ECE466 Compilers Syllabus -- Spring 2021 -- Prof. Jeff Hakner

Special COVID Syllabus

Overview

This is a graduate-level course in Compilers. Although the title of the course is "compilers" we will also be discussing interpreters and related concepts. In many CS curricula, much of the material that I present in this course would actually be in a 300-level undergrad course, and some of the material might be in a followup grad-level course. In our curriculum, we only have this one course in compilers. Therefore, ECE466 must serve as an introduction to compilers, as well as a taste of graduate-level topics in compilers.

Student Expectations

Because this is being run at the graduate level, it is important for students to understand the different set of expectations. There is very little "hand-holding" in this course. Students are expected to do a substantial amount of independent learning. The "lecture notes" do not comprise the entirety of the material for which the student is responsible.

This is a capstone project-based course in which we'll be (collectively, yet individually) building a rudimentary compiler. Therefore the individual homework assignments are really mileposts in the journey to project completion. It should also become apparent that each assignment builds upon the previous assignments.

Therefore I make this critical recommendation: **Do not fall behind!** The complexity of the work (both intellectual complexity and technical complexity) increases as we go forward. A delay of say a single week in completing the first assignment might seem inconsequential in a 15 week semester, but it will put you in the bad position of being unable to start on the second assignment while the rest of the class is moving ahead. And so on, as we move through the project.

I would also expect that students read the chapter material ahead of the lectures. You will be in a better position to ask questions during lecture if you are already somewhat familiar with what is being discussed.

Online Class Code of Conduct

Normally, I run a Luddite classroom with a strict prohibition of "devices" being used. Obviously, that is all out the window this year! I propose the following ground rules to maximize the effectiveness of our online classes:

* To allow everyone time to join the TEAMS meeting, fiddle with their audio and video settings, etc. I will be starting the online class at 15 minutes past the hour, i.e. 6:15 PM. Of course, you are free to join in earlier, and I will be there too, but I won't transact any business until that time.

* Try to be in an environment that is free from background noise. If you have a noisy background, or you need to momentarily create noise such as typing, please mute your

microphone. Remember to un-mute when you need to talk! (keyboard shortcut: Alt-Shift-M)

* Please have your video ON generally throughout the class. Of course, if you need to step away briefly that is fine, just as you might need to briefly leave a physical classroom.

* I am required to confirm "participation" of all students in online classes, in lieu of physical attendance observations. If you are going to be unable to attend the live online lectures, please document that to me.

Major Objectives

In ECE466, we have the following broad learning objectives:

* How a compiler and interpreter work

* Using specialized compiler programming tools such as Flex and Bison

* Gaining a more detailed understand of a sample language (we'll be using C)

* Assembly Language

* How the linker/loader and other OS-dependent tools work with the compiler

Office Hours / Contact

The best way to reach me with a question is via email to `hak@cooper.edu`. I will try to answer student email questions within 12 hours, including weekends. If we need to, we can schedule a one-on-one TEAMS meeting.

Find lecture notes, assignments, and supporting material at
`http://faculty.cooper.edu/hak/ece466/`

Grading & Group Work

In a graduate course, it is very unusual for grades other than A or B to be given out. I will base the grade primarily on your submitted project work. I will also have some additional take-home test/quiz assessment points. I will schedule individual meetings with each student at two times during the semester, to discuss project progress and to assess engagement and comprehension of the course.

Therefore, in the context of ECE466, "A" will be earned if the project is completed successfully and all other assessments are satisfactory. "B" will be given out for partial project completion. "C" is reserved for a disaster such as minimal progress on the project, or significant problems with the assessments.

Normally I insist on each student working individually. As a concession to the COVID learning environment, I will permit two-person group work. You may also work alone. Consider your partner selection very carefully as it will be binding for the entirety of the course.

Textbook

There is no required textbook for this class. I suggest a number of useful references on the course website.

Aho, Alfred V., Lam, Monica S., Sethi, Ravi and Ullman, Jeffrey D.: Compilers: Principles, Techniques, and Tools.. 2nd edition. Addison-Wesley, 2006.

The first edition of the textbook is also quite workable, especially for the earlier units of this course.

Students might find the O'Reilly book on Lex and Yacc (two tools which are used to automate the front-end of the compiler) helpful. However, much of the information in this book is dated and available for free online or in the man pages, which students should absolutely read thoroughly!

The `make` program will prove very useful in developing the compiler. There is an O'Reilly book on make, but again for the purposes of this project, there is enough free information available online.

Harbison, Samuel P and Steele, Guy L.: C: A reference manual. 5th edition. Prentice Hall, 2006.

This is basically a regurgitation of the ISO C standard, but with better examples and explanations, and a discussion of backwards compatibility issues with ANSI C (1989 standard) and classic K&R C.

On the course web site, I give links for various versions of the C language standard, and several assembly language references.

Course Outline

Below is the intended list of topics and the time allotted to each. This is subject to change depending on how the class progresses and scheduling issues.

```
Unit 1: Introduction to Compilers & Interpreters, Lexical Analysis
 (assignment: lexer)


Unit 2: Parsing (syntactical analysis)
 (assignment: first parser)

Unit 3: Error handling, memory allocation, other misc. issues

Unit 4: Semantic Analysis, symbol tables, type systems
 (assignments: declaration parsing/type representations, completing parser)


Unit 5: Intermediate Representations
 (assignment: generation of IR)
```

Unit 6: Architecture-Neutral Optimiziation

Unit 7: The Target Environment / Assembly Language

Unit 8: Target Code Generation
(assignment: completion of the compiler to assembly language)