Compiler Construction (201-1**2061**)
Fall 2012-2013
Syllabus
Last modified on October 23, 2012

Contents

To view this document

This document is written in the *Portable Document Format*. PDF is about more than just great looking documents: PDF documents may include hyperlinks (which all have boxes around them) that let you navigate both within the document as well as to places on the WWW. *This means that if you are looking at a printed version of this document, you are missing important links to software, documentation, and websites!* To view this document properly you should get the latest version of Adobe's Acrobat Reader, which is a free program for viewing PDF, and is available for a large number of platforms, including MS Windows, Apple Macintosh, Linux, and others. Throughout this course, we will be using PDF files to distribute information, homework assignments, examples of code, etc. Get the Acrobat Reader and get used to using it; It is a great tool.

1 Welcome

Welcome to Compiler Construction! This is a required third-year course in the computer science and software engineering programs, and is offered in the first semester of each year. Compiler Construction is not an easy course, but we the teaching staff think this is an exciting course in which you will learn many useful skills. If you apply yourself, then by the end of the semester you will have accomplished the following:

- You will gain additional insight into programming languages (building on what you have learned in the *Principles of Programming Languages* course you took previously.
- Understand the major components of the compiler: Syntactic analysis, semantic analysis, and code generation, and you will gain hands-on experience in crafting these components yourself.
- Learn about compiler optimizations: What compilers do to generate code that is faster, shorter, and performs better. We believe this is the most exciting part of the course. You will get a chance to implement many of these optimizations, and see for yourself how they improve the code.
- Be able to apply what you have learned about compilers to other areas in computer science where syntactic and semantic analysis, code generation, and translation are needed.

And you will have written a compiler!

2 Course Description

The course will cover the following topics:

- BACKGROUND: Some additional background material in micro-architecture, programming languages and Scheme, including functional and object-oriented abstraction, Continuation-Passing Style, the CPS-transformation, defunctionalization, the register & stack machine transformations, and threaded code.
- SCANNING: Regular expressions, DFA, NDFA, ϵ-moves. Hand-coding a scanner for Scheme.
- PARSING: CFGs, LL & LR parsing, and attribute grammars. Hand coding of various parsers, and using parser-generation tools in C and Java.
- MACRO EXPANSION: Syntactic transformations, reduction to basic forms in the language, variables, meta-variables and syntactic hygiene.
- INTERMEDIATE CODE GENERATION: Variables, procedures, scope, parameter-passing mechanisms, lexical addressing, semantic analysis, intermediate-level optimizations.
- LOW-LEVEL CODE GENERATION: Code generation for the basic forms, low-level optimizations, instruction scheduling, targeting the Intel x86 microarchitecture.

The following are related to compiler construction, and will be covered in brief:

- Garbage collection: Reference counting, stop-and-copy, mark-and-sweep, generational garbage collection

- The top-level: Value cells, function cells, property cells
- Run-time support: Special issues in the compilation of dynamic, interactive programming languages.

Of course, some minor variations are possible throughout the semester, but the above list describes our *intentions* rather faithfully.

3 Contact information

3.1 Teaching Staff

Your teaching staff for this course is:

Instructors		
Mayer Goldberg		
	Office Hours	Mondays, 12:00-14:00
	Office	Alon (37), Room #106
	Tel	08-64**7-7873**
	Email	gmayer@little-lisper.org
Roman Manevich		
	Office Hours	Tuesdays, 13:00-15:00
	Office	Alon (37), Room #312
	Tel	08-64**2-8009**
	Email	romanm@cs.bgu.ac.il
Teaching Assistants		
Avi Hayoun		
	Office Hours	Mondays, 16:00-18:00
	Office	Alon (37), Room #109
	Tel	08-64**2-8054**
	Email	hayounav@cs.bgu.ac.il
Vadim Levit		
	Office Hours	Wednesdays, 14:00-16:00
	Office	Alon (37), Room #214
	Tel	08-64**2-8039**
	Email	levitv@bgu.ac.il
Daniel Zatulovsky		
	Office Hours	Sundays, 16:00-18:00
	Office	Alon (37), Room #124
	Tel	08-64**TBA**
	Email	zatulovs@cs.bgu.ac.il
Graders		

3.2 How to Contact Us

For questions related to the course material or administration, please contact us via the course forum, so that other studens may see the answers. For personal issues, you may contact us by email. We are online several times during the day, on most weekdays, and will try to answer your email as soon as we can. You may, of course, call us at our office, or stop by. You are certainly welcome to stop by during our office hours. If you cannot make our office hours, send us an email and we will try to set up a time to meet with you at a more convenient time for you.

