

CS 671 – Semester Projects

The final project in CS 671 is a very important component of the course which will amount to 40% of your final grade. You may work alone or in groups of two. Like many other graduate course projects, the intent of the CS 671 course project is to give you experience in doing publishable research.

There are three graded components to the project:

1. The Written Proposal (30%) – after discussing your project ideas with Professor Hazelwood and getting the thumbs up, you will need to follow the proposal template provided.
2. The Oral Presentation (15%) – during our exam timeslot, you will present your project to the class (for groups of two, make sure that you both get some face-time during the presentation).
3. The Final Write-up (55%) – a 10-page conference quality report using the template provided.

Project Scope

It is in no one's best interest for you to work on a "throw away" project for this course.

I highly encourage you to come up with a project for this course that would either:

- be useful/relevant to your own research area
- be part of your thesis work
- leverage your expertise in another area to improve the state of compilation or optimization today
- be planned for publication (FYI – Micro deadline is June; CGO + VEE are Sept)

Given the above guidelines, I am fairly flexible regarding topics. However, I must make the following disclaimers:

1) It is very important to me that your project is "doable" in the two months remaining in this course. I see a lot of proposed projects that would take an experienced researcher 5+ years to complete, and I won't permit these as course projects unless you propose a clearly defined two-month sub-project and evaluation plan.

2) It is also very important that your project is well defined. This means that you should have already come up with all of the ideas that will go into the project. Coming up with an idea cannot be one of the deliverables of your proposed project.

Half-Baked Ideas (in the event that you absolutely cannot come up with your own project)

- **Compiler-Inserted "Hints"**. Can you think of metadata that we may want to embed into applications? For security? Run-time optimization? (Clearly specify what hints you will insert, why, what infrastructure you'll use to insert the hints, and how they'll be leveraged.)

- **Dynamic Forward Compatibility:** New instructions in the ISA (e.g. SSE5) provide no benefit if the compiler didn't generate these instructions in the binary. Can we statically or dynamically remedy this fact?
- **Optimization Combinations.** Some people have looked at machine learning algorithms for determining which combinations of optimizations work best for a given program (See the feedback-drive, performance-counter based approach taken by John Cavazos at CGO 2007.)
- **Program Temperature Analysis and Optimization.** Can temperature be tied back to instruction sequences? If so, is this behavior repeatable? Can you come up with a temperature virus that exercises the worst case? Can you develop an optimization that resolves the issue?
- **Code Region-Based OS Scheduling.** Is there anything about upcoming code sections that an OS could leverage in making scheduling decisions?
- **Dynamic Data Re-Layout.** The milc benchmark in SPEC 2006 has an access pattern that "breaks" the Intel hardware prefetcher. Is there a way to detect this? Can we remedy it at run-time and/or provide feedback to a compiler to rearrange the sequences and/or insert the proper prefetch instructions?
- **Code Size Optimizations** (Again, I need the specific details on the exact optimization you plan.)
- **Program Visualization Tools** (1 person max). It would be useful to see a graphical depiction of the control flow, memory footprint, performance, and/or power footprint of either
 - (a) the compilation process itself, or
 - (b) a running program
- **Other Projects.** Something that involves one or more of the following:
 - Hardware performance counters
 - Microsoft Phoenix
 - Intel Pin or VTune

Projects That You May Not Choose (so please don't ask)

- Automatic parallelization (yawn)
- I'm going to build [insert large system here with no motivation, no detailed implementation plan, and no contingency plan]
- Anything that has already been done before – it is your responsibility to do a literature review to be certain