San José State University
Department of Computer Science

CS/SE 153
Concepts of Compiler Design

Section 1
Fall 2022

Course and Contact Information

Instructor:	Ron Mak
Office location:	Campus office TBA
Email:	ron.mak@sjsu.edu
Website:	http://www.cs.sjsu.edu/~mak/
Office hours:	Tu 3:15 – 4:15 PM online via Zoom
Class days/time:	TuTh: 4:30 – 5:45 PM
Classroom:	Tuesday: online via Zoom Thursday: MacQuarrie Hall 233
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 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 Format

This is a "hybrid" class that will meet online via Zoom on Tuesdays and in-person on Thursdays. To participate in classroom activities, submit assignments, and take tests/exams remotely, a student must have a computer with adequate internet connection and bandwidth for accessing Canvas and attending Zoom video meetings. A smartphone or tablet with a camera capable of running Zoom is also needed for video recording of your test environment during the tests/exams.

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 Goals

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

Programming language translation in various forms:

- **Interpreter with an interactive symbolic debugger.** Execute a program written in a procedural language and be able to set breakpoints, single-step, examine and set variable values, etc.
- **Language conversion.** Convert a program written in one high-level language to an equivalent program written in another high-level language.
- **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 compile them into assembly language code that you can then assemble and 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** with an interactive **symbolic debugger** 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: Use the **ANTLR 4** compiler-compiler tools.
CLO 8: Engineer a large, complex software application.

Required Text

Title:	**The Definitive ANTLR 4 Reference, 2nd edition**
Author:	Terence Parr
Publisher:	Pragmatic Bookshelf, 2013
ISBN:	978-1934356999
	http://www.antlr.org

Recommended Text

Title:	**Writing Compilers and Interpreters, 3rd edition**
Author:	Ronald Mak
Publisher:	Wiley Publishing, Inc., 2009
ISBN:	ISBN: 978-0-470-17707-5
	Source files

Online Pascal Tutorials

We will use Pascal as the example source language.

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

Other Useful Tutorials

Install the ANTLR 4 plug-in for Eclipse, etc.

Software to Install

You should install and use an interactive development environment (IDE) such as Eclipse or IntelliJ. To develop a compiler for your language, you will need to download and install the ANTLR 4 package and its Eclipse or IntelliJ plugin, and then modify them to generate the compiler components in Java. This is relatively straightforward on the Mac and Linux platforms. However, the Windows platform may have compatibility challenges. Therefore, if you're on Windows, you should download and install the Windows Subsystem for Linux and then download and run Ubuntu (a variant of Linux): https://docs.microsoft.com/en-us/windows/wsl/install-win10

Some useful tutorials:

- "Install Ubuntu on Windows 10 and on VirtualBox" http://www.cs.sjsu.edu/~mak/tutorials/InstallUbuntu.pdf
- "Configure Ubuntu for Software Development" http://www.cs.sjsu.edu/~mak/tutorials/ConfigureUbuntu.pdf
- "Install and Configure ANTLR 4 for Ubuntu and MacOS X" http://www.cs.sjsu.edu/~mak/tutorials/InstallANTLR4.pdf

Course Requirements and Assignments

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

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.

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 three hours per unit per week) for instruction, preparation/studying, or course related activities, including but not limited to internships, labs, and clinical practica. Other course structures will have equivalent workload expectations as described in the syllabus."

Team Compiler Project

In addition to the team-based assignments, each project team will develop a compiler project during 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 statement.
- 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).

Note: Taking an existing compiler and simply replacing the source language's keywords is not an acceptable project.

Each team will give an oral presentation at the end of the semester that includes a demo of its compiler. The rest of the class (along with the instructor) will score each presentation based on a given set of criteria. *Each student on a team will receive the same score for the project.*

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 assembly code your compiler generates for some key constructs of the source language.

Postmortem Report

At the end of the semester, each student must also turn in a short (1 or 2 pages) **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.

