Wellesley College ◊ C301 Compiler Design ◊ Fall 2000

COURSE INFORMATION

Professor:	Franklyn Turbak (please call me "Lyn")
Office:	SCI 121B (behind the mini-focus consultant's desk)
Phone:	x3049
E-mail:	fturbak@wellesley.edu (or "Franklyn Turbak" in First Class)
Lectures:	SCI E111, Tuesday/Friday, 11:10pm – 12:20pm
Office Hours:	Tuesday 1:30pm – 4pm; Wednesday 8pm —10pm. Appointments can be made for other times. I sometimes may have to reschedule office hours to attend a meeting or talk. During a typical week this semester, I will spend all of Monday and Friday afternoons doing research, usually at Boston University, so these tend not to be good times to meet. I will also be unavailable on Thursday afternoons.

COURSE OVERVIEW

A compiler is a program that translates programs from one language to another, typically from a high-level language (such as ML or Java) to a low-level language (such as Pentium machine code or Java Virtual Machine byte codes). In this course, you will learn about the many stages of a compiler and the techniques and tools for implementing these stages.

There are many different dimensions of the course. Among the most important are:

- **Theory:** A significant corpus of theory has been developed to describe each of the many stages of compilation. Examples include automata theory for lexical analysis, context free grammars for parsing, type theory for type checking, and fixed point theory for data flow analysis. We will study the theory associated with the stages and learn about many standard algorithms that have been developed for them (e.g., scanning, parsing, type checking, instruction selection, register allocation).

- **Practice:** You will be getting significant hands-on experience with the stage of a compiler by implementing, largely from scratch, a sequence of compilers. We will start by implementing a simple compiler for a relatively simple source language, and then use more sophisticated techniques as we extend the source language with more complex features.

- **Tools:** A number of tools have been developed to simplify the process of constructing and testing compilers. We will be using two standard tools: ML-Lex, a lexical analyzer generator that simplifies the construction of a lexical scanner; and ML-Yacc, a parser generator that facilitates parser construction. We will also use the SPIM simulator for the MIPS computer architecture.

- **Programming Language Design:** As a an implementor of programming languages, you will gain new appreciation for various aspects of programming language design.

- **Software Engineering:** Compilers are infamous for their complexity. They tend to be very big programs constructed out of many parts. There are usually many ways to implement each part, with lots of tradeoffs involved (e.g. time vs. space, simple vs. complex). Furthermore, the compiler must handle a plethora of tiny details in order to correctly implement the source language.

All of this means that constructing a compiler is a significant software engineering project. In order to manage the complexity of the compiler, we will use a combination of standard tools and algorithms and good programming methodology based on modularity and abstraction. Our implementation language (ML) is particularly well-suited for supporting good programming methodology.

PREREQUISITES

The official prerequisites for CS301 are CS240, *Machine Organization*, and CS251, *Programming Languages*:

· From CS40, you are assumed to have familiarity with assembly language and low-level computer organization (binary representations of data and instructions, registers, stacks, memory, etc.). This knowledge will be helpful for understand code generation and other issues in the back end of the compiler. You are not assumed to be familiar with the MIPS assembly language we will be using this semester; that will be taught in class.

· From CS251 and/or previous programming experience, you are assumed to be familiar with functional programming. This will be helpful for programming in ML, which will be our implementation language this semester. Understanding various features of the programming languages we will be compiling (e.g., namespaces, scope of names, types, data structures, etc.) will also be important.

Although CS231 (*Algorithms*) and CS235 (*Languages and Automata*) are not official prerequisites, we will be making use of some of the ideas from these courses (e.g., automata, grammars, graph algorithms), which will be introduced on an as needed basis.

Also helpful are (1) significant programming experience and (2) experience in working with the Linux operating system. Don't worry: if you don't have these now, you will by the end of the course!

It's pretty clear from the above discussion that the compiler course depends on a *lot* of other material. For this reason, it is often referred to as a *capstone course* – one that nicely ties together the material from the rest of the computer science curriculum.

BOOKS, NOTES, & PAPERS

Textbook: The required textbook this semester is *Modern Compiler Implementation in ML (MCIML)* by Andrew Appel. (Make sure you get the hardcover edition that says "Reprinted with corrections, 1999" on page iv; previous editions had *lots* of errors.) We will be covering most of Part I of the book (Chapters 1 – 12). However, the "onion skin" approach I take in CS301 will require us to "jump around" these chapters a bit.

