CMPT 379 - D100 Principles of Compiler Design

Instructor(s): Tom Shermer

Instructor's Objectives:
This course covers the theoretical foundations as well as practical techniques for the construction of a compiler for a high-level programming language. Topics include lexical analysis, parsing, type checking, code generation and optimization. Students will implement an actual compiler for a high-level programming language.

Calendar Objective/Description:
Principles of Compiler Design

Prerequisites:
see go.sfu.ca

Topics:
- Formal-language theory: alphabets, words, languages, regular expressions, finite automata
- Compiler design and implementation principles
- Lexical Analysis: scanners, symbol tables
- Parsing: context-free grammars, parse trees, ambiguity, top-down and bottom-up parsing
- LL(1) parsing: efficient top-down parsing, left-factoring
- Shift-reduce and table-driven parsers: bottom-up parsing
- Type checking: typing rules, types as inferencing rules, Context checking for programs
- Context and code generation: from high-level to machine code
- Introduction to optimization: local, global, and interprocedural.

Grading:
The grade distribution will be handed out at the start of classes.
Students must attain an overall passing grade on the weighted average of exams in the course in order to obtain a clear pass (C- or better).

Required Books:
Engineering a Compiler, Cooper, Torczon, Morgan Kaufmann, 2011, 9780120884780, 2nd Edition

Reference Books:


Academic Honesty Statement::
Academic honesty plays a key role in our efforts to maintain a high standard of academic excellence and integrity. Students are advised that ALL acts of intellectual dishonesty will be handled in accordance with the SFU Academic Honesty and Student Conduct Policies (http://www.sfu.ca/policies/gazette/student.html).