ECE663 Advanced Optimizing Compilers

Tentative Syllabus

Spring 2012 Mo/We/Fr 10:30-11:20 PM, EE 224

Instructor	Prof. R. Eigenmann
Tel	49-41741
Email	eigenman@purdue.edu
Office	EE325
Office Hours	tbd
Secretary:	Jill Comer, EE325b
Course Web Page:	http://engineering.purdue.edu/~eigenman/ECE663/

Prerequisites: EE573 or equivalent. Substantial programming experience.

Text: 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 source-to-source preprocessors and code generators. Specific emphasis will be put on techniques for exploiting parallelism in multiprocessors and multicores. Each student will conduct a compiler implementation project and present the results in class.

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 program transformations.
- Explain research issues related to these techniques, including 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  9 Sem starts    M W F    1
            16               . W F    2      Jan 16: M.L. King Day, no lecture

            23               M W F    3      Jan 23: draft project proposal due
            30               M W F    4      Jan 30: final project proposal due
        Feb  6               M W F    5
            13               M W F    6
            20               M W F    7
            27               M W F    8       Feb 29 Midterm exam
        Mar  5               M W F    9       Mar 5: first project report due
            12              Spring break
            19               M W F    10
            25               M W F    11
        Apr  2               M W F    12
             9               M W F    13
            16               M W F    14
            23               M W F    15      Apr 29: final project report due
            30             final exams week   Final exam date: TBD
```

Tentative Course Outline:

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

Exams: There will be a midterm and a final exam. The midterm exam covers the material approximately through "Parallelism Enabling Techniques". Specifics will be posted on the course web page. The final exams is comprehensive. All exams are "open textbook and notes." However it is strongly recommended that you page through the textbook and notes as little as possible during the exam. Searching in your notes can slow you down significantly. Use the textbook only in "emergencies".

Class Projects: Each student will propose and implement a compiler pass or an infrastructure module within the Cetus compiler infrastructure. Exceptions are possible, but need instructor approval. Additional project details will be presented in the first week of class. The projects may be implemented in groups of up to 2 students.

A draft and final project proposal as well as an intermediate and final project report is to be submitted by email, in PDF format, to the instructor as per the class schedule, before the end of the indicated day. The project proposal should include: A description of the project to be done; a clear description of the outcomes and deliverables you expect to have at the end of the project; and a timeline, indicating when you expect to have completed what part of the project (3-4 milestones during the semester). The final (intermediate) project report is approximately 10 (5) pages, excluding code, 11pt font, double spaced. The front page of each report must include the project title, student name(s) and email address(es), course number (ECE663), and whether it is the draft proposal, final proposal, intermediate, or final project report.