Goals:

1. A basic understanding of compiler construction

   (a) Theory
      i. Finite Automata
      ii. Context Free Grammars
      iii. Derivations and Parse Trees

   (b) Techniques and topics
      i. Lexical Analysis.
      ii. Symbol tables.
      iii. Parsing Algorithms, top-down vs. bottom-up parsing
          A. Operator precedence parsing
          B. Predictive parsing, LL(1) grammars and recursive descent
          C. SLR parsing, automatic construction of LR parsing tables
      iv. Attribute Grammars and Annotated Parse Trees
      v. Syntax Directed Definitions, and Semantic actions
      vi. Syntax trees
      vii. Semantic checking
      viii. Intermediate code and code generation
      ix. Introduction to code optimization (as time allows)
          A. Basic Blocks
          B. Reaching Definitions
          C. Live Variable Analysis
          D. Data Flow Equations and an Iterative Algorithm for Solving

   (c) Miscellaneous Compiler Topics
      i. Compiler development for a new architecture
         A. “Bootstrapping”
         B. Cross compilers
      ii. Compiler development tools
         A. Scanner generators (e.g., lex, flex)
         B. Parser generators. (e.g., yacc, bison)

2. Develop your programming abilities and organizational skills
Expectations:

1. Class participation / Communicate when things get confusing.
2. Write a compiler for a simplified language.
3. Graduate students are required to implement additional language features in their compiler project.

Grading: 3 exams (55%), 1 programming project, collected/reviewed in stages (45%).

Disability Notice:
If you have a disability that may require an accommodation for taking this course, then please contact the Learning Assistance Center (758-5929) within the first two weeks of the semester.