Warning: while this book has a much more modern approach to the material than other texts, it is often terse to the point of being incomprehensible. I will provide additional background and examples in class to clarify the material.

This book is available at the Wellesley bookstore. You might be able to find a better deal by purchasing a copy on-line.

Course Notes: I will hand out notes for some topics during the semester. However, I will not have notes on all topics I discuss in class, so I strongly recommend that you take detailed notes in class.

Papers: During the semester, we will read a few papers from the literature.

Recommended Books: The following books are not required, but are good references:

· *ML for the Working Programmer, 2nd edition (MLWP)* by Lawrence Paulson (Cambridge University Press, 1996). An excellent resource for learning how to do higher-order typed programming in ML. It contains many nice examples of typeful programming, higher-order functions, immutable data structures, and interpreters.

· *Compilers: Principles, Techniques, and Tools*, by Alfred Aho, Ravi Sethi, and Jeffrey Ullman (Addison-Wesley, 1986). Called the "dragon book" because of its cover. Although somewhat dated, this book is considered by many to be canonical compiler book.

· *Advanced Compiler Design and Implementation*, by Steven S. Muchnick (Morgan Kaufman, 1997). A relatively new book with encyclopedic coverage of many compiler topics.

Other Resources: There are a number of other resources that you may find handy during the semester:

· The CS301 home page (http://cs.wellesley.edu/~cs301) contains a link to a resources page that itself has links to on-line materials about ML, SPIM, Linux, and so on.

· The Computer Science Department Resource Room (SCI 173) has many books and journals that are relevant to this course.

· The Science Center Library houses many relevant books on compilers and programming languages. An easy way to find out what's available is to consult the on-line library catalog. To do this, execute `telnet library` from a Linux box.

· The MIT Laboratory for Computer Science has an excellent computer science library on the first floor of building NE43 (also known as "Tech Square") on the MIT campus. This is an especially good place to find journals and technical reports. For an on-line catalog, telnet to `reading-room.lcs.mit.edu`.

· The Barker Engineering Library at MIT (on the 5th floor of building 10, under the big dome) houses an extensive collection of computer science books. The on-line catalog is accessible by telneting to `library.mit.edu`.

COURSE DIRECTORY AND HOME PAGE

The CS301 course folder is located on nike.wellesley.edu in the directory /usr/users/cs301. This directory contains material relevant to the class, including course software, and on-line versions of lecture notes, assignments, and programs. From Netscape, all this information is available via links the following URL:

http://cs111.wellesley.edu/~cs301

The CS301 directory can be accessed via Linux FTP, Fetch, or Winsock-FTP by connecting to nike.wellesley.edu and navigating to /usr/users/cs301.

COURSE CONFERENCES

There is a CS301-F00 conference in First Class with two subconferences:

- *CS301-F00 Announcements* will be used to make class announcements, such as corrections to assignments and clarifications of material discussed in class.
- *CS301-F00 Q&A* is a forum for you to post questions or comments. They will be answered by me or a classmate. This is also a good place to find people to form a study group.

You should plan on reading the CS301 conferences on a regular basis.

HOMEWORK

There will be ten weekly problem sets during the semester. The main focus of each problem set will be part of a compiler implementation project, but there will sometimes be pencil and paper problems as well.

Most of the assignments will be rather challenging. Keep in mind that programming often consumes more time than you think it will. **Start your assignments early!** This will give you time to think about the problems and ask questions if you hit an impasse. Waiting until the last minute to begin an assignment is a recipe for disaster.

Because of the challenging nature of the assignments, you will be allowed to collaborate with a partner on all assignments, and turn in a single submission for the both of you. The grade received on such a submission will be given to both team members. Of course, it is expected that each member of the team will carry her weight. You do not have to work with a partner, but it may be wise to do so. I strongly recommend that you change partners throughout the semester to keep collaborations "fresh". I also recommend that if you did not take CS251 with me during Spring 2000 that you initially collaborate with someone who did.

You should submit both a "hardcopy" and "softcopy" of each assignment. Turn in the hardcopy under my office door by the specified time. You should submit the softcopy to the drop folder ~cs301/drop/psYY/ZZZ on Nike, where YY is the problem set number and ZZZ is your Nike username. If you work with a partner, you need turn in only a single harcopy and softcopy for the two of you.

PROBLEM SET HEADER SHEETS

I would like to get a sense for how much time it takes you to do your CS301 problem sets. Please keep track of the time you spend on each problem of your problem sets, and include this information on the problem set header sheets that I will provide at the end of each problem set. (If you work with a partner, each member of the team should turn in her own sheet.) Turn in this header sheet as the first page of your hardcopy submission.

