

San José State University  
Department of Computer Science

CS/SE 153  
Concepts of Compiler Design

Section 1  
Fall 2020

**Course and Contact Information**

Instructor: Ron Mak  
Office location: ENG 250 (but working from home)  
Email: [ron.mak@sjsu.edu](mailto:ron.mak@sjsu.edu)  
Website: <http://www.cs.sjsu.edu/~mak/>  
Office hours: TuTh 4:30 - 5:30 PM online via Zoom  
Class days/time: TuTh 9:00 – 10:15 AM online via Zoom  
Classroom: online  
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 course adopts a synchronous online classroom delivery format. 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](http://www.cs.sjsu.edu/~mak) at <http://www.cs.sjsu.edu/~mak> and on the [Canvas Learning Management System course login website](http://sjsu.instructure.com) 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 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 Text

Title:	<i>The Definitive ANTLR 4 Reference</i> , 2 <sup>nd</sup> edition
Author:	Terence Parr
Publisher:	Pragmatic Bookshelf, 2013
ISBN:	978-1934356999
	<a href="http://www.antlr.org">http://www.antlr.org</a>

## Recommended Text

Title:	<i>Writing Compilers and Interpreters:</i>
Author:	<i>A Software Engineering Approach</i> , 3 <sup>rd</sup> edition Ronald Mak
Publisher:	Wiley Publishers, Inc., 2009
ISBN:	978-0-470-17707-5
Source files:	<a href="http://www.cs.sjsu.edu/~mak/CMPE152/sources">http://www.cs.sjsu.edu/~mak/CMPE152/sources</a> (both Java and C++ source files are available)

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

<a href="#">Pascal Tutorial</a> looks very good. It even has an online compiler.
<a href="#">Learn Pascal</a> also looks good, although it doesn't appear to cover set types.

Some online websites to compile and run Pascal programs:

<a href="http://rextester.com/l/pascal_online_compiler">http://rextester.com/l/pascal_online_compiler</a>
<a href="https://www.tutorialspoint.com/compile_pascal_online.php">https://www.tutorialspoint.com/compile_pascal_online.php</a>
<a href="https://www.jdoodle.com/execute-pascal-online">https://www.jdoodle.com/execute-pascal-online</a>

## Software to Install

You should install and use an interactive development environment (IDE) such as Eclipse. To develop a compiler for your language, you will need to download and install the ANTLR 4 package and its Eclipse 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 has significant 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 Eclipse for Java and C++ Development”  
<http://www.cs.sjsu.edu/~mak/tutorials/InstallEclipse.pdf>
- “Install and Configure ANTLR 4 for Ubuntu and MacOS X”  
<http://www.cs.sjsu.edu/~mak/tutorials/InstallANTLR4.pdf>
- “Install and Configure ANTLR 4 for C++”  
<http://www.cs.sjsu.edu/~mak/tutorials/InstallANTLR4Cpp.pdf>

## 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.

***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](http://www.sjsu.edu/senate/docs/S16-9.pdf) at <http://www.sjsu.edu/senate/docs/S16-9.pdf>.
- Office of Graduate and Undergraduate Program's [Syllabus Information web page](http://www.sjsu.edu/gup/syllabusinfo/) 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.

## 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.

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

This course or 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).

## Students are Not Allowed to Record

Students are prohibited from recording class activities (including class lectures, office hours, advising sessions, etc.), distributing class recordings, or posting class recordings. Materials created by the instructor for the course (syllabi, lectures and lecture notes, presentations, etc.) are copyrighted by the instructor. This university policy ([S12-7](#)) is in place to protect the privacy of students in the course, as well as to maintain academic integrity through reducing the instances of cheating. Students who record, distribute, or post these materials will be referred to the Student Conduct and Ethical Development office. Unauthorized recording may violate university and state law. It is the responsibility of students that require special accommodations or assistive technology due to a disability to notify the instructor.

