Concordia University
Department of Computer Science
and Software Engineering

Compiler Design
COMP 442/6421 --- Winter 2017

Contact Information

name:	Dr. Joey Paquet
office:	EV 3.221
phone:	(514) 848-2424 ext. 7831
office hours:	Wednesdays 10:00-12:00
e-mail:	paquet@encs.concordia.ca
www:	www.cse.concordia.ca/~paquet

Schedule

lectures	LECT NN	M------	17:45-20:15	MB 3.445	Paquet, Joey	paquet@encs.concordia.ca
laboratories	LAB NN NI	M------	20:30-22:20	H 819	Laleh, Touraj	t_laleh@encs.concordia.ca
	LAB NN NJ	M------	15:45-17:30	H 811	Laleh, Touraj	
	LAB NN NK	M------	20:30-22:20	H 811	Erfani, Mostafa	m_erfa@encs.concordia.ca

Calendar Description

Prerequisites (COMP442): COMP228 or SOEN228 or COEN311; COMP335; COMP352 or COEN352; (COMP6421) : COMP5201, COMP5361; COMP5511. Compiler organization and implementation: lexical analysis and parsing, syntax-directed translation, code optimization. Run-time systems. A project.

Course outline

This course is oriented on the design and implementation of a compiler. Most lectures are directly related to the project. Assignments sequentially cover all the implementation steps of the compiler. The final examination is used to assess the students' theoretical understanding of the material covered in class, which is a fundamental component of this course.

Grading

The evaluation will be based on assignments (4X10%), final project demonstration (30%), and a final examination (30%). Late assignments are assessed a penalty of 50% for each late working day. In all assignments, good design of programs, documentation, and proper testing carry considerable weight. At the end of the course, each student must demonstrate the capabilities of the complete compiler. The final examination covers all material covered in class. The grading scheme used is the same for all students, undergraduate or graduate. The numeric-to-letter grading conversion is made according the class average.

Textbooks

Main Reference

C.N. Fischer, R.K. Cytron, R.J. LeBlanc Jr., *Crafting a Compiler*, Adison-Wesley, 2009.

Other Relevant Sources

T.W. Parsons. *Introduction to Compiler Construction*, W.H. Freeman and Company, 1992.

A.V. Aho, R. Sethi and J.D. Ullman. *Compilers, Principles, Techniques, and Tools*, Addison-Wesley, 1986.

K.C. Louden. *Compiler Construction: Principles and Practice*, International Thomson Publishing Inc., 1997.

Project Details

The project is about the design and implementation of a compiler for a simple programming language. The project is divided into four assignments. Each assignment corresponds to the design and implementation of a major component of the compiler, and makes use of the code base of all previous assignments. Thus, the project involves a substantial amount of incremental coding. You can write the compiler in any language you are proficient with. You are not allowed to use compiler-generation tools. You are allowed to use any computer that is available to you for the implementation. However, you must do the final project demonstration in the allocated laboratory. The project is due on the last week of classes, where final project demonstrations are to be done individually. No extensions of this deadline is possible. Students are encouraged to discuss the design and implementation issues of the project among them. However, each student must work on his/her individual implementation of the project. Note that you are responsible for the design of a complete set of tests for each part of the project. You are encouraged to cooperate with other students on this matter. Completeness of testing will be a major issue in the grading of the assignments and the project.

Graduate Attributes

As part of either the Computer Science or Software Engineering program curriculum, the content of this course includes material and exercises related to the teaching and evaluation of graduate attributes. Graduate attributes are skills that have been identified by the Canadian Engineering Accreditation Board (CEAB) and the Canadian Information Processing Society (CIPS) as being central to the formation of Engineers, computer scientists and information technology professionals. As such, the accreditation criteria for the Software Engineering and Computer Science programs dictate that graduate attributes are taught and evaluated as part of the courses. The following is a description of the list of graduate attributes covered in this course, along with a description of how these attributes will be incorporated in the course.

Knowledge base: Compilation process. Parsing techniques such as CLR, SLR, LALR, recursive descent and table-driven predictive parsing. Syntax-directed translation, intermediate translation languages, symbol tables. Grammars, attribute grammars, attribute migration, grammar transformation. Structure and functioning of run-time systems.
Problem analysis: Determine appropriate parsing and compilation techniques to be applied for different language constructs. Grammar analysis and transformation.
Design: Design and implement a full compiler including lexical analysis, parsing, semantic analysis, code generation, and run-time system.
Use of tools: Use of an appropriate tools, programming language and libraries for the development of a full implementation of a compiler. Use of analysis tools to transform/validate lexical and grammatical specifications.
Communication skills: Deliver the final project in an oral presentation.

Learning Outcomes

Demonstrate knowledge of the theory involved in compilers and its practical implementation.	[ind. 1.3]
Identify, formulate, and develop theoretical models of different parts of a compiler.	[ind. 2.1, 2.2]
Develop a compiler design adapted to the language as specified.	[ind. 4.3]
Implement and test a compiler.	[ind. 4.4]
Demonstrate operational use of appropriate tools, language, and libraries to implement a compiler.	[ind. 5.1, 5.2]
Deliver an operational product demonstrated to respect specifications and design constraints.	[ind. 7.4]