CSC/CPE 431 – Compiler Construction

1. CSC/CPE 431 – Compiler Construction

2. credit units  4  contact hours  6

3. Course Coordinator: Aaron Keen

4. Textbook (or other required material): None

5. a. Course Description:
Intermediate code representations, memory management, functions and parameter passing, code transformations and optimizations, code generation, register allocation. 3 lectures, 1 laboratory. Crosslisted as CPE/CSC 431.

b. Prerequisite: CSC 430

c. Required/Elective/Selective Elective for CPE, CSC, EE, SE

<table>
<thead>
<tr>
<th></th>
<th>CSC</th>
<th>CPE</th>
<th>SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elective</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Selective Elective</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6. a. Course Learning Objectives
The student will be able to:

- Specify and implement intermediate representations of a program.
- Implement transformations from either “high-level” language constructs or an existing intermediate representation to either an intermediate representation or an assembly representation.
- Demonstrate the mapping of “high-level” language constructs to assembly code.
- Explain the operation of the run-time stack and parameter passing.
- Implement a register allocation algorithm.
- Explain the steps of select code transformation algorithms (including selections from those leveraging iterative dataflow analyses and abstract interpretation).
- Analyze the advantages and disadvantages of select code transformation algorithms.
- Given a specification, implement an optimization algorithm.
- Discuss the effects of code transformations in the context of other transformations (e.g., expose additional opportunities, increase register pressure).

b. Level at which Student Outcomes are addressed
(“B” = Basic level, “I” = Intermediate level, “A” = Advanced level)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSC</td>
<td>A</td>
<td>A</td>
<td>I</td>
<td></td>
<td>B</td>
<td>A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
7. Major Topics Covered: (number of lecture hours per)

- Intermediate code representation (6)
  - Abstract syntax trees
  - Control flow graphs
  - Linear intermediate representation
  - Static single-assignment
  - Symbol tables
- Memory management (2)
  - Stack management for parameter passing
- Code transformations and Optimizations (12)
  - Iterative dataflow analysis
  - Abstract interpretation
  - Examples include
    - Sparse conditional constant propagation
    - Local value numbering
    - Useless code elimination
    - Tail-recursion (call) optimization
    - Constant folding
    - Loop invariant code motion
- Register Allocation (2)
- Code generation (3)
  - Assembly review
  - Consideration for and effects of constraints on, e.g., register use in specific instructions
  - Register management across function calls