San José State University
Department of Computer Science

CS/SE 153
Concepts of Compiler Design

Section 1
Fall 2019

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 4:30 - 5:30 PM
Class Days/Time:	TuTh 9:00 – 10:15 AM
Classroom:	Class: MacQuarrie Hall MH 222
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

Class meetings will each consist of a lecture and a lab session.

Faculty Web Page and Canvas

Course materials, syllabus, assignments, grading criteria, exams, and other information will be posted at my faculty website at http://www.cs.sjsu.edu/~mak and on the Canvas Learning Management System course login website at http://sjsu.instructure.com. You are responsible for regularly checking these websites to learn of any updates. You can find Canvas video tutorials and documentations at http://ges.sjsu.edu/canvas-students

Course Catalog Description

"Theoretical aspects of compiler design, including parsing context free languages, lexical analysis, translation specification and machine-independent code generation. Programming projects to demonstrate design topics."

Course Goals

This course will concentrate on practical aspects of compiler construction, programming language design, and engineering a large, complex software application.

- **Compiler construction and language design.** Design and build a working compiler for a programming language that you invented. Write sample programs in your language and then compile them into executable machine code that you can run.
- **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, students will be able to:

CLO 1: Develop a **scanner** and a **parser** for a programming language.
CLO 2: Perform **syntactic and semantic analyses** of source programs.
CLO 3: Generate **symbol tables** and **intermediate code** for source programs.
CLO 4: Develop an **interpreter** that executes a source program in a suitable runtime environment.
CLO 5: Design the **grammar** for a programming language and feed it into a **compiler-compiler**.
CLO 6: Develop a **compiler** that translates a source program into **executable machine code**.
CLO 7: Engineer a large, complex software application.

Required Texts

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/CS153/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++.

We will use Pascal as an example source language. 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.

Some online websites to compile and run Pascal programs:

http://rextester.com/l/pascal_online_compiler
https://www.tutorialspoint.com/compile_pascal_online.php
https://www.jdoodle.com/execute-pascal-online

Source Codes

Source codes from the *Writing Compilers and Interpreters* textbook, in both the original Java and ported to C++, are available for download at http://www.cs.sjsu.edu/~mak/CS153/sources.

Course Requirements and Assignments

You must have good Java programming skills and be familiar with development tools such as Eclipse.

You will form project teams of four students each. *Team membership is mandatory for this class.* The teams will last throughout the semester. Once the teams are formed, you will not be allowed to move from one team to another, so form your teams wisely!

Weekly team-based **lab assignments** will provide practice with compiler design techniques and give you experience adding new features to a large legacy code base. Each student on a team will receive the same score for each team assignment.

Each team will submit its assignments into Canvas, where the rubric for scoring each will be displayed. Each assignment and project will be worth up to 100 points. Late assignments will lose 20 points and an additional 20 points for each 24 hours after the due date.

Because class meetings each will have a lab session, you are expected to have done the required reading before coming to class. 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 Program's Syllabus Information web page at http://www.sjsu.edu/gup/syllabusinfo/

"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

In addition to the team assignments, each student team will work on a compiler project throughout the semester. Each team will develop a working compiler for a newly invented language or for an existing language. Teams will be able to write, compile, and execute programs written in their invented or chosen languages. *Each student on a team will receive the same score for the team project.* Each project involves:

- Invent a new programming language or choose a subset of an existing language.
- Develop a grammar for the language.
- Generating a compiler for the language using the ANTLR compiler-compiler. Other components may be borrowed from the compiler code given in the class.

A **minimally 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.

Exams

The midterm and final examinations will be closed book. There can be no make-up midterm examination unless there is a documented medical emergency. Make-up final examinations are available only under conditions dictated by University regulations.

The exams will test understanding (not memorization) of the material taught during the semester and now well each of you participated in your team assignments and project.

Grading Information

Individual total scores will be computed with these weights:

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

* *team scores*
** *individual scores*

Course grades will be based on a curve. The median total score will earn a B–. Approximately one third of the class will earn higher grades, and another one third will earn lower grades.

Postmortem Report

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

- A brief description of what you learned in the course.
- An assessment of your accomplishments for your team assignments and design 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 course grade.

Classroom Protocol

It is very important for each student to attend classes and to participate. Mobile devices 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 Program's Syllabus Information web page at http://www.sjsu.edu/gup/syllabusinfo/.

CS/SE 153
Concepts of Compiler Design

Section 1
Fall 2018

Course Schedule (subject to change with fair notice)

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

Because class meetings each will have a lab session, you are expected to have done the required reading before coming to class.

Week	Dates	Topics	Readings
1	Aug 22	Overview of the course What are compilers and interpreters? *Form programming teams*	WCI 1, 2
2	Aug 27 Aug 29	A software framework for compilers and interpreters Scanning (lexical analysis) **Lab: Write Pascal programs**	WCI 3
3	Sep 3 Sep 5	Basic scanning algorithm Symbol table management Top-down recursive-descent parsing Syntax diagrams **Lab: Scanning**	WCI 4, 5
4	Sep 10 Sep 12	Parse assignment statements and expressions Intermediate code (parse trees) Interpret assignment statements and expressions Parsing control statements Parser error handling **Lab: Parsing**	WCI 6, 7
5	Sep 17 Sep 19	Interpret control statements Runtime error handling Scope and the symbol table stack Parsing declarations **Lab: Executing control statements**	WCI 8, 9
6	Sep 24 Sep 26	Parsing declarations*, cont'd* Semantic actions and type checking **Lab: Type declarations**	WCI 9, 10
7	Oct 1 Oct 3	Parse programs, procedures, and functions Parse procedure and function calls Runtime memory management The runtime stack and activation frames **Lab: Executing programs**	WCI 11, 12

Week	Dates	Topics	Readings
8	Oct 8 Oct 10	Pass parameters by value and by reference **Lab: Executing programs, *cont'd*** *Midcourse review* ***Midterm exam Thursday, October 10***	WCI 12
9	Oct 15 Oct 17	A simple DFA scanner BNF grammars for programming languages The ANTLR compiler-compiler **Lab: ANTLR 4 grammar**	ANTLR 1-4
10	Oct 22 Oct 24	Generate a scanner and a parser with ANTLR ANTLR listener and visitor interfaces **Lab: ANTLR 4 grammar, *cont'd***	ANTLR 5, 6
11	Oct 29 Oct 31	Pcl, a tiny subset of Pascal ANTLR-generated compiler components for Pcl The Java Virtual Machine (JVM) architecture Jasmin assembly language Code templates and code generation **Lab: Code generation**	WCI 15 ANTLR 7, 8
12	Nov 5 Nov 7	Pcl2 code generation example Code for expressions Code for assignment statements Code for control statements Code for procedure and function calls **Lab: Code generation, *cont'd***	WCI 16, 17 ANTLR 9
13	Nov 12 Nov 14	Code to call `System.out.printf()` Code for string operations Code for arrays and records The Pascal runtime library Code to pass parameters by value and by reference **Lab: Code generation, *cont'd***	WCI 18
14	Nov 19 Nov 21	Compiled vs. interpreted code Context-free vs. context-sensitive grammars Bottom-up parsing with yacc and lex Code optimization **Lab: Code generation, *cont'd***	
15	Nov 26	Runtime memory management Garbage collection algorithms **Lab: Code generation, *cont'd***	
16	Dec 3 Dec 5	Team compiler demos (optional)	
Final Exam	***Monday, Dec 16***	Time: 7:15 - 9:30 AM Room: MacQuarrie Hall MH 222	