Technology Requirements

Students are required to have an electronic device (laptop, desktop, or tablet) with a camera and microphone. SJSU has a free equipment loan program available for students: https://www.sjsu.edu/learnanywhere/equipment/index.php

Students are responsible for ensuring that they have access to reliable Wi-Fi during tests. If students are unable to have reliable Wi-Fi, they must inform the instructor, as soon as possible or at the latest one week before the test date to determine an alternative. See Learn Anywhere website for current Wi-Fi options on campus.

Exams

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. Instant messaging, e-mails, texting, tweeting, file sharing, or any other forms of communication with anyone else during the exams will be strictly forbidden.

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

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.

Zoom Classroom Etiquette

- **Mute your microphone.** To help keep background noise to a minimum, make sure you mute your microphone when you are not speaking.
- **Be mindful of background noise and distractions.** Find a quiet place to "attend" class, to the greatest extent possible.
- Avoid video setups where people may be walking behind you, people talking, making noise, etc.
- Avoid activities that could create additional noise, such as shuffling papers, listening to music in the background, etc.
- **Position your camera properly.** Be sure your webcam is in a stable position and focused at eye level.
- **Limit your distractions and avoid multitasking.** You can make it easier to focus on the meeting by turning off notifications, closing or minimizing running apps, and putting your smartphone away (unless you are using it to access Zoom).
- **Use appropriate virtual backgrounds.** If using a virtual background, it should be appropriate and professional and should not suggest or include content that is objectively offensive and demeaning.

Recording Zoom Classes

The online portions of this course (i.e., lectures, discussions, student presentations) will be recorded for instructional or educational purposes. The recordings will be posted to the class webpage. The recordings will be deleted at the end of the semester. **If you prefer to remain anonymous** during these recordings, then please communicate with the instructor about possible accommodations (e.g., temporarily turning off identifying information from the Zoom session, including student name and picture, prior to recording).

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 2022
Instructor: Ron Mak

Course Schedule (subject to change with fair notice)

Week	Dates	Topics
1	Aug 23 Aug 25	Overview of the course What are compilers and interpreters? *Form programming teams* Syntax diagrams A simple scanner Basic scanning algorithm **Lab: Scanner**
2	Aug 30 Sep 1	A simple parser Top-down recursive-descent parsing Symbol tables Parse trees Parse assignment statements and expressions Parse control statements **Lab: Parser**
3	Sep 6 Sep 8	Visit parse tree nodes Execute assignment statements and expressions Execute control statements Syntax and semantics A simple DFA scanner **Lab: Execute simple programs**
4	Sep 13 Sep 15	BNF grammars for programming languages The ANTLR compiler-compiler Generate a scanner and a parser with ANTLR ANTLR parse tree visitor interfaces An ANTLR-based Pascal interpreter Execute statements and expressions with visitors
5	Sep 20 Sep 22	Declarations and the symbol table Scope and the symbol table stack **Lab: ANTLR 4 grammar**
6	Sep 27 Sep 29	Multipass compilers Parsing array and record declarations Strong typing and type checking Cross-reference listing **Lab: Pascal interpreter**

Week	Dates	Topics
7	Oct 4 Oct 6	Runtime memory management The runtime stack and stack frames Programs, procedures, and functions Procedure and function calls A language converter: Pascal to C++
8	Oct 11 Oct 13	***Midterm exam Tuesday, October 11*** The Java Virtual Machine (JVM) architecture Jasmin assembly language **Lab: Language converter**
9	Oct 18 Oct 20	Code templates and code generation Code for expressions Code for assignment statements Code for control statements
10	Oct 25 Oct 27	Code for procedure and function calls Code to call **`printf()`** Code for arrays and records **Lab: Code generation**
11	Nov 1 Nov 3	Code to pass parameters by value and by reference Runtime libraries Code optimization Compiling object-oriented languages
12	Nov 8 Nov 10	Backend compiler architecture Static vs. dynamic scoping Runtime memory management*, cont'd* Garbage collection algorithms
13	Nov 15 Nov 17	A simple source-level debugger An integrated development environment (IDE) Context-free vs. context-sensitive grammars Bottom-up parsing with yacc and lex
14	Nov 22	*(TBD)*
15	Nov 29 Dec 1	*Project presentations*
16	Dec 6	Project lab
	Thursday, Dec 8	***Final exam*** Time: 2:45 – 5:00 PM