Course Description

Techniques and tools for construction of compiler and interpreter back-ends, including: interpreter design; code generation strategies for standard programming constructs; intermediate representations; optimization techniques; run-time organization, including functions, objects, and closures; run-time systems. Design and implementation of an interpreter and a compiler back-end for a small programming language.

Prerequisites

This class requires CS 201, 202, 300, 311, and 320; passed with grades of C or better. These courses provide you with the background that you will need to understand the course material and develop solutions to the assignments that you will be given. Although it is not formally listed as a prerequisite, it is also assumed that you have experience programming in a high-level language such as C, C++, Java, etc. These skills are essential because this course is centered around substantial programming projects such as implementing components of a compiler for a realistic language.

If you have not completed the prerequisites listed above, then you may not have the background that you need to pass this class. In this case, by default, you will not be able to continue in the class and you should contact the instructor at the earliest opportunity if you can demonstrate that you do, in fact, have the necessary background to be considered for an exception and avoid an administrative drop.

Students interested in the topic of programming language implementation are strongly encouraged (but not required) to take CS 422 as a follow-on to CS 421, Programming Language Implementation: Syntax and Static Semantics. These two courses are designed to complement one another and, together, provide comprehensive coverage of compiler and interpreter construction from raw source code to executable programs.

Course Objectives/Student Learning Outcomes

Upon the successful completion of this course students will be able to:

1. Implement an interpreter for a simple but realistic language.
2. Describe and apply the basic concepts of sequential control abstraction, including structured control constructs and subroutines.
3. Explain runtime organization of program execution, including activation records, static and dynamic access links, and frame and stack pointers.
4. Explain different parameter-passing modes and their implementation implications.
5. Describe standard runtime representations for arrays, records, objects, and first-class functions.
6. Describe the general goals and limitations of compiler optimization.
7. Give examples of peephole, local, and global optimizations, and explain the general techniques involved in each of them.
8. Explain the role of an intermediate representation in an optimizing compiler.
9. Explain the concept of garbage collection and the typical algorithms.
10. Implement a code generator for a simple but realistic language, producing native code for a real machine.

Outline of Course Content

The course will be taught using one lecture and one instructor-led lab session for each week of the term. Lab sessions will be used to provide opportunities for more hands-on exercises and experimentation, and will be used both to introduce new material and to reinforce topics presented in lectures.

<table>
<thead>
<tr>
<th>Week</th>
<th>Lecture Topic</th>
<th>Lab Topic</th>
<th>Assignments (weighting)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interpreters</td>
<td>The &quot;Interpretour&quot;</td>
<td>Complete the &quot;Interpretour&quot;</td>
</tr>
<tr>
<td>2</td>
<td>Introduction to Code Generation</td>
<td>Code Generation Lab</td>
<td>Interpreter (10) assigned</td>
</tr>
<tr>
<td>3</td>
<td>Assembly Code and Assembly Code Generation</td>
<td>Assembly Code Programming</td>
<td>Interpreter in progress</td>
</tr>
<tr>
<td>4</td>
<td>Runtime Organization and Functions</td>
<td>Stack Handling</td>
<td>Interpreter due, CodeGen (10) assigned</td>
</tr>
<tr>
<td>5</td>
<td>Objects and First-class Functions</td>
<td>Midterm preparation</td>
<td>CodeGen in progress, review for midterm</td>
</tr>
<tr>
<td>6</td>
<td>Midterm exam (25)</td>
<td>Closures and objects lab</td>
<td>CodeGen due, RTO (15) assigned</td>
</tr>
<tr>
<td>7</td>
<td>Intermediate Representations</td>
<td>IR Code Generation</td>
<td>RTO in progress</td>
</tr>
<tr>
<td>8</td>
<td>Optimization techniques</td>
<td>Optimization and IR hands on</td>
<td>RTO due, IR (10) assigned</td>
</tr>
<tr>
<td>9</td>
<td>Runtime systems and garbage collection</td>
<td>Garbage collection lab</td>
<td>IR in progress</td>
</tr>
<tr>
<td>10</td>
<td>Advanced topics</td>
<td>Course review / Final preparation</td>
<td>IR due (end of week), review for final</td>
</tr>
<tr>
<td>11</td>
<td>Final exam (30)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Course Requirements and Method of Evaluation

There will be five project/homework assignments during the term. Details of these assignments, including content and submission deadlines, will typically be made available during or after the lecture or lab sessions in the weeks indicated by the schedule above. Most of these assignments will involve significant programming work that you will be required to complete using the Java programming language (or other languages, as appropriate). There will be a minimum passing grade on each assignment, and students must pass every assignment to avoid receiving an F or X grade.

