# CS 597d - Advanced Topics in Computer Science: Synergistic Hardware-Compiler Architecture Design

**INSTRUCTOR:**

David August
CS407, 258-2085
august@cs.princeton.edu
Office hours by appointment

**LECTURES:**

Time: Monday and Wednesday 11:00-11:50
Room: CS302

**COURSE WEB PAGE:**

`http://www.cs.princeton.edu/courses/archive/fall99/cs597d/`
The course web page is the authoritative source for all course information. Announcements, lecture notes, and project information will be posted regularly.

**DESCRIPTION:**

This advanced computer architecture course explores the nature of and the motivation for recent trends in uniprocessor computing. Emphasis is placed on a commonality among many of these trends, increased reliance on sophisticated compilation. The compiler, no longer just a consideration in instruction-set architecture design, has become the driving factor in many architectural innovations. Predication, speculation, value prediction, and other hardware/compiler techniques which exploit instruction-level parallelism will be explored using real codes. The course includes a project involving the IMPACT Research Compiler and a working EPIC (Explicitly Parallel Instruction Computing) architecture similar to Intel's IA-64.

**SUPPLEMENTAL TEXT:**

The primary reading will be the assigned papers. This can be optionally supplimented with *Advanced Compiler Design and Implementation* by Steven S. Muchnick.

**PREREQUISITES:**

- Computer Architecture (CS471 or equivalent)
- Compiler Design (CS320 or equivalent)

To fully appreciate the course, students should be familiar with assembly language, the C programming language, control flow graphs, data dependence graphs, and processor pipelining.

**PROJECT:**

There will be a single class project done in collaboration with the instructor using the IMPACT Research Compiler. The project will be designed to familiarize the student with the IMPACT Research Compiler, a valuable tool for computer architecture research. The ultimate goal of the project is to explore a novel architectural feature which requires advanced compiler support. Idealy, the result will be publishable.

**STUDENT LECTURES:**

Students may opt out of the project by preparing and giving two lectures in collaboration with the instructor. The subjects of the lectures can be as listed in the tentative course schedule or can be on any topic related to compiler/microarchitecture codesign.

**READINGS:**

As a graduate student, if all you do in a single day is read, understand, and critique a conference or journal paper, that day is not wasted. Each lecture topic has a set of corresponding conference and/or journal papers associated with it. Before a new topic is covered in class, students submit an observation of one of the relevant papers, chosen by the student. To get the most from the class, I encourage students to read all the papers.

Instructions for obtaining each paper and for submitting your observations are available on the course web page. Student observations will be posted on the course web page to encourage consideration and discussion by the rest of the class. On average, there will be one paper observation due before each lecture. Grading for the readings is given by $Observations/(Assigned - 4)$.

**PARTICIPATION:**

High-performance computer architecture is far from a closed topic, and much opportunity exists to compare and contrast alternative solutions. The readings have been selected to present a variety of views on techniques for achieving high performance and to generate productive class discussion. Participation involves listening, constructive comments, and meaningful questions.

**GRADING:**

- Each Lecture: 25%
- Project: 50%
- Readings: 40%
- Participation: 10%

If both the project and a lecture are done, the more favorable grade is used.

**TENTATIVE COURSE SCHEDULE:**

| Lect. | Date | Topic |
|---|---|---|
| 1 | 9/20 | Introduction - Course overview, logistics, and opportunities |
| 2 | 9/22 | Branch Prediction - current hardware methods |
| 3 | 9/27 | Branch Prediction - synergistic methods |
| 4 | 9/29 | Profiling - traditional compiler methods |
| 5 | 10/4 | Profiling - synergistic methods |
| 6 | 10/6 | The EPIC Philosophy - Playdoh, IA-64 |
| 7 | 10/11 | Instruction Scheduling - dynamic scheduling |
| 8 | 10/13 | Instruction Scheduling - acyclic scheduling, heuristics |
| 9 | 10/18 | Control Speculation - hardware OOO execution |
| 10 | 10/20 | Control Speculation - sentinel scheduling |
| 11 | 10/25 | Control Speculation - trace scheduling, compensation code |
| 12 | 10/27 | Control Speculation - superblock, treegions |
| FALL BREAK (10/30 to 11/7) | | |
| 13 | 11/8 | Predication - predication architectures |
| 14 | 11/10 | Predication - compilation, if-conversion, hyperblocks |
| 15 | 11/15 | Predication - predicate relationship analysis |
| 16 | 11/17 | Predication - program decision logic optimization |
| 17 | 11/22 | Predication - predicate-aware dataflow analysis |
| 18 | 11/24 | Predication - partial reverse if-conversion framework |
| 19 | 11/29 | Memory Dependence Analysis - pointers, sync-arcs, sync-vars |
| THANKSGIVING BREAK (11/25 to 11/28) | | |
| 20 | 12/1 | Value Speculation - memory dependence speculation |
| 21 | 12/6 | Value Speculation - value profiling and speculation (guest lecture) |
| 22 | 12/8 | Cache Management - current methods |
| 23 | 12/13 | Cache Management - alternative memory management |
| 24 | 12/15 | Course Summary - research opportunities |