Compilers
Spring 2009

INSTITUTO
SUPERIOR
TÉCNICO

Instructor: *Dr. Pedro Diniz*, e-mail :pedro.diniz@tagus.ist.utl.pt
Lectures: Tuesdays and Thursdays, 2.30 – 4.00 PM, phone: 214 233 517
Office hours: Tuesdays from 14.00 to 16.00 and Thursdays from 12.30 to 14.00,
Office: 2N.9.13

Teaching Assistants: *Prof. Jan Cederquist*, e-mail: jan.cederquist@tagus.ist.utl.pt
Office: 2N3.11, phone:
Office hours: TBD

João Sacramento, e-mail: joao.sacramento@ist.utl.pt
Office: 2N3.1
Office hours: Mondays from 14.00 to 16.00

Description: This course is intended to give the students a thorough knowledge of compiler design techniques and tools for modern computer programming languages. This course covers advanced topics such as data-flow and control-flow analysis, code generation and basic code optimization.

Prerequisites: Students should be familiar with the concepts in theory of computation (*e.g.,* regular sets and context-free grammars); design and analysis of algorithms (*e.g.,* asymptotic complexity, divide and conquer and dynamic-programming techniques); and master programming using dynamic, pointer-based data structures either in C or C++ programming languages. Students should also be familiar with basic concepts of imperative programming languages such as scoping rules, parameter passing disciplines and recursion.

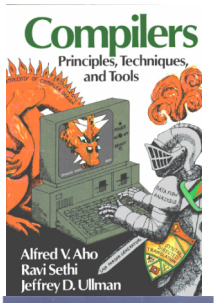
Assignments: We expect to have 5 individual homework assignments throughout the course in addition to a Mid-term and a Final in-class exam. Homeworks are due at the beginning of the class as the due date. Late homeworks will not be accepted as the solutions are posted within hours of the end of the class. Hardcopies of homeworks only. No e-mails of homeworks accepted. In extreme circumstances please ask for a fellow student to bring your homework in a sealed envelope.

There will also be 3 individual programming projects in this class. The goal of these projects is to have the students experiment compiler analysis and code generation in practice. These projects will cover topics we study in class and will the vast code basis available at the class web site you should have no problem completing each of them in under a week. For each project the TA will have a specific set of test programs some of which we make available to you before hand for testing. We do have additional test programs we use for our own evaluation and whose output will be compared against your project's outputs. At the end of each programming assignment we will make available an implementation that you might want to use for the following programming project.

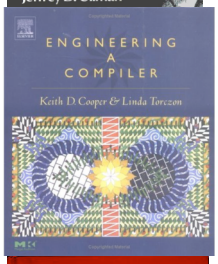
Grading: We will assign 30% to homeworks (some might have larger percentages due to difficulty), 30% for the programming projects (5%, 10% and 15% respectively), 20% for the mid-term and 20% for the final exam (known as second test). We will, in addition, assign extra credit of up to 2% for students who actively participate in web-based class discussion (see Class Material section below) in helping clarify questions other students submit. Thee are no make-up exams ("repescagem") nor special session exams ("época especial"). **All exams in addition to regular exams shall be oral examinations unless otherwise stated.**

Textbook: **There is no required textbook for this class**. Instead I'll draw most of the material from existing textbooks such as the one listed below. The material presented in class in addition to the resources you can draw from the WWW should be more than adequate. Occasionally, I'll distribute supplemental material in class to cover topics which organization is not easy to find elsewhere.

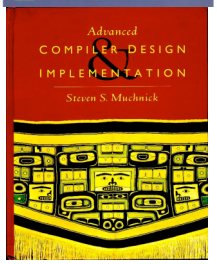
Title: *Compilers: Principles, Techniques and Tools*
Author: A. Aho, R. Sethi and J. Ullman
Publisher: Addison-Wesley
ISBN: 0-201-10088-6



Title: *Engineering a Compiler*
Author: Keith Cooper and Linda Torczon,
Publisher: Morgan-Kaufman Publishers
ISBN: 1-558600-698-X



Title: *Advanced Compiler Design and Implementation*
Author: S. Muchnick,
Publisher: Morgan-Kaufmann Publishers
ISBN: 1-558600-320-4



Class Material: Please gain access to course-related material on-line at the class website (http://webdei.rnl.ist.utl.pt/~ped/Teaching/2009/Compilers/). All students must register the class forum at http://conferenceserver.rnl.ist.utl.pt/compilers/. You can download course notes, check your grades, post questions, participate in discussion groups, keep current with class announcements, and do many other useful things.

Do not mail questions to the TA or instructor. Instead, post all questions on class' forum. Your questions will be answered as promptly as possible, and typically within 24 hours after being posted.

Lectures: Below is a tentative description of the contents of each lecture excluding the exams.

Lecture 1: Introduction; Lexical Analysis, Regular Languages and Finite Automata

- Introduction to compilation and programming languages
- Typical compiler structure and its internal phases
- Lexical Analysis: Its goal and input/output
- Finite Automata Theory and Regular Expression.
- Automatic Recognition of REG (example of Lex/Flex).

Lecture 2: Syntactic Analysis, Context-Free Languages and Push-Down Automata

- Syntactic Analysis: Its goal and input/output
- Context Free Languages and PDA.
- Bottom-Up and Top-Down parsing methods (LL(1) and LR(k) engines)
- Automatic Recognition of CFL (example of Bison/Yacc)
- Ambiguity and how to avoid it.

Lecture 3: Semantic Analysis and Syntax-Directed Translation

- Limitations of Syntactic Analysis.
- S-Attributed Grammar and L-Attributed Definitions.
- Translation to Syntax Trees and DAGs.
- Evaluation Schemes.

Lecture 4: Intermediate Representation and Intermediate Code Generation

- Introduction to Intermediate Code Generation, High-level Overview
- Intermediate Representation: Three-Address Instructions
- Syntax-Directed Translation Schemes for Code Generation
- Code for Expressions, Assignment, and Arrays
- Boolean and Relational Operators, Conditionals, & Control flow
- Back-patching

Lecture 5: The Procedure Abstraction, Run-Time Environment and Storage Allocation

- Introduction, discussion of control abstraction, name spaces, and external interfaces
- Symbol Tables & Storage Layout
- Allocating Storage and Establishing Addressability
- Finishing up Run-time Structures for OOLs.

Lecture 6: Code Generation and Instruction Selection

- Introduction to Code Generation, High-level Overview
- Basic Blocks and Traces
- Code for Expressions, Assignment, and Arrays in DAGs and Basic Blocks
- Boolean and Relational Operators, Conditionals, & Control flow

Lecture 7: Register Allocation

- The Importance of Register Allocation and Assignment
- Top-Down Simple Allocation Algorithm and Its Limitations
- Global Register Allocation via Graph Coloring.

Lecture 8: Introduction to Optimization

- Control-Flow Analysis: Dominators and Post-Dominators
- Finding Loops and Loop Invariant Code
- Strength Reduction, Constant Propagation and Constant Folding
- Basic Induction Variable Recognition

Lecture 9: Wrap-Up

- Compilers in perspective
- Further research in Compilers