EE663, Spring 2002
Course Format

• Lectures: 70% by instructor; include hands-on exercises
• Class presentations. Each student will give a presentations on a selected paper from the list on the course web page.
• Projects: Implement a compiler pass within either the Gnu C 3.0 infrastructure or a new research infrastructure to be designed in this class.
  – Friday of Week #1: Announcement of project details.
  – Friday of Week #2: Preliminary project outlines due; Discuss with instructor
  – Friday of Week #3: Project proposals finalized.
• Exams: 1 mid-term, 1 final exam

EE663 Project

Design an optimizing compiler for C programs.
• We will design two compilers, one based on the GNU C3.0 infrastructure, the other based on a new infrastructure being designed at Purdue (called Cetus).
• Every student will implement a pass for one of these two compilers. We will implement the passes:
  – reduction recognition and parallelization
  – induction variable substitution
  – scalar and array privatization
  – data dependence test
EE663 Project continued

• The students can work cooperatively. In particular, the two students working on the same pass for the two infrastructures can do the high-level design jointly.
• Evaluation will be based on individual pass performance with bonuses for overall design and success on the compilers.

EE663 Project continued

Initial Assignments:
– learn about the Gnu C 3.0 infrastructure (start with Google)
– learn about the Cetus infrastructure (will be presented in lectures and handouts)
– learn the basics about the passes reduction parallelization, induction variable substitution, privatization and data dependence test (in class and from recommended papers).
– pick a pass and infrastructure for your project.
Overall Structure of the Cetus Parallelizing Compiler

<table>
<thead>
<tr>
<th>Primary student tasks:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task for one student:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Induction Variable Substitution</th>
<th>Reduction Parallelization</th>
<th>Privatization</th>
<th>Data Dependence Test</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Base Interface</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defines the internal representation in terms of an abstract data structure. All elements of the representation can only be read and modified via access functions.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| **Base**                        |                            |               |                     |
| contains the internal representation (abstract syntax tree) The organization is hidden from the developer of passes. (all you need to know is that it exists) |