4 Lectures and Discussion Sections

The schedule of classes specifies two lectures per week, per section, for most weeks throughout the semester. While it is impossible to cover the exact material at the exact pace in both sections, the sections are synchronized by the end of the week; This means that despite any slight variations in the lectures in different sections, by the end of the week, all sections will have covered the same material.

You should attend each and every lecture. While attendance does not contribute directly to your final grade, you will find it very difficult to make up for lost material if you skip classes.

5 Course Work

The course work for compiler construction consists of

- Four homework assignments
- A mid-term quiz
- Final Project. Must have a passing grade (≥ 56) to pass the course.
- Frontal examination on the project. Must have a passing grade (≥ 56) to pass the course.
- Final examination (must have a passing grade (≥ 56) to pass the course)

The homework assignments will be in Python, Scheme, ML, C, Java, X86 assembly language, and some specialized software tools. The course work is mandatory. *You will need to score a grade of at least 1% on each assignment to be allowed to take the final exam.* Put otherwise, this means that if you ever get below 1% on an assignment, you will have failed this course. *Two or more assignments that contain substantial amounts of identical code will all get a grade of zero automatically, independently of any charges of academic dishonesty.* More on this later. We recognize that the homework is demanding, and we will do our utmost to coordinate your assignments with other third year course work, so as to balance out your workload.

We urge you to submit your homework on time. We will allow late submission of homework but your grade will be depreciated by 5% daily, for every late day (need not be a whole day). More on this later. Exceptions will be made only in cases of reserve duty (מילואים) or a serious medical problem that requires hospitalization longer than a day.

The assignments and final project in this course are to be done individually, but we will permit you to work in pairs if you so choose. If you work in larger groups, your grade will be reduced according to the following formula: A group of n students, for $n \geq 3$, will have their grade computed out of $200/(n-0.5)$. For example, if you work in a group of 3, the maximum grade of each partner will be 80. If you work in a group of 4, the maximum grade of each partner will be 57.

If you obtain help on an assignment other than from the teaching staff or from your partner, you are *committing academic* dishonesty. More on this later.

5.1 Course Material

For this course, we recommend that you acquire the book Modern Compiler Design, which is available at צומת ספרים. You will also benefit from reading Structure and Interpretation of Computer Programs (your PPL textbook), Compilers . We will frequently distribute notes and scanned pages from other sources; Watch for related announcements on the course website.

5.2 Software You Will Need

The assignments are designed so that they can be completed on Windows, Unix, Linux, or OSX, so you can complete them all on your own personal computers, without relying on departmental computing. During the semester, you may find it difficult to get access to departmental workstations during normal hours; If at all possible, try to work on your assignments from home.

For assignments done in Python, we recommend you use the latest version of Python 3.

For assignments done in Scheme, we recommend you use Petite Chez Scheme. This is the Scheme system that will be used to develop *our* solutions to the homework, quiz and exam problems, as well as grade your homework. If you must use another Scheme to develop your code, make sure that your code runs correctly under Chez Scheme.

For assignments done in ML, we recommend you use the Standard ML of New Jersey. This is the ML that will be used to develop *our* solutions to the homework, quiz and exam problems, as well as grade your homework. If you must use another ML to develop your code, make sure that your code runs correctly under SML/NJ.

For assignments done in Java, You will need a Java compiler and run-time support. You can download the standard JDK or use any other compatible JDK tool.

For assignments done in C and X86 assembly language, we will be using the GNU C compiler. No other compiler is acceptable. When compiling C, be sure to use the compilation flags `-Wall`, `-ansi`, `-strict`, and make sure to address any warnings or errors. Your code must be sufficiently portable so as to run on the university Linux machines. You can accomplish this under any OS, if you are careful to address any issues the compiler raises.

