


Contents

1 Course Information	2
1.1 Website	2
1.2 Catalog Description	2
1.3 Course Staff	2
1.4 Class	2
1.5 Recommended Text	2
1.6 Grading Scheme	2
1.6.1 Assessments	2
1.6.2 % Score to Letter Grade	3
1.7 Software Needed	3
1.7.1 Piazza	3
1.7.2 Gradescope	3
1.7.3 Programming Environment	3
1.7.4 Zoom	3
1.8 CS Account	3
1.9 Policies	3
1.9.1 Classroom	3
1.9.2 Piazza	4
1.9.3 Makeup Exam	4
1.9.4 Regrade Request	4
1.9.5 Collaboration	4
1.9.6 Accomodations for Students with Disabilities	4
1.9.7 Campus Closure	4
2 Topics Covered	4
3 Assignments	5
3.1 The List	5
3.2 Submitting Your Work	6
3.3 How the Assignments will be Scored	6
3.3.1 Correctness (80%)	6
3.3.2 Code Clarity and Efficiency (10%)	6
3.3.3 Notes File (10%)	6

1 Course Information

1.1 Website

<https://www.swamiiyer.net/cs451/> 


1.2 Catalog Description

Introduction to compiler organization and implementation, including formal specifications and algorithms for lexical and syntactic analysis, internal representation of the source program, semantic analysis, run-time environment issues and code generation. Students will write a compiler for a reasonably large subset of a contemporary language, targeted to a virtual machine.


Prerequisites: CS310  and CS420  or CS622 ; or permission of the instructor.

Students who successfully complete this course will be able to: write parsers and produce an abstract syntax tree (AST); analyze and generate code for a programming construct represented by an AST; and allocate physical registers (a limited resource) to a program expressed in terms of virtual registers (an unlimited resource).


1.3 Course Staff

Swami Iyer  will be the primary instructor for the course. He will be assisted by a graduate teaching assistant (TA).

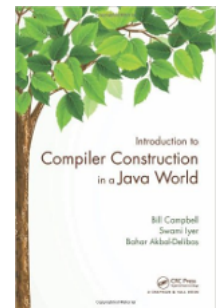
1.4 Class

In each class, the instructor will present the material  for that class. Roughly once a week, the instructor will also conduct an online quiz towards the end of a class on recently covered material.

1.5 Recommended Text

Introduction to Compiler Construction in a Java World by Bill Campbell, Swami Iyer, and Bahar Akbal-Delibas 

This text enables a deep understanding of the Java programming language and its implementation. It covers all of the standard compiler topics, including lexical analysis, parsing, abstract syntax trees, semantic analysis, code generation, and register allocation.



1.6 Grading Scheme

1.6.1 Assessments

Item	% of Final Grade
Programming Assignments (6)	42
Exams (2)	48
Participation	10

- The goal of the programming assignments is to make sure that you can apply the concepts learned in class to enhance the functionality of the base *j--* and *iota* compilers.
- The closed-book exams will test your understanding of the theoretical concepts covered in class.

- Your participation score will be based on weekly in-class quizzes. Each quiz, conducted at the end of a class, will test your understanding of the material covered recently. Each question in a quiz is worth 1 point. Each quiz score will be normalized to 100 points. Only your best 10 quiz scores will count towards the final grade.
- You can earn 0.01x% extra points if x% of the class completes the end-of-semester course evaluation.
- If your overall score falls within half a percent of a higher grade, your score will be elevated to that grade.

1.6.2 % Score to Letter Grade

[93, 100]: A, [90, 93): A-, [87, 90): B+, [83, 87): B, [80, 83): B-, [77, 80): C+, [73, 77): C, [70, 73): C-, [67, 70): D+, [63, 67): D, [60, 63): D-, [0, 60): F

1.7 Software Needed

1.7.1 Piazza

We will use Piazza [↗](#) as the Q&A platform for the course. If you have any general questions about the assignments, exams, or the lecture material, the most effective way to get them answered is by posting them on Piazza. You can expect your questions to be answered by the course staff or your peers.

