ECE468 Introduction to Compilers & Translation Engineering

Tentative Syllabus

Fall 2004 Tu/Th 9:00-10:15 AM, EE117

Instructor	Prof. R. Eigenmann
Tel	49-41741
Email	eigenman@ecn
Office	EE334C
Office Hours	Mo 4:30-5:30PM / We 8:00-10:00AM / Fr 8:00–9:00AM
Secretary:	Connie Boss, EE339, 49-43649
Course TA	Mohamed Gomaa, gomaa@ecn.purdue.edu, 49-40724
	Office Hours: Mo 10:00AM–12:00PM, Tu 10:30AM–12:30PM in EE249
Project TA	Sang-Ik Lee, sangik@ecn.purdue.edu, 49-43550
	Office Hours: Mo 1:30–3:30pm, We 1:30–2:30pm, Th 3:00-4:00pm Enad302
Course Web Page:	http://www.ece.purdue.edu/~eigenman/ECE468

Prerequisites: Proficiency in C language. C++ or Java experience is of advantage. ECE363 (Microprocessor Systems and Interfacing), ECE368 (Data Structures).

Text: Fischer and LeBlanc, Crafting a Compiler with C, Benjamin/Cummings, 1991, ISBN 0-8053-2166-7

Additional References: (1) Cooper and Torczon, Engineering a Compiler, Morgan Kaufmann. (2) Muchnick, Advanced Compiler Design&Implementation, Morgan Kaufmann,ISBN 1-55860-320-4. (3) Aho, Sethi and Ullman, Compilers: Principles, Techniques and Tools, 1986.

Description: The design and construction of compilers and other translators. Topics include compilation goals, organization of a translator, grammars and languages, symbol tables, lexical analysis, syntax analysis (parsing), error handling, intermediate and final code generation, assemblers, interpreters, and an introduction to optimization. Emphasis is on engineering, from scratch, a compiler or interpreter for a small programming language - such as a subset of C. The course includes a substantial project component, in which the students will stepwise implement such a compiler.

Course Outcome: After completing this course, the student will have demonstrated:

1. understand the terminology, representation and use of formal languages and grammars (1, 2, a)
2. understand the terminology and techniques of Lexical analysis, parsing, semantic processing, code generation, and optimization (1, 3, a, e)
3. an ability to design and implement a compiler, translator or interpreter for a small language based on their knowledge of (1) and (2) (1, 3, 4, a, b, c, e, k)

In order to receive a passing grade in the course, students must show proficiency in *all* topics: (1) terminology, (2) representations, and (3) use/application of formal languages; compiler techniques (5) lexical analysis, (6) parsing, (7) semantic processing, (8) code generation, and (9) optimization; and project skills (10) compiler design from specifications and (11) problem solving.

Course Grading:

Tests	50%	(10% each of three midterm exams, 20% final exam)
Homework	10%	
Class participation	5%	
Projects	35%	

Regrading policy: any information that you feel will affect your grade, including grade disputes and emergencies that prevent you from attending class, must be sent by email to the instructor. In general, the instructor will not respond to these notes or change your intermediate grades. However, the information will be factored into your final class grade. Note that a regrade request may increase *or* decrease your grade. Regrade requests must be received within a week from the assignment of the grade. Requests received after the last week of classes will not affect your grade.

You must check with the TA before the end of the last week of classes that the grades on the TA's file agree with your records. If you don't check this information, you may receive a grade other than the one you earned.

Student pictures: As part of your first homework assignment you must see the TA to take a digital photo. This is for the purpose of the TAs and instructor getting to know you and properly assigning credits and grades. You will only get credit for class participation if the instructor is in posession of your picture.

Academic Honesty: You are expected to do all programming and homework assignments on your own or within the designated group, respectively. You may not copy files or solutions from other students/groups or from previous years' courses. Your friends may not do part of all of assignments for you.

You may discuss concepts and ideas with other students. You may answer general questions about programming language rules, help someone interpret an error message, or locate a bug. In general, if the instructor or TA would provide a particular form of assistance, you may provide it too. When in doubt, check with the instructor or TA.

You must protect your work so that others cannot copy from it. You must get informed about possible ways others may break into your computer account and you must take all reasonable measures to prevent such misuse. If your work is being copied from, you may get involved in lengthy misconduct investigations and there is a chance you are found guilty of contributing to cheating. Such investigations may delay the assignment of your final grade and your graduation.

