San José State University
Department of Computer Engineering

CMPE 152 Compiler Design

Section 4 (class) and Section 5 (lab)
Fall 2017

Course and Contact Information

Instructor:	Ron Mak
Office Location:	ENG 250
Email:	ron.mak@sjsu.edu
Website:	http://www.cs.sjsu.edu/~mak/
Office Hours:	TuTh 3:00 - 4:00 PM
Class Days/Time:	Class: TuTh 4:30 - 5:20 PM Lab: Tu 6:00 - 8:45 PM
Classroom:	Class: ENG 403 Lab: ENG 206
Prerequisites:	CS 47 or CMPE 102, CS 146, and CS 154 (with a grade of "C-" or better in each); Computer Science, Applied and Computational Math, or Software Engineering majors only; or instructor consent.

Course Format

This course will be taught primarily face-to-face instruction. Course materials, syllabus, assignments, grading criteria, exams, and other information will be posted on the SJSU Canvas course site at http://sjsu.instructure.com/ You are responsible to check Canvas regularly for class work and exams. You also can find Canvas video tutorials and documentations at http://ges.sjsu.edu/canvas-students

Faculty Web Page and MySJSU Messaging

Course materials such as syllabus, handouts, notes, assignment instructions, etc. can be found on my faculty web page at http://www.sjsu.edu/people/firstname.lastname and/or on Canvas Leaning Management System course login website at http://sjsu.instructure.com. You are responsible for regularly checking with the messaging system through MySJSU at http://my.sjsu.edu to learn of any updates.

Piazza will be available for announcements and to serve as an online discussion forum for the class. You are responsible for responding to enrollment invitations.

Course Catalog Description

"Principles of lexical analysis, finite state automata and parsing; issues of variable declarations, variable types, control statements, function calls, nested scopes and efficient assembler target code."

Course Goals

- **Compiler construction.** Design and build a working compiler for a programming language that you invented. Write sample programs in your language, compile your programs into byte code for the Java Virtual Machine to produce .class files, and then successfully run your programs on the JVM.
- **Software engineering.** Employ the best practices of object-oriented design and team-based software engineering. A compiler is a large, complex program! Managing the development of such a program requires learning *critical job skills that are highly desired by employers*.

Course Learning Outcomes (CLO)

Upon successful completion of this course, you will be able to:

- Develop a scanner and a parser for a procedure-oriented programming language.
- Generate a symbol table and intermediate code.
- Perform semantic analysis such as type checking.
- Develop an interpreter that creates a suitable runtime environment from the intermediate code and the symbol table and executes the source program.
- Use a compiler-compiler to generate a parser and a scanner based on a written grammar for an existing programming language or an invented language.
- Develop a compiler that generates assembly language object code that can be assembled into executable code for a real or a virtual machine.

Required Texts/Readings

Textbook

Title:	**Writing Compilers and Interpreters**, *3rd edition*
Author:	Ronald Mak
Publisher:	Wiley Publishers, Inc.
ISBN:	978-0-470-17707-5
Source files:	http://www.cs.sjsu.edu/~mak/CMPE152/sources (both Java and C++ source files are available)
Title:	**The Definitive ANTLR 4 Reference,** *2nd edition*
Author:	Terence Parr
Publisher:	Pragmatic Bookshelf
ISBN:	978-1934356999
	http://www.antlr.org

We will use the **ANTLR 4 compiler-compiler** during the second half of the course, so you won't need the ANTLR text until then. ANTLR 4 can generate compiler components written in either Java or C++.

Other Readings

We will use Pascal as an example source language during semester. These online Pascal tutorials are helpful:

Pascal Tutorial looks very good. It even has an online compiler.
Learn Pascal also looks good, although it doesn't appear to cover set types.

Course Requirements and Assignments

You must have good C++ programming skills and know how to use software development tools such as Eclipse CDT.

You will work during the semester in small four-person teams. Weekly assignments during the first part of the semester will provide practice with compiler design techniques and give students experience adding new features to a large legacy code base. During the latter part of the semester, each student team will develop a working compiler for an existing language or for a *newly invented language*. Teams will be able to write, compile, and execute programs written in their chosen or invented languages.

This is a challenging course that will demand much of your time and effort throughout the semester.

The university's syllabus policies:

- University Syllabus Policy S16-9 at http://www.sjsu.edu/senate/docs/S16-9.pdf.
- Office of Graduate and Undergraduate Programs' Syllabus Information web page at http://www.sjsu.edu/gup/syllabusinfo/

The University's Credit Hour Requirement:

"Success in this course is based on the expectation that students will spend, for each unit of credit, a minimum of 45 hours over the length of the course (normally 3 hours per unit per week with 1 of the hours used for lecture) for instruction or preparation/studying or course related activities including but not limited to internships, labs, clinical practica. Other course structures will have equivalent workload expectations as described in the syllabus."

Team Compiler Project

Each student team will work on a compiler project throughout the semester. Each project involves:

- Choosing either an existing language (or subset of a language) or inventing a new language.
- Creating a grammar for the language.
- Generate a compiler for the language using the ANTLR compiler-compiler. Other components may be borrowed from the compiler code given in the class.

An acceptable compiler project has at least these features:

- Two data types with type checking.
- Basic arithmetic operations with operator precedence.
- Assignment statements.
- A conditional control statement (e.g., IF).
- A looping control statement.
- Procedures or functions with calls and returns.
- Parameters passed by value or by reference.
- Basic error recovery (skip to semicolon or end of line).
- Nontrivial sample programs written in the source language.
- Generate Jasmin assembly code that can be successfully assembled.
- Execute the resulting **.class** file.

