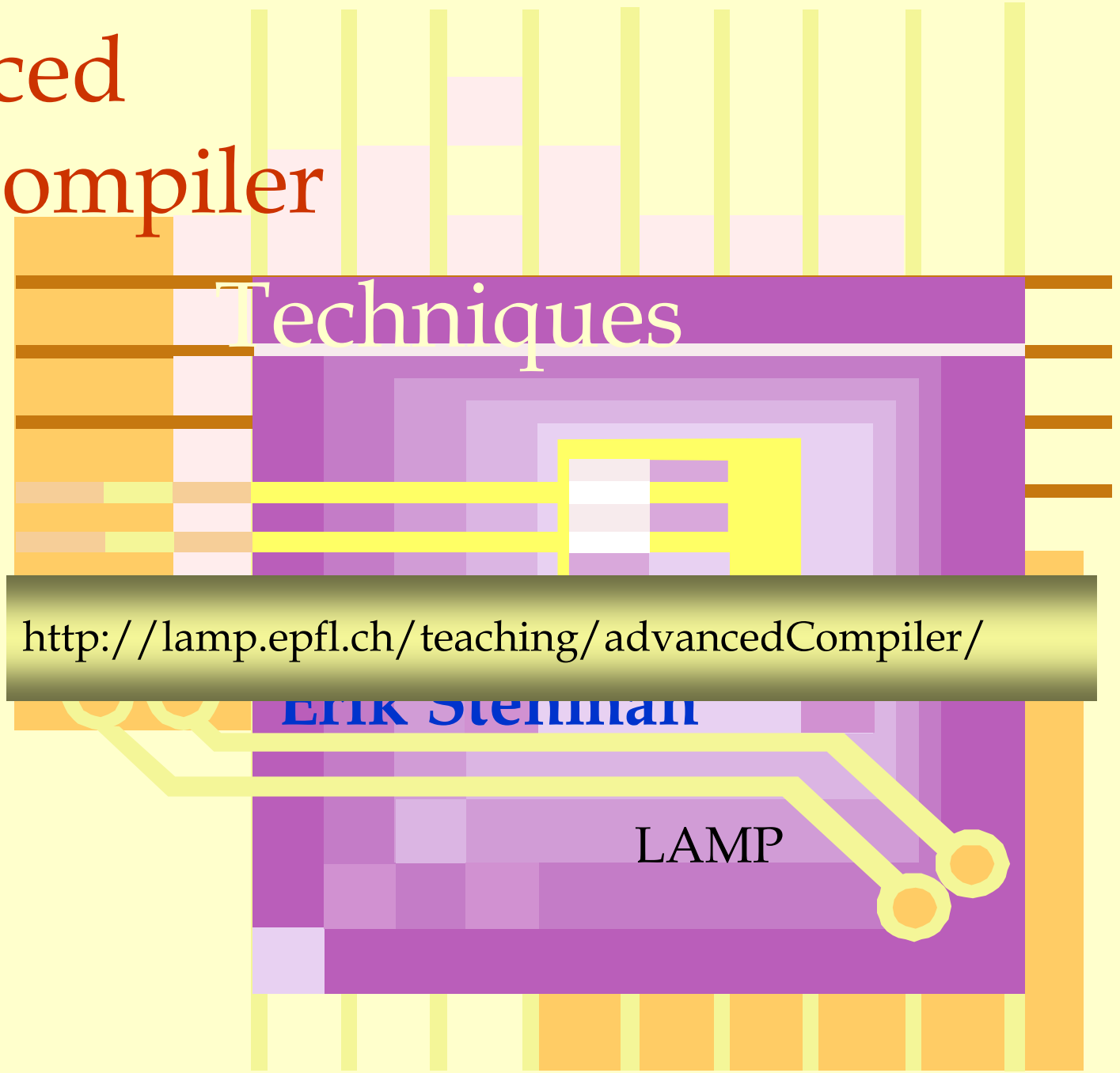
Advanced Compiler Techniques



Introduction

- What is this course about?
- How will this be taught?
- Who is teaching the course?
- Where to find more information?
- Why is this course interesting?