Please remember that while you may develop your code on any operating system you like, we will be testing and grading your code only on departmental Linux machines. Therefore, we cannot overemphasise how important it is that you reserve a couple of days prior to the submission, for testing and debugging your code on the departmental Linux machines. If your code will not run on the departmental Linux machines, then you will have failed the assignment regardless of how well your code may run on other platforms. Due to the large number of students enrolled in the course, and our very limited resources for grading, we cannot accept any appeals related to the platform you used for running your code: If your code does not run on the latest Python, Chez Scheme, SML/NJ, Java, and C, on our departmental Linux systems, then you will surely fail the assignment.

If your assignment fails to load/run/compile on the graders' machine, using the software that we recommend that you use, you will lose points liberally. *The graders will neither make corrections to your assignment or allow you to make corrections ipso facto. Be mindful, responsible, and test your work carefully.*

5.3 Late Work

The assignment is due by 13:00. For every late weekday (including weekends and holidays), you will be deducted a penalty of 5% (compounded). This penalty will apply in all but two situations, as specified by the Faculty of Natural Science:

- You are called for military reserve duty (מילואים) **for longer than a day**. You are required to submit a copy of your מילואים summon to the TA.
- You are suffering from a medical condition that required your hospitalization **for longer than a day**. In such case, you must provide suitable documentation by a qualified physician.

In the above two cases you are excused from submitting the assignment (i.e., you are given a complete exemption). No extensions are given.

In all other personal circumstances, you will need to decide how much late penalty you are willing to suffer.

Assignments build on previous ones, and after completing the assignments, you will be left with the final project of combining your code into a complete compiler. The final project is mandatory, i.e., you will not be able to pass the course without having passed it,[1] and no exemptions will be given for it, even if you did get exemptions for specific assignments. If you went to מילואים or were hospitalized, please submit the necessary documentation as soon as possible, and apply for an *extension*.

5.4 Grading Your Assignments

Each assignment will be graded for *correctness, quality,* and *conformity*. Correctness means that your code does what it is supposed to. Quality means that your code is written well. Conformity means that you named your files, classes, procedures correctly, that you followed the interface that we specified, that your code built cleanly, and in general, that testing your code did not require the grader to "fix" it in any way to get it to compile and run. Testing your assignment for correctness will be done through a combination of manual and automated testing. Quality will be tested manually. Conformity will be tested negatively.

Each question in each assignment will come with a *sample run*, demonstrating how we are going to run and test your code. If you name your procedures differently, or use a different number of arguments, or a different order of arguments, your code will fail our automatic testing mechanism, and you will lose points liberally.

5.4.1 The Frontal Exam

During the last week of classes, you will be asked to sign up for a frontal examination of your final project. You will need to pass the frontal exam with a grade of 56 or better, in order to pass the course.

5.5 Your Final Grade

Your final grade is essentially a *weighted average* of the following components:

Homework Assignments	15%
Project	10%
Frontal Exam	5%
Quiz	15%
Final Exam	55%
Total	100%

You must pass the final exam, and the final project in order to pass the course. This means a grade of 56 or better.

6 Academic Dishonesty

Academic dishonesty is defined in this class as any act of obtaining, soliciting or making available assignment-related or exam-related material in a way that is not explicitly permitted in the assignment or exam instructions. These include:

[1]Passing the final project means receiving a grade of 56 or better.

- Copying, soliciting or making available portions of a homework assignment, including from students who took the course in previous years, or from other courses taught in other institutions.
- Obtaining, soliciting or making available information related to the mid-term quiz or final examination, other than receiving your personal copy of either, from the teaching staff, during the exam.

If you commit any of the above, you're guilty of academic dishonesty. Neither we, nor the disciplinary board (ועדת משמעת) distinguish between dishonestly *taking* or *giving* information, and we will show no leniency towards students who share their work.

We will not tolerate academic dishonesty in this course. The penalty for academic dishonesty will be severe, and will begin with a zero grade in the given assignment (resulting in a failure in the course), a complaint filed with the disciplinary board (ועדת משמעת), and a detailed report to be placed in the student's permanent academic records. Similarly, we reserve the right to check for academic dishonesty *anytime* after you will have submitted an assignment, and re-assigning failing grades in cases where academic dishonesty has been established.

Please remember the following:

- This course is a required in the computer science and software engineering programs.
- You can only fail a course *twice* before you will be barred from taking it.

This means that failing a course on ground of academic dishonesty can have serious consequences as far as your ability to complete your degree. Do not cheat!