

# ECE663 Advanced Optimizing Compilers

Tentative Syllabus

Spring 2004 Mo/We/Fr 1:30-2:25 PM, EE 117

**Instructor** Prof. R. Eigenmann  
**Tel** 49-41741  
**Email** eigenman@ecn  
**Office** EE334C  
**Office Hours** by appointment (see finger info and send mail with suggested time)  
**Secretary:** Connie Boss, EE339  
**Course Web Page:** <http://www.ece.purdue.edu/~eigenman/ECE663/>

**Prerequisites:** EE573 or equivalent. Substantial programming experience.

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

Course notes and research papers will be used.

Background texts:

Michale Wolfe, *High Performance Compilers for Parallel Computing*, Addison-Wesley, ISBN 0-8053-2730-4.

Utpal Banerjee, *Dependence Analysis*, Kluwer, ISBN 0-7923-9809-2.

Ken Kennedy and John R. Allen, *Optimizing Compilers for Modern Architectures: A Dependence-based Approach*, Morgan Kaufmann Publishers, ISBN 1558602860.

Cooper and Torczon, *Engineering a Compiler*, Morgan Kaufmann, 2004, ISBN 1-55860-698-X.

**Description:** This course presents the concepts needed to design and implement advanced optimizing pre-processors and code generators. Specific emphasis will be put on techniques for exploiting parallelism in multiprocessors and in microarchitectures. Each student will conduct a compiler implementation project. Each student will also present a research paper from a list of selected articles.

**Objective:** Students who successfully complete this course will have demonstrated that they can

- explain the various passes of an optimizing compiler; including program analysis, dependence analysis, enabling transformations, loop restructuring, and instruction level parallelization,
- describe implementation methods and performance characteristics of these passes and tasks.
- explain program analysis techniques used to decide on correctness and profitability of the various program transformations.
- explain research issues related to these techniques, known solutions, differences between alternative solutions, and open issues.
- implement an optimizing compiler pass in the context of a realistic compiler.

**Course Grading:**

Tests 50%. (20% midterm, 30% final exam)  
 Participation in class 10%.  
 Class Presentation 10%.  
 Projects 30%

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 within a week from the event. 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.

**Course schedule:**

45 lectures (eff. 15 weeks) plus final exam

				week	
Monday	Jan 12	Sem starts	M W F	1	
	19		. W F	2	Jan 23: draft project proposal due
	26		M W F	3	Jan 30: final project proposal due
	Feb 2		M W F	4	
	9		M W F	5	
	16		M W F	6	
	23		M W F	7	
	Mar 1		M W F	8	March 3: Midterm exam
	8		M W F	9	Mar 12: first project report due
	15	Spring break			
	22		M W F	10	
	29		M W F	11	
	Apr 5		M W F	12	
	12		M W F	13	
	19		M W F	14	
	26		M W F	15	Apr 30: final project report due
	May 3	final exams week			Final exam date: TBD

(. Jan 19: M.L. King Day, no lecture)

**Tentative Course Outline:**

	Topic	#weeks
1.	Introduction and Motivation	1
2.	Effectiveness of parallelizing compilers	1
3.	Basic Transformations	1
4.	Program Analysis I	1
5.	Advanced Loop Optimizations	1
7.	Program Analysis II	2
8.	Performance of Compiler Techniques	1
9.	Instruction Level Parallelization	2
10.	Class Presentations	3

**Class Projects:** Each student will implement an optimizing compiler pass or an infrastructure module within the Cetus compiler infrastructure. Project details will be presented in the first week of class. Much of the implementation work is done individually. However, the design and the final, overall evaluation will be done collaboratively.

A draft and final project proposal as well as an intermediate and final project report is to be submitted to the instructor as per the class schedule, at the beginning of the lecture. Final (intermediate) project report is approximately 10(5) pages, excluding code, 10pt font, double spaced. The front page of each report must include the project title, student name, course number (ECE663), and whether it is the draft proposal, final proposal, intermediate project report, or final project report.

As time permits, project results will be presented in class.

**Class Presentations:** Each student will present a paper from the selected list of papers on the class web page. The presentation will be 35 minutes, followed by questions. Good timing of the class presentation will factor into the grade.