Teachers

- Lecturer
  - Erik Stenman
    - have been hacking compilers for money since 1996. Have been hacking for fun since 1980.
    - Office: INR315, 021-69 37593.
- Assistant
  - Michel Schinz
    - whom you all know and love.
    - Office: INR318, 021-69 34209.

Course Content

- Optimization Techniques
- Implementation techniques for high level languages (HLL).

Course Content

- Optimization Techniques
  - Theory for analysis and optimization
  - Optimization algorithms
- Implementation techniques for high level languages (HLL).
  - Virtual Machines
  - Memory Management
  - High level constructs

Course Goals

- Give some theoretical framework for compiler optimizations.
- Give a general orientation on optimization techniques.
- Give an understanding of how some higher level constructs are implemented.

Non-Goals and Requirements

- This course will not try to teach you all possible optimizations, or even all common optimizations.
- We will not talk about parallel machines.
- You are supposed to be familiar with basic compiler concepts: scanning, parsing, semantic analysis, and simple code generation. (These topics will not be touched.)
- You are supposed to be used to programming in Java.

Course Structure

- The course will be made up of lectures, articles, two projects, and an oral exam.
- The lectures will be given with slides like this one, and the slides will be available on the web:
  `http://lamp.epfl.ch/teaching/advancedCompiler/`
- I will try to have the slides on the web at least a day before the lecture.

Preliminary Schedule

1. Introduction
2. Control-Flow Analysis & Foundations of Data-Flow Analysis
3. Reaching Definitions, Available Expressions, and Liveness Analyses. Introduction to Abstract Interpretation.
4. Static Single Assignment Form, SSA-based Dead Code Elimination & Sparse Conditional Constant Propagation
5. [cont] Static Single Assignment Form, SSA-based Dead Code Elimination & Sparse Conditional Constant Propagation
6. Partial Redundancy Elimination & Lazy Code Motion
7. Loop Optimizations
8. Global Register Allocation
9. Code Scheduling
10. Implementation of higher order functions, processes, and objects
11. Automatic Memory Management
12. Virtual Machines, Interpretation Techniques, and Just-In-Time Compilers
13. Presentations

The Projects

- There will be two projects in the course and you may work in groups of two persons.
- Project 1: A simple register allocator.
  - The main goal of the first project is to get familiar with the compiler framework that we will use for the second project.
  - The task is to implement a Sethi-Ullman tree-based register allocator for a given compiler.
- Project 2: Optimizations.
  - The goal of the second project is to get a concrete understanding of different optimization techniques.
  - The task will be to implement different optimizations in the given compiler in order to achieve a given speedup on a set of benchmarks.

Literature

Expect to read a lot for this class, especially in order to complete the projects.

- Course Book:
  - Andrew W. Appel,
    Modern compiler implementation in Java (second edition).
    Cambridge University Press, 2002, ISBN 052182060X.
- Alternative:
  - Keith Cooper and Linda Torczon,
    Engineering a Compiler, Morgan Kaufmann, October 2003.
- Reference:
  - Steven Muchnick,
    Advanced Compiler Design and Implementation,
    Morgan Kaufmann, August 1997.
- Additional articles that will be handed out.

The Exam

- There will be an oral exam **during the last week** of the course.
- The exam will concentrate on the understanding of the concepts taught in the course, and not on details of specific algorithms.

Note:
The examination form of the course has changed from "Branche à examen (oral) avec contrôle continu" to "**Contrôle continu**"

Feedback

- This is the first time this course is given so the format and the content is not set in stone.
- At the end of next week there will be an evaluation of the course so far, and you will have chance to influence the rest of the course to a great extent.

Why is this course interesting?

- Optimization is challenging—you can not write an optimal compiler: there is always room for improvements.
- The course will give you many techniques and tools that you can use in other areas.
- You will gain a better understanding of how a compiler works and what to expect of the code generated by compilers.
- It is fun!