Most of the project/homework assignment will be designed to be worked on over a period of several weeks. A student who delays starting or working on a project will likely find: that they do not have enough time to ask clarifying questions; that they may not be able to complete the programming part of the assignment; and that they may struggle with material that is covered in lectures and labs after the assignment has been announced. We reserve the right to require submission of work at intermediate milestones, ahead of the original deadline (but with at least six days notice), if we feel that this is necessary to ensure that students are staying on track.
There will be one mid-term exam and one final exam, both of which will be closed-book. Exams will cover topics from lectures, emphasizing material that is not directly covered by or relevant to programming assignments, so lecture attendance is important. Exams are scheduled in advance and, unless a prior arrangement is made, a grade of zero will be recorded for missed exams.

There may be some assignments that are not formally assessed. This might include written exercises, additional reading, or watching prepared videos, etc. These will be designed to help you master the subject, to keep pace with the lectures, and to prepare for the exams, so it will be very important for you to keep up to date with those exercises.

The weightings that will determine your final grade are as follows: graded projects/assignments: 45%; midterm: 25%; final: 30%.

Required Texts and/or Required Reading List

The slides that are presented in lectures and all of the materials and notes that are provided in labs are an essential part of the course and are required reading.

There is no required textbook for this class, but all students are strongly encouraged to do background reading in parallel with our coverage of topics in the class. There are lots of good textbooks on this subject and we list some of our favorites below. However, we do not expect you to buy or have access to any particular text: for example, we will not set exercises that come from a textbook, or require you to have read specific sections of any particular book. So if you find a compiler text in the library, or borrow one from a friend, that will probably still be a good choice for you.

Of course, there are also lots of good resources on the Internet. In particular, we mention "Basics of Compiler Design" by Torben Mogensen, which is available for free in pdf form from http://www.diku.dk/~torbenm/Basics/. You can also find lots of good information on the topics of these classes in places like Wikipedia, or simply by searching on Google.

If you are really looking for a hard copy book to add to your library, our personal favorites (in no particular order) are as follows:


Computing Facilities
This course will be taught using the Java programming language (JDK Version 1.6 or later) in combination with x86-64 assembly language. In particular, the course will include significant programming assignments that require the use of these languages. We will ensure that all of the software that is needed to complete these assignments is installed and tested on the Linuxlab computers, which can be used directly in the lab or accessed remotely via ssh/PuTTY, for those who are not on campus. Assignments may also be completed using other machines/operating systems, but you are responsible for ensuring that the necessary tools are installed and functioning correctly. However, if you want assistance from the course staff, then you are strongly encouraged to do your development on Unix or Linux because help for Windows may be limited.

**Academic Integrity**

We follow the standard guidelines for academic integrity. It is permissible to discuss assignments with other students, but you must develop the solution yourself (although you can consult the tutors for help in debugging). *Do not, under any circumstances, copy any part of another person's solution and submit it as your own.* Writing code for use by another, or using another's code in any form (even with their permission) will be considered cheating. Cheating on an assignment or exam will result in an automatic zero grade for that piece of work, and the initiation of disciplinary action at the University level. Please refer to [http://www.pdx.edu/dos/codeofconduct](http://www.pdx.edu/dos/codeofconduct) for details of the general PSU Student Code of Conduct.

**Disabilities**

If you are a student with a documented disability who is registered with the Disability Resource Center, please contact the instructors immediately to arrange any needed academic accommodations, and let us know as soon as possible if you feel that your needs are not being met. If you have accommodations that include taking tests at the University test center, you should take steps to make the necessary reservations at the earliest possible opportunities; the times and dates for both the midterm and final exams are already fixed for 5/4 and 6/8, respectively, as described above.

**Subjective Matters**

To ensure consistency, all questions about matters that are potentially subjective—such as possible grading errors or determining whether an exceptional circumstance warrants an extension of a deadline—should be addressed to the primary instructor for this class.