1.7.2 Gradescope

We will use Gradescope [↗](#) to grade your programming assignments and exams and for the in-class quizzes.

1.7.3 Programming Environment

To write and execute Java programs in this course, you will need a laptop (Linux, Mac, or Windows) properly configured with the necessary software. Click here [↗](#) for setup instructions.

1.7.4 Zoom

We will use Zoom [↗](#) to hold remote office hours.

1.8 CS Account

In order to use the computing resources of the department, and in particular, those in the UNIX/PC Lab (M-3-0731), you need to setup a CS account. With your CS account credentials, you can connect to our designated server (`users.cs.umb.edu`) remotely using SSH. With the same credentials, you can also sign into the Linux systems in the CS Lab. In addition, you can sign into the Windows systems in the lab with the same username and an initial password `abcd_1234`, which you must change the first time you sign in.

Visit CS Labs Portal [↗](#) to register for a portal/CS account and confirm via email. If you already have a CS account, use the same username. The next step is to sign into the portal and select your courses for the term. You will be notified via your UMB email once the course directories and your account are created.

1.9 Policies

1.9.1 Classroom

Come to class on time and stay for the entire session. If you have to leave early, let the instructor know in advance. Have your mobile phone silenced or turned off during the entire session. Use of earphone/headphone during the session is not permitted. Use of a laptop during the session is permitted only for class purposes. Do not talk to each other during the session. If you have any questions, bring them up to the instructor.

1.9.2 Piazza

If you have a question, first make sure that it has not already been asked/answered. Clearer questions get better answers, so re-read your question before you post it. Ask your questions early. Posts are categorized using channels, so pick an appropriate tag for your post. Use the platform only for questions that can be asked in a general way, without sharing code or other assignment-related work. However, if you are stuck on a problem despite your valiant efforts to solve it, you may seek help from the course staff by posting your code privately, as properly formatted text (not images). Any post that is inappropriate or violates the academic honesty code will be deleted by the course staff.

1.9.3 Makeup Exam

You must provide appropriate documentation if you were/are unable to take an exam on the scheduled date and want to arrange a makeup exam. The documentation must be a letter from the Dean of Students [↗](#) if the type of your absence is among those listed on their website. For other types of absences, the supporting documentation must be emailed to the instructor directly.

Note: There will be no makeup of missed quizzes.

1.9.4 Regrade Request

If you have any concerns about the grading of a particular assignment or exam, you may submit a regrade request [↗](#) via Gradescope. You must submit the request within a week from the date the assignment or exam grades are published, or else your request will be turned down.

1.9.5 Collaboration

Click here [↗](#) for the collaboration policy and the penalties for infractions of the policy.

1.9.6 Accommodations for Students with Disabilities

Section 504 of the Americans with Disabilities Act of 1990 offers guidelines for curriculum modifications and adaptations for students with documented disabilities. If applicable, students may obtain adaptation recommendations from the Ross Center for Disability Services [↗](#). The student must present these recommendations and discuss them with the instructor within a reasonable period, preferably by the end of Add/Drop period.

1.9.7 Campus Closure

In the event of a campus closure, all class-related activities will be conducted remotely, via Zoom. If there is an exam scheduled to take place on that day, the exam will be postponed to the next suitable date.

2 Topics Covered

- Course Mechanics [*Lecture 1*]
- Programming Environment [*Lecture 1*]
- Chapter 1: Compilation
 - Preliminaries [*Lecture 2*]
 - Overview of the *j--* to JVM Compiler [*Lecture 3 and 4*]
- Chapter 2: Scanning
 - Preliminaries [*Lecture 5*]
 - Handcrafting a Scanner [*Lecture 6*]
 - Generating a Scanner [*Lecture 7*]

