

Master Course Syllabus
School of Engineering and Computer Science
Washington State University Vancouver

CS 452
Compiler Design
3 Semester Hours
(3 lecture hours)

Catalog Description

Design of lexical analyzers, syntactic analyzers, intermediate code generators, code optimizers and object code generators.

Prerequisite Courses

CS 317 Automata and Formal Languages
CS 355 Programming Language Design

Prerequisite Topics

- Regular Languages, Context-Free Languages and Grammars.
- Proficient in an imperative programming language.
- Programming language design issues: e.g., type checking, scope, memory management, and object-oriented support.

Measured Course Outcomes

Students taking this course will (among other things):

1. Modify language grammars so that they are unambiguous and are appropriate for either LL (top-down recursive descent) and/or LR parsing schemes (*Contributes to performance criterion A-1*).
2. Construct a parser for a (perhaps small) programming language that generates code for some target (perhaps pedagogical) execution model (*Contributes to performance criterion K-4*).
3. Use a scanner or parser generator tool (e.g., LEX, YACC) (*Contributes to performance criterion I-3*).

Required Textbooks

Compilers: Principles, Techniques, and Tools (2nd Edition) by Alfred V. Aho, Monika Lam, Ravi Sethi and Jeffrey D. Ullman. Addison-Wesley.

Reference Material

Major Topics Covered in the Course

1. Language Translation Overview
 - a. Interpreters and compilers
 - b. Source language types and target execution models
 - c. Virtual Machines, Just in Time (JIT) compilers
2. Lexical Analysis (scanning)
3. Syntax Analysis (parsing)
 - a. Top-down LL parsing schemes, recursive descent parsing
 - b. Bottom-up LR parsing schemes, using parser generator tools
 - c. Static semantic checking
4. Semantic Processing and Code Generation
5. Symbol Tables
6. Run-time System support
 - a. Static, stack, heap memory allocation
 - b. Dynamic heap memory management schemes, garbage collection
7. Translation of language features
 - a. Processing declarations, expressions, data structure references
 - b. Translating flow-control structures, procedures, and functions
 - c. Object-oriented support
8. Global and local optimization

Laboratory Projects

1. Recursive Descent Parser for small, pedagogical language.
2. LALR(1) parser using parser generator tool (e.g., YACC).

CSAB Category Content

	FUNDAMENTAL	ADVANCED		FUNDAMENTAL	ADVANCED
Data Structures	0	0	Computer Organization and Architecture	0	1
Algorithm & Software Design	0	0	Concepts of Programming Languages	0	2

Oral and Written Communications

None Specified

Social and Ethical Issues

None Specified

Theoretical Content

Topic	Hours
Formal languages translation.	12
Regular languages and automata.	1
Context free languages, grammars, parse trees.	3
Attribute grammars and syntax trees.	3

Problem Analysis

1. Regular and context-free language theory is used to describe a programming language's lexical and syntactical structure. This allows for formal analysis of these language elements.

Solution Design

1. Language translation as a sequence of well-defined phases provides a powerful example of a divide-and-conquer problem solving technique.

CC2001

This course provides coverage of topics in the following areas (hours listed are minimums):

AL7. Automata theory [elective]	1
PL2. Virtual machines [core]	1
PL3. Language translation [core]	9
PL4. Declarations and types [core]	2
PL5. Abstraction mechanisms [core]	1
PL6. Object-oriented programming [core]	2
PL8. Language translation systems [core]	9
PL10. Programming language semantics [elective]	3

Course Coordinator: Wayne O. Cochran
Last Updated: January 3, 2007 (Approved)
Syllabus Version Number: 1.01