata.
- Using the Lex Scanner Generator, a TINY language and scanner.
- Hashing, hash tables and symbol tables.
- Context-free grammars, derivations and parse trees, abstract syntax trees, ambiguous grammars, extended BNF notations, Syntax of TINY.
- Recursive-Descent parsing, Syntax tree construction, LL(1) Parsing, First and Follow sets, Predict function, LL(1) parse table.
- Bottom-up parsing, LR parsers, LR(0) items and parsing, SLR(1) parsing.
- Using the Yacc parser generator, eliminating ambiguity and conflicts, error recovery, Yacc parser generation for TINY.
- Semantic Processing: attribute grammars, syntax-directed translation, semantic processing techniques.
- Processing declarations, symbol attributes, dealing with scope, fields and records.
- Data Types and type checking.
- Intermediate code, data structures for code generation, basic code generation techniques.

Assignments

Assignments can be done in groups of 2 or at most 3 students.

- Scanner generation with Lex, symbol table for identifiers and literals.
- Recursive descent parsing, generating a syntax tree.
- Yacc specification, type checking, intermediate code generation.

Grading

Written assignments and quizzes: 15%

Programming Assignments: 30%

Midterm Exam: 20% or 25%

Final Exam: 35% or 30%