## Proctoring Software and Exams

Exams will be proctored in this course through Respondus Monitor, LockDown Browser, and Zoom video meeting. Please note it is the instructor's discretion to determine the method of proctoring. If cheating is suspected the proctored videos may be used for further inspection and may become part of the student's disciplinary record. Note that the proctoring software does not determine whether academic misconduct occurred but does determine whether something irregular occurred that may require further investigation. Students are encouraged to contact the instructor if unexpected interruptions (from a parent or roommate, for example) occur during an exam. Please refer to the online exam instructions for details of the setup and requirements.

## Technical Difficulties

- **Internet connection issues:** Canvas autosaves responses a few times per minute as long as there is an internet connection. If your internet connection is lost, Canvas will warn you but allow you to continue working on your exam. A brief loss of internet connection is unlikely to cause you to lose your work. However, a longer loss of connectivity or weak/unstable connection may jeopardize your exam.
- **Other technical difficulties:** Immediately notify the instructor and explain the problem you are facing. Your instructor may not be able to respond immediately or provide technical support. However, the current state of your exam and communication will provide a record of the situation.

Contact the SJSU technical support for Canvas:

Technical Support for Canvas  
[Email: ecampus@sjsu.edu](mailto:ecampus@sjsu.edu)

Phone: (408) 924-2337

<https://www.sjsu.edu/ecampus/support/>

### **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](http://www.sjsu.edu/gup/syllabusinfo/) at <http://www.sjsu.edu/gup/syllabusinfo/>.

# CS/SE 153

## Concepts of Compiler Design

Section 1  
Fall 2020

**Course Schedule** (subject to change with fair notice)

Week	Dates	Topics	Readings
1	Aug 20	Overview of the course What are compilers and interpreters? <i>Form programming teams</i> <b>Lab: Write Pascal programs</b>	1
2	Aug 25 Aug 27	Syntax diagrams A simple scanner Basic scanning algorithm A simple parser Top-down recursive-descent parsing <b>Lab: Scanner</b>	2
3	Sep 1 Sep 3	Symbol tables Parse trees Parse assignment statements and expressions Parse control statements Syntax and semantics <b>Lab: Parser</b>	3
4	Sep 8 Sep 10	Visit parse tree nodes Execute assignment statements and expressions Execute control statements A simple DFA scanner BNF grammars for programming languages The ANTLR compiler-compiler <b>Lab: Execute simple programs</b>	4
5	Sep 15 Sep 17	Generate a scanner and a parser with ANTLR ANTLR parse tree visitor interfaces An ANTLR-based Pascal interpreter Execute statements and expressions with visitors <b>Lab: ANTLR 4 grammar</b>	5
6	Sep 22 Sep 24	Scope and the symbol table stack Parsing declarations Strong typing and type checking <b>Lab: Pascal interpreter</b>	6
7	Sep 29 Oct 1	Runtime memory management The runtime stack and stack frames Programs, procedures, and functions Procedure and function calls	7

Week	Dates	Topics	Readings
8	Oct 6 Oct 8	<b>Midterm exam Tuesday, October 6</b>  A language converter: Pascal to C++ Structured data: arrays and records <b>Lab: Language converter</b>	
9	Oct 13 Oct 15	The Java Virtual Machine (JVM) architecture Jasmin assembly language Code templates and code generation Code for expressions	
10	Oct 20 Oct 22	Code for assignment statements Code for control statements Code for procedure and function calls <b>Lab: Code generation</b>	
11	Oct 27 Oct 29	Code to call <code>printf()</code> Code for arrays and records Code to pass parameters by value and by reference Runtime libraries	
12	Nov 3 Nov 5	Compiled vs. interpreted code Context-free vs. context-sensitive grammars Bottom-up parsing with yacc and lex Code optimization	
13	Nov 10 Nov 12	Runtime memory management, <i>cont'd</i> Garbage collection algorithms A simple source-level debugger	
14	Nov 24	<i>(TBD)</i>	
15	Dec 1 Dec 3	Project presentations	
<b>Final Exam</b>	<b>Wednesday, Dec 9</b>	Time: 7:15 - 9:30 <b>AM</b> online	