LATE HOMEWORK POLICY

I realize that it is not always possible to turn in problem sets on time. On the other hand, turning in one problem set late can make it more difficult to turn in the next problem set on time. I have decided on the following policy for this course this term:

All problem sets will be due by 5pm on the specified due day (typically a Friday). A problem set can be turned in *n* days late if it is accompanied by *n* Lateness Coupons. If you work with a partner, each of you needs to attach one Lateness Coupon per person per day late.

At the end of this handout, you will find ten Lateness Coupons that you can use throughout the term. Use them wisely: you only get ten, and they are not copyable or transferable between students.

You may turn in late problem sets by slipping them under my office door. Of course, if I hand out solutions before you turn in a late problem set, you are bound by the Honor Code not to examine these solutions.

In extenuating circumstances (e.g.,, sickness, personal crisis, family problems), you may request an extension without penalty. Such extensions are more likely to be granted if they are made *before* the due date.

COLLABORATION POLICY

I believe that collaboration fosters a healthy and enjoyable educational environment. For this reason, I encourage you to talk with other students about the course and to form study groups. Because the programming assignments will be particularly challenging, I encourage you to work on your assignments in teams of two members, though you may work individually if you wish.

Each (one-person or two-person) team is required to **compose its own solution to each assignment**. In particular, while you may discuss strategies for approaching the programming assignments with other teams and may receive debugging help from them, **each team is required to write all of its own code**. It is **unacceptable** (1) to write a program with another team and turn in two copies of the same program or (2) to copy code written by other teams. However, it is OK to borrow code from the book and from materials handed out in class.

In keeping with the standards of the scientific community, **you must give credit where credit is due**. If you make use of an idea that was developed by (or jointly with) others, please reference them appropriately in your work. E.g., if person *X* gets a key idea for solving a problem from person *Y*, person *X*'s solution should begin with a note that says "I worked with *Y* on this problem" and should say "The main idea (due to *Y*) is ..." in the appropriate places. It is **unacceptable** for students to work together but not to acknowledge each other in their write-ups.

When working on homework problems, it is perfectly reasonable to consult public literature (books, articles, etc.) for hints, techniques, and even solutions. However, you must reference any sources that contribute to your solution.

COURSE GRADE

There are no exams in this course. The course grade will depend entirely on your problem sets and your class participation, as shown below:

Problem sets (total)	90%
Class Participation	10%

The default ranges for grades are expressed as a percentage of total points (excluding extra credit):

A	93.33 -- 100

A-	90 -- 93.32
B+	86.66 -- 89.99
B	83.33 -- 86.65
B-	80 -- 83.32
C+	76.66-79.99
C	73.33-76.65
C-	70 -- 73.32
D	60 -- 69.99
F	below 60

I reserve the right to lower boundaries between grades, but I will not raise them. This means that I can grade on a curve, but only in your favor.

The above information is intended to tell you how I grade. It is not intended to encourage a preoccupation with point accumulation. You should focus on learning the material; the grade will take care of itself. If you are dissatisfied with the grade you will receive based on the above scale, I encourage you to turn in extra credit problems to raise your grade.

EXTRA CREDIT

To make up for points lost on problem sets, students often request extra credit problems. In order to give everyone the same opportunity, I will sometimes include extra credit problems on the problem sets. The extra credit problems will often be more difficult than the other problems, but they provide the opportunity to earn extra points toward your course grade. You should only attempt extra credit problems after completing the assigned problems.

Extra credit problems are entirely optional. Extra credit points will only be factored into course grades after I have partitioned the grade scale into letter grades. Thus, doing the extra credit problems may raise your course grade, but not doing extra credit problems will not lower your course grade.

You may do extra credit problems either individually or with a single partner.

For maximum flexibility, **you may turn in extra credit problems at any time during the term** (through the last day of class). However, experience has shown that students who leave extra credit problems until the end of the term rarely turn them in. It is in your best interest to complete extra credit problems in a timely fashion. I will not hand out solutions to extra credit problems, but you are encouraged to discuss them with me in person.

PROGRAMMING/COMPUTERS

We will be using the ML programming language to implement compilers this semester. Those of you who took CS251 with me in Spring, 2000 have a little experience in ML; others probably do not. We will have extra tutorials on ML early in the semester to bring you up to speed.