Punishment for academic misconduct can be severe, including receiving an F in the course or even being expelled from school. All cases of cheating will be reported to the Dean of Students. Note, that the Dean of Student will make an independent determination of the students' guilt. This procedure may continue even if the instructor assigns a final grade for ECE468

Course schedule:

```
30 lectures (eff. 15 weeks) plus final exam

                                    week    project step due
                                            (Friday 11:59PM)
Monday  Aug 23 Sem starts    T R    1              P1
           30                T R    2              P2
        Sep  6               T R    3
           13                T R    4              P3            Exam 1, Tu Sept 14
           20                T R    5
           27                T R    6              P4
        Oct  4               T R    7              P5
           11                . R    8                            Tu: October Break
           18                T R    9                            Exam 2, Tu Oct 19
           25                T R    10             P6
        Nov  1               T R    11
            8                T R    12             P7
           15                T R    13                           Exam 3, Th Nov 18
           22                T .    14                           Th: Thanksgiving
           29                T R    15             P8
        Dec  6               T R    16

        Dec 13      final exams week                            Final exam date TBA
```

Tentative Course Outline:

	Topic	#weeks	Reading
1.	Structure of a compiler	1	Chapters 1,2
2.	Scanning, Scanner Generators	2	Chapter 3
3.	Grammar, Parsing, Parser Generators	2	Chapters 4,5,6
4.	Semantic processing	1	Chapter 7
5.	Symbol Tables and Declarations	1	Chapters 8,10
7.	Processing Expressions, & control structures	2	Chapters 11,12
8.	Procedures and Functions	2	Chapters 13
9.	Code Optimizations	2	Chapter 15, 16, handouts
10.	Program Analysis	1	handouts
	Exams	2	

Reading: Reading the textbook sections corresponding to the presented lecture material is an essential part of the course.

Homework: Homework will be assigned after each lecture. Homework is due once a week, at the beginning of the Tuesday lecture. It is due on the following lecture if there is no regular lecture on Tuesday. The TA will collect homowork outside the classroom *before* class starts. There will be a deduction for homework turned in late. Only effort is graded on homework. Students may see the TA during office hours for detailed feedback.

Exams: There will be three exams during regular class hours. The final exam schedule will be announced on the course web page when known. All exams are comprehensive. For midterm exams, the emphasis is on the recently learned material. All exams are "open textbook and notes." However it is strongly recommended that you page through the textbook as little as possible during the exam. Searching in your notes can slow you down significantly. Use the textbook only in "emergencies". Midterm exams may include a question about

the syllabus.

Any conflict with the exam schedule need to be brought to the instructor's attention in the first lecture. For emergencies that prevent attendance of a scheduled exam, you must present documentation confirming exceptional reasons beyond your control. You must also contact the instructor, TA or secretary in person as soon as you become aware of the cause. Don't rely on email, as it may arrive late or not at all. The following are examples of insufficient reasons: non-emergency doctor visits, travel delays of less than a day, emergencies of persons other than immediate family members, job interviews scheduled after the first lecture.

Projects: In a group of two students, you will implement a compiler for a simple language. Starting with the code given in the textbook for the Micro language, you will extend the language to a "useful" mini language and add compiler passes for code generation and simple optimizations. You can choose the implementation language for your project. Groups will be formed in the first lecture and at the end of the second lecture. Students not attending these lectures will be assigned to groups by the TA. The first two project steps will be done individually.

General implementation guidelines:

- The specification of the project is given on the course web pages. It includes Examples and Q&A to clarify details. If you think a project specification is unclear or ambiguous, ask for clarification *before* you begin the project step. In general, no questions can be answered on the day of the project due date. Make sure you check the course web page before the submission of each project step, so you can take advantage of the latest Q&As posted.
- Your compiler should be able to handle programs up to 1000 lines of code. The use of dynamic data structures is not mandatory. However, if you use C++ or Java as an implementation language, it is highly recommended that you use the available dynamic container types for symbol tables and other data structures. If you use C, the project TA may approve the use of libraries for such functions. However, note that, unless approved by the TA, the use of code created by others is not allowed.
- You are encouraged to turn in your project steps *before* the deadline. The given dates are latest, *hard deadlines*. No extensions will be given. Keep in mind that ECN machines may be unavailable at times, and may be slow due to overloads before project deadlines.

Project grading policy: each project step will be graded separately. In general, you will need to fix problems to make the next project step work properly. However, no regrading will be done after such improvements. Most weight will be given to your final compiler, which is expected to generate code that executes correctly on the TINY simulator. 60% of the project grade can be achieved if your final compiler executes the predefined test programs correctly. Additional 10% are given if your compiler correctly executes several undisclosed test programs. The remaining 30% of the project grade are given for the intermediate project steps.

The project TA will handle all questions regarding the projects. For consistency reasons, the course TA and instructor will defer all such questions to the project TA and the project Q&A page.