- No crashes (e.g., null pointer exceptions).

Each team will write a report (5-10 pp.) that includes:

- A high-level description of the design of the compiler with UML diagrams of the major classes.
- The grammar for your source language, either as syntax diagrams or in BNF.
- Code templates that show the Jasmin code your compiler generates for some key constructs of the source language.

Final Examination

Besides a final project from each team, there will be a written in-class final examination for each student. The exam will test understanding (not memorization) of the material taught during the semester and now well each you participated in your project team.

Grading Information

Each assignment will be worth 100 points. For each team assignment, each team member will receive the same score. Late assignments will be penalized 25% and an additional 25% for each subsequent day.

Individual *total scores* will be computed with these weights:

30%	Assignments
35%	Compiler project
15%	Midterm exam
20%	Final exam

Class grades will be based on a curve. The median total score will earn a B. Depending on how all the total scores cluster above and below the median, approximately one quarter of the class will earn higher grades, and another one quarter will earn lower grades.

There can be no make-up midterm or final exams without a valid medical excuse.

Postmortem report

At the end of the semester, each student must also turn in a short (1 page) individual postmortem report that includes:

- A brief description of what you learned in the course.
- An assessment of your accomplishments for your project team on the assignments and the compiler project.
- An assessment of each of your other project team members.

Only the instructor will see these reports. How your teammates evaluate you may affect your class grade.

Classroom Protocol

It is very important for each student to attend classes and to participate. Cell phones in silent mode, please.

University Policies

Per University Policy S16-9, university-wide policy information relevant to all courses, such as academic integrity, accommodations, etc. will be available on Office of Graduate and Undergraduate Programs' Syllabus Information web page at http://www.sjsu.edu/gup/syllabusinfo/.

CMPE 152 Compiler Design

Section 4 (class) and Section 5 (lab)
Fall 2017

This schedule is subject to change with fair notice which will be communicated through emails and announcements via Canvas and Piazza.

- WCI = *Writing Compilers and Interpreters, 3rd edition*
- ANTLR = *The Definitive ANTLR 4 Reference, 2nd edition*

Course Schedule

Week	Date	Topics	Readings
1	Aug 24	Overview of the course What are compilers and interpreters? A software framework for compilers and interpreters *Form programming teams*	WCI 1, 2, 3
2	Aug 29 Aug 31	Syntax diagrams Scanning (lexical analysis) Symbol table management Top-down recursive-descent parsing	WCI 4 WCI 5
	Aug 29 lab	Install software Write, compile, and run a Pascal program	
3	Sept 5 Sept 7	Parsing assignment statements and expressions Intermediate code (parse trees)	WCI 5
	Sept 5 lab	Create a scanner and a symbol table	
4	Sept 12 Sept 14	Interpreting assignment statements and expressions Parsing control statements Parser error handling	WCI 6, 7
	Sept 12 lab	Add new control statements	
5	Sept 19 Sept 21	Interpreting control statements Runtime error handling Parsing declarations	WCI 8, 9
	Sept 19 lab	Execute new control statements	
6	Sept 26 Sept 28	Parsing declarations*, cont'd* Semantic actions and type checking	WCI 9, 10
	Sept 26 lab	Add new data types	
7	Oct 3 Oct 5	Scope and the symbol table stack Parsing programs, procedures, and functions Parsing procedure and function calls Runtime memory management The runtime stack and activation frames	WCI 11, 12
	Oct 3 lab	Implement runtime memory management	

Week	Date	Topics	Readings
8	Oct 10 Oct 12	Passing parameters by value and by reference ***Midterm exam Thursday, October 12 (75 min.)***	WCI 12
	Oct 10 lab	Interpret Pascal programs *Midcourse review*	
9	Oct 17 Oct 19	A simple DFA scanner BNF grammars for programming languages The ANTLR compiler-compiler	ANTLR 1-4
	Oct 17 lab	Create an ANTLR grammar	
10	Oct 24 Oct 26	Generating a scanner and a parser with ANTLR	ANTLR 5, 6
	Oct. 24 lab	Generate a scanner and parser for your programming language.	
11	Oct 31 Nov 2	The Java Virtual Machine (JVM) architecture Jasmin assembly language Code templates and code generation	WCI 15 ANTLR 7, 8
	Oct 31 lab	Design code templates	
12	Nov 7 Nov 9	Code for expressions Code for assignment statements	WCI 16 ANTLR 9
	Nov 7 lab	Code generation for assignment statements and expressions	
13	Nov 14 Nov 16	Code for procedure and function calls Code to pass parameters by value and by reference Code for string operations	WCI 17
	Nov 14 lab	Code generation for procedures and functions	
14	Nov 21	Code for control statements Code for arrays Code for records	WCI 18
	Nov 21 lab	Code generation for control statements, arrays, and records	
15	Nov 28 Nov 30	Executing compiled Pascal programs Bottom-up parsing Lex and Yacc Code optimization	
	Nov 28 lab	Code generation for Pascal programs	
16	Dec 5 Dec 7	Compiling object-oriented languages An interactive source-level debugger A multi-threaded GUI-based debugger Heap, stack, and garbage collection *Course review*	WCI 13, 14, 19
	Dec 5 lab	Project presentations	
Final Exam	Wednesday Dec 13	Time: 2:45 - 5:00 PM Room: ENG 403	