1 Introduction

Welcome to ECE351! This is an introductory compilers course. The calendar description lists a number of topics:

- Programming paradigms, compilation, interpretation, virtual machines.

We will cover many of the topics listed above. We will do so by following the book “Crafting a Compiler” (ISBN 13: 978-0-13-606705-4), and lecture notes that I provide. Please purchase a copy asap. For the lab part of the course, a manual authored by Professor Derek Rayside will be used. It is available on the course website.

1.1 Course Website

All relevant course materials will be available from the course website linked from https://ece.uwaterloo.ca/~vganesh/teaching.html

1.2 Course Objective

The objectives of the course are two-fold: First, to teach students the theoretical aspects of compilers and language design, and second, to enable students to build a practical rudimentary compiler of their own. The course is structured as follows: We will have one lecture every week that focuses on theory, and another lecture that focuses on a mix of theory and lab.

This course is very hands on, i.e., you will write a circuit synthesizer and a circuit simulator for circuits described in a subset of VHDL. A circuit synthesizer is essentially a compiler for circuits. The lab notes describe these exercises in detail.

Through the lab you will learn two important skills: First, you will understand how language design can be seen as a general form of problem solving. Second, you will learn how to “implement” a language by constructing an appropriate compiler.

Often language design and problem solving are seen as distinct activities. The point of the lab is to show that language design, in many ways, is the most general way to solve a problem. By most general
solution I mean that your solution solves an entire class of problems, not merely one problem or an instance of a problem. For example, suppose that you need to multiply two matrices. The least general, and most direct, thing you could do is to actually multiply the matrices. More generally, you could write a program that multiplies two arbitrary matrices, and then use that program to multiply your two matrices of interest. Most generally, you could design a language that not only allows you to write programs to multiply matrices but also allows you to express programs to do more complex mathematical operations. Your language could be something similar to a suitable subset of Matlab, then write a program in that language that requests your two matrices be multiplied, then execute that program with the interpreter/compiler you just developed. In order to solve problems at this level of abstraction you need to understand the ideas listed in the course description, and you will need to improve your programming skills.

We will use the term transformer or translator to describe a program that reads and writes structured text. As you will see, any sufficiently large transformer is actually comprised of many small transformers: the transformation is broken down into steps that typically pass through one or more intermediate forms before the final output is produced. These intermediate forms are typically simpler to process mechanically and less convenient for human use than the original input language.

Our **VHDL** transformers will make use of a variety of intermediate forms. In the first half of the term we will write transformers just for these intermediate languages. Only the second half of the term will we process **VHDL** input. The lab notes give a detailed schedule of the weekly labs, as well as a summary of some necessary computer skills.

## 2 Instructor Office and Contact, Classroom and Lab Location

Below find some logistical details about the course. TA and Lab instructor details are on the course website.

<table>
<thead>
<tr>
<th>Instructor:</th>
<th>Vijay Ganesh</th>
<th>DC2530</th>
<th><a href="mailto:vganes@uwaterloo.ca">vganes@uwaterloo.ca</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Midterm:</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lab Due Date:</td>
<td>Fridays 11:59PM</td>
<td>Git</td>
<td>every week (except midterm week)</td>
</tr>
<tr>
<td>Lecture Time/Place:</td>
<td>MF 10 AM</td>
<td>QNC 1502</td>
<td>See Schedule</td>
</tr>
<tr>
<td>Lab Time/Place:</td>
<td>TWFh 3 PM</td>
<td>E2-2363</td>
<td></td>
</tr>
<tr>
<td>Files:</td>
<td>Git</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discussion:</td>
<td>Piazza</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Files will be distributed and collected through Git. Each student will have their own private repository. We will have shared repositories for skeleton code and test cases. Your Nexus/Waterloo credentials (username and password) should give you appropriate access to the repositories.
3 Collaboration

Interaction is an essential part of learning for most people. You will work in at most 2-person teams, for both labs and assignments. Declare your team structure in all your labs and assignments. Once your team structure is defined at the beginning of the course, you will not be able to change without explicit permission from the instructor.