- JavaCC Scanner for *j--* [Lecture 8]
- Chapter 3: Parsing
 - Preliminaries [Lecture 9]
 - Top-down Recursive Descent Parsing [Lecture 10]
 - Top-down LL(1) Parsing [Lecture 11]
 - Bottom-up LR(1) Parsing [Lecture 12]
 - JavaCC Parser for *j--* [Lecture 13]
- Chapter 4: Type Checking
 - Preliminaries [Lecture 14]
 - Pre-analysis of *j--* Programs [Lecture 15 and 16]
 - Analysis of *j--* Programs [Lecture 17 and 18]
- Chapter 5: JVM Code Generation
 - Preliminaries [Lecture 19]
 - Classes and their Members [Lecture 20]
 - Control, Message, Field Selection, and Array Access Expressions [Lecture 21]
 - Assignment, String Concatenation, Cast, and Other Operations [Lecture 22]
- Chapter 6: The *iota* Compiler
 - Preliminaries [Lecture 23]
 - High-level Intermediate Representation (HIR) [Lecture 24]
 - Low-level Intermediate Representation (LIR) [Lecture 25]
 - Register Allocation [Lecture 26 and 27]

3 Assignments

3.1 The List

There are 6 programming assignments in all. These are due at midnight (11:59 PM to be precise) on the dates indicated on the Calendar [📅](#) page of the course website.

#	Title	Goal
1	Supporting Simple Operations	Become familiar with <i>j--</i> and the Java Virtual Machine (JVM); and extend the <i>j--</i> language by adding support for some arithmetic operators, conditional expression, and do statement.
2	Scanning	Modify the handcrafted scanner to support multiline comments; additional tokens (reserved words and operators); and <code>long</code> and <code>double</code> literals in <i>j--</i> .
3	Parsing	Modify the handcrafted parser to support <code>long</code> and <code>double</code> basic types; additional operators; and <code>for</code> , <code>break</code> , <code>continue</code> , and <code>switch</code> statements in <i>j--</i> .
4	Scanning and Parsing with JavaCC	Modify the <code>j--.jj</code> file used for generating a scanner and parser (using JavaCC) for <i>j--</i> to support multiline comments; <code>long</code> and <code>double</code> basic types; additional tokens (reserved words and operators); conditional expression; and <code>do</code> , <code>for</code> , <code>break</code> , <code>continue</code> , and <code>switch</code> statements.
5	Type Checking and Code Generation	Implement type checking and JVM code generation for the programming constructs that were added to <i>j--</i> in Assignment 3 (Parsing).

6 Register Allocation Implement the graph coloring based register allocation algorithm in *iota*.

3.2 Submitting Your Work

You will use Gradescope [↗](#) to submit your Java programs (ie, `.java` files) and the `notes.txt` file. Make sure that you only submit files listed under the “Files to Submit” section of the assignment writeup.

You may submit your files as many times as you like, up until the assignment deadline. The most recent submission is considered active by default and your score on the active submission is your official score for the assignment as well. You have the option of making any of your previous submissions active.

Note: If your active submission is partial, your assignment score will also be partial, so in order to be eligible for full credit, make sure you have an active submission containing all the required files for the assignment.

3.3 How the Assignments will be Scored

3.3.1 Correctness (80%)

Your solution to each assignment problem will be evaluated for correctness by an autograder. Each test that is used for this purpose is worth some number of points; your solution will receive all the points from a test that passes and 0 points from a test that does not.

3.3.2 Code Clarity and Efficiency (10%)

Your solutions will additionally be checked by a TA for code clarity and efficiency. Your solution to each problem will receive some number of points if it passes all the autograder tests for that problem and 0 otherwise. In addition, your solution will receive some number of points if it follows good programming principles (ie, is clean, well-organized, uses meaningful variable names, includes useful comments, and is efficient) and will be marked down otherwise.

3.3.3 Notes File (10%)

The given `notes.txt` file for an assignment must be uploaded with the three sections (`#1` mandatory, `#2` if applicable, and `#3` optional) filled in as appropriate. In section `#1`, for each problem, you must state its goal in your own words and describe your approach to solve the problem along with any issues you encountered and if/how you managed to solve those issues. For each problem, your notes will receive some number of points if the goal and approach subsections meet our expectations and will be marked down otherwise.