Concordia University
Department of Computer Science and Software Engineering

Compiler Design
COMP 442/6421 --- Winter 2015

Contact Information

name	Joey Paquet
office	EV 3-221
phone	(514) 848-2424 ext. 7831
office hours:	Fridays 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	H544	Paquet, Joey	paquet@encs.concordia.ca
laboratories	LAB NN NI	M------	20:30-22:20	TBD	Laleh, Touraj	t_laleh@encs.concordia.ca
	LAB NN NJ	M------	15:45-17:30	TBD	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

Assignments (4)	40%
Final Examination	30%
Final Project	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.

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 both the Computer Science and 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. This particular course aims at teaching and evaluating 3 graduate attributes. The following is a description of these attributes, along with a description of how these attributes will be incorporated in the course.

Problem analysis *is the ability to use appropriate knowledge and skills to identify, analyze, and solve complex engineering problems in order to reach substantiated conclusions.* This course covers this attribute in the following ways: Determine appropriate parsing and compilation techniques to be applied for different language constructs. Grammar analysis and transformation.

Design *is the ability to design solutions for complex, open-ended engineering problems and to design systems, components or processes that meet specified needs with appropriate attention to health and safety risks, applicable standards, and economic, environmental, cultural and societal considerations.* This course covers this attribute through the design and implement a full compiler including lexical analysis, parsing, semantic analysis, code generation, and run-time system.

Use of tools *is the ability to create, select, apply, adapt, and extend appropriate techniques, resources, and modern engineering tools to a range of engineering activities, from simple to complex, with an understanding of the associated limitations.* This course covers this attribute through the use of a grammar analysis tool, and an appropriate programming language and libraries for the full implementation of a compiler.

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.