4 Reference Material

The textbook for the course is Crafting a Compiler. Past offerings of this course have recommended Modern Compiler Implementation in Java or Programming Language Pragmatics. Those are both good books, but we think Crafting a Compiler is a better fit for this course. All three books are available at the library. The intellectual content is widely available on the internet. In the past some students have reported that they have found the internet to be a big place and they have had some difficulty figuring out which parts of it are relevant to the course. Perhaps your searching skills are better. The tools that we will use are documented largely online.¹

5 Marking Scheme

<table>
<thead>
<tr>
<th></th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labs</td>
<td>33</td>
</tr>
<tr>
<td>Assignments</td>
<td>9</td>
</tr>
<tr>
<td>Mid-term</td>
<td>8</td>
</tr>
<tr>
<td>Final Exam</td>
<td>50</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>

a. You must pass the final exam in order to pass the course.

b. All labs are weighted equally; thus, each contributes to 3% of your final lab mark. (Note that there are 12 labs, but lab 0 is not graded.)

c. You must earn at least 50% of the available lab marks in order to pass the course.

d. Late policy: You can submit up to 1 lab and 1 assignment late without penalty. Any additional late submissions will be awarded 0 points, unless otherwise authorized by the instructor.

6 Classroom Style

The projector will be used to display the textbook, the course notes, slides, the lab manual, and the code. The board will be used to primarily to work through problems. All problems will be taken from

¹parboiled.org

Speak up if you do not understand what is being discussed in class. Chances are what you do not understand is some background material that you weren't actually taught in your previous courses.
a text book or the course notes, so you will not have to write them down yourself.
7 University Policies

Academic Integrity: In order to maintain a culture of academic integrity, members of the University of Waterloo community are expected to promote honesty, trust, fairness, respect and responsibility.

http://uwaterloo.ca/academicintegrity/

Grievance: A student who believes that a decision affecting some aspect of his/her university life has been unfair or unreasonable may have grounds for initiating a grievance. When in doubt please be certain to contact the department’s administrative assistant who will provide further assistance.

http://uwaterloo.ca/academicintegrity/

Policy 70, Student Petitions and Grievances, §4, http://secretariat.uwaterloo.ca/Policies/policy70.htm

Discipline: A student is expected to know what constitutes academic integrity to avoid committing an academic offence, and to take responsibility for his/her actions. A student who is unsure whether an action constitutes an offence, or who needs help in learning how to avoid offences (e.g., plagiarism, cheating) or about rules for group work/collaboration should seek guidance from the course instructor, academic advisor, or the undergraduate Associate Dean.

http://uwaterloo.ca/academicintegrity/

For information on categories of offences and types of penalties, students should refer to Policy 71, Student Discipline, http://secretariat.uwaterloo.ca/Policies/policy71.htm

For typical penalties check Guidelines for the Assessment of Penalties, http://secretariat.uwaterloo.ca/guidelines/penaltyguidelines.htm

Appeals: A decision made or penalty imposed under Policy 70 (Student Petitions and Grievances) (other than a petition) or Policy 71 (Student Discipline) may be appealed if there is a ground. A student who believes he/she has a ground for an appeal should refer to Policy 72 (Student Appeals).

http://secretariat.uwaterloo.ca/Policies/policy70.htm

http://secretariat.uwaterloo.ca/Policies/policy71.htm

http://secretariat.uwaterloo.ca/Policies/policy72.htm

Note for Students with Disabilities: The Office for Persons with Disabilities (OPD) collaborates with all academic departments to arrange appropriate accommodations for students with disabilities without compromising the academic integrity of the curriculum. If you require academic accommodations to lessen the impact of your disability, please register with the OPD at the beginning of each academic term.

OPD: Needles Hall, Room 1132

e.g., TurnItIn.com

Plagiarism Detection Software: The instructors may, at their discretion, use plagiarism detection software.