You will also be learning MIPS assembly language programming this semester. MIPS assembly language will be the “target” language of the compilers you will write. We will use the SPIM simulator for simulating MIPS on our Pentium-based machines. (If time permits, we will also consider the Pentium as a compiler target.)

All programming will be done on the Linux workstations in the mini-mini focus. These machines are configured so that (1) you can log into them with your Nike username and password (2) you can access the Nike directory /usr/users “transparently” via /usr/users on any Linux machine. If you are not familiar with the Linux environment, browse the on-line Linux documenation, or ask me or a classmate for a tutorial.

You need not be sitting in front of a Linux machine console in order to work on your assignments. You can connect remotely to any of the department's Linux machines via a client like telnet or ssh and do your work remotely. The names of the ten Linux machines are: corfu, crete, cyprus, ithaca, kithera, lefkas, paxi, rhodes, santorini, and zante. (Alternatively, if you have a personal computer, you can install Linux and the course software on it.)

AFTER HOURS ACCESS TO THE SCIENCE CENTER

Many students prefer to do their programming at the Linux consoles in the mini-mini focus. An advantage of this location is that it is easier to collaborate with partners, and you are more likely to meet other students taking the course.

A disadvantage of working in the mini-mini focus is that it can be cumbersome to get into the Science Center after hours. If the building is closed but the Science library is open, you can get into the building by entering the library and taking the elevator from the ground floor.

If both the building and the library are closed, things are more complex. You must (1) go to the Sage door with a "buddy" and (2) call campus police from the Sage entrance. They will send over a squad car with an officer to let you and your "buddy" into the building. Campus police will not let you into the building without a buddy; this is the official Science Center policy.

SAVING WORK

Each CS301 student will have a password-protected account on the CS department file server (also known as nike.wellesley.edu and cs.wellesley.edu). You will have a limited amount of space on this server to store your course-related files.

You are also expected to keep copies of all your course work on floppy disks or zip disks. Removable disks are a frail medium that you should handle carefully. Store and transport them in suitable protected containers. Do not subject them to temperature extremes, put them near magnetic fields, store them unprotected in your pockets, etc. Even if you handle floppy and zip disks carefully, they are still prone to failure. For this reason, you should regularly back up your floppy or zip disks!

Every time you insert a disk into a computer, you may be transmitting a computer virus! Viruses are nasty software fragments that can erase information on your computer or cause other malfunctioning. In order to reduce the spread of computer viruses, make sure that any personal computers you use have appropriate virus protection software installed.

FINDING HELP

Since this is a 300-level class, it is expected that you will take more initiative in solving your own problems than in lower-level courses. For instance, if you don't understand something in the reading, you should look for alternative explanations in the library or on-line. Similarly, if you have trouble with the ML language or SMLNJ system, you should be willing to hunt for documentation that will help you.

Of course, I will be happy to answer any questions you might have. Your classmates are also a good resource for answering questions. But because the class is small, there are no tutors for the course, and I will be off-campus doing research several days a week, you will need to hone your self-help skills.

The best time to see me is during my scheduled office hours (which are listed at the top of this handout). If these times are not convenient, we can set up an appointment at some other time. You can set up an appointment by talking with me in person, calling me on the phone, or sending me email. You can also ask questions by sending me email or posting on the CS301-F00 conference. I read my email on a regular basis, and will check it even more frequently in the few days before an assignment is due.

FEEDBACK

I am eager to hear your feedback on the course! You can talk to me in person or send me email.

STUDENTS WITH SPECIAL NEEDS

If you have any disabilities (including "hidden" ones, like learning disabilities), I encourage you to meet with me so that we can discuss accommodations that may be helpful to you.

LATENESS COUPONS

Below are ten Lateness Coupons. A problem set that is *n* days late must be accompanied with *n* Lateness Coupons in order to be accepted. That is, each coupon gives you one extra day to turn in a problem set. You may use them in any manner in which you wish -- e.g., turn in every problem set one day late, or turn in one problem set ten days late. Coupons are not transferable between students. If you are a member of a two-person team and are turning in a late assignment, *each* member must turn in one coupon for each day late.

CS301 Lateness Coupon #1

CS301 Lateness Coupon #2

CS301 Lateness Coupon #3

CS301 Lateness Coupon #4

CS301 Lateness Coupon #5

CS301 Lateness Coupon #6

CS301 Lateness Coupon #7

CS301 Lateness Coupon #8

CS301 Lateness Coupon #9

CS301 Lateness Coupon #10