Computer Science and Engineering CSE 4723
Compiler Construction

CREDIT/CONTACT HOURS: Credit Hours: 3, Contact Hours: 45

COORDINATOR:
Dr. Yoginder Dandass

Supplemental Material: No required material

SPECIFIC COURSE INFORMATION:
Catalog Description: Formal treatment of context-free programming language translation and compiler design concepts, including: lexical, syntactic and semantic analysis; machine-structures, run-time environments, binding strategies, compilers, and interpreters.

a. Prerequisites: Credit or registration in CSE 4713-6713
b. Required/Elective
   Computer Science – Elective
   Software Engineering – Elective
   Computer Engineering – Elective

SPECIFIC GOALS OF THE COURSE:

a. Specific Outcomes of Instruction:
   1. Students will learn the fundamental architecture of a compiler and how to construct compilers using tools such as flex (lex) and bison (yacc).
   2. Students will learn how to specify and classify grammars through the analysis of the features in the grammar.
   3. Students will learn the role of lexical analysis and how to generate a lexical analyzer.
   4. Students will learn how to perform syntax analysis, top-down parsing, bottom-up parsing, and intermediate code generation.
   5. Students will learn how to generate optimized code from the intermediate code.

b. Criterion 3 Outcomes:
   Note: Parenthesized list indicates the ABET EAC and CAC outcomes addressed by each performance criteria.
   1. Students will learn the fundamental architecture of a compiler and how to construct compilers using tools such as lex and yacc. (EAC: a, c, j, k; CAC: a, b, c, i, k)
   2. Students will learn how to specify and classify grammars through the analysis of the features in the grammar. (EAC: a, e; CAC: a, b, j)
   3. Students will learn the role of lexical analysis and how to generate a lexical analyzer. (EAC: a, b, c, e, k; CAC: a, b, c, i, k)
4. Students will learn how to perform syntax analysis, top-down parsing, bottom-up parsing, and intermediate code generation. (EAC: a, b, c, e, k; CAC: : a, b, c, i, k)

1. Students will learn how to generate optimized code from the intermediate code. (EAC: a, b, c, j; CAC: a, b, c, j)

TOPICS COVERED: (number of class hrs)

1. Compiler Architecture 3
2. Lexical Analysis, flex/lex, design of a lexical analysis generator 12
3. Grammars and syntax analysis 6
4. Syntax-directed translation, bison/yacc and intermediate code representation 12
5. Runtime environment, stack, code generation 6
6. Optimization 6