CMPT 379 - D100 Principles of Compiler Design

Instructor(s): Apala Guha

Calendar Objective/Description:
This course covers the key components of a compiler for a high level programming language. Topics include lexical analysis, parsing, type checking, code generation and optimization. Students will work in teams to design and implement an actual compiler making use of tools such as lex and yacc.

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.

Prerequisites:
MACM 201, (CMPT 150, CMPT 295 or ENSC 215) and CMPT 225.

Topics:
- Formal language definitions
- Lexical analysis
- Context free grammars and parse trees
- Top-down parsing
- Bottom-up parsing
- Type checking
- Intermediate code generation
- Target code generation
- Control flow graphs, dataflow analysis example e.g. reachability analysis
- Constant propagation, dead code elimination etc.
- Register allocation

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 Kaufman, 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).