CS 2210–Compiler Design
2018 Spring Term

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time:</td>
</tr>
<tr>
<td>Days:</td>
</tr>
<tr>
<td>Room:</td>
</tr>
<tr>
<td>Webpage:</td>
</tr>
</tbody>
</table>

Contact Information

<table>
<thead>
<tr>
<th>Instructor: Wonsun Ahn</th>
<th>Grading TA: Zhenjiang Fan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office: 6115 SENSQ</td>
<td>Office: 6501 SENSQ</td>
</tr>
<tr>
<td>Email: <a href="mailto:wahn@pitt.edu">wahn@pitt.edu</a></td>
<td>Email: <a href="mailto:zhf16@pitt.edu">zhf16@pitt.edu</a></td>
</tr>
<tr>
<td>Office Hours: Mon 2:30 – 4:00 Wed 1:30 – 3:00</td>
<td>Office Hours: WedThu 12:30 – 2:00</td>
</tr>
</tbody>
</table>

Description

This course takes you through the design and implementation of current high-level languages. Both frontend and backend issues are dealt with. The frontend is the part of a compiler that analyzes a program to produce a form of internal representation (IR) that is more easily understood by the compiler. The frontend comprises components such as: lexical analysis, syntax analysis, and semantic analysis. The backend is the part of a compiler that takes the IR and produces machine code that can be efficiently executed by the processor. The backend comprises components such as: target code generation, register allocation, and code optimization. The course also touches upon the management of storage locations such as processor registers, the program stack, the program heap and the role of the compiler and language runtime.

Prerequisites

There are no prerequisites for this class but you are expected to have basic knowledge of computer organization pursuant of CS 0447 – Computer Organization and Assembly Language Programming. You should know what an instruction set architecture (ISA) is, what CPU registers are, what CPU caches are, and what assembly instructions look like. You should also be comfortable programming in C. If you have any questions about the prerequisite material for the course, please ask at the beginning of the term.
Textbooks


Class Policies

Exams: There will be a midterm and a final. The exams will be closed book/notes. Cheating on exams will not be tolerated. Anyone caught cheating will be given a zero for the test and reported to the department following University procedures.

Projects: There will be four course projects, which will take you through the implementation of a compiler for a simple toy programming language called Mini-Java, starting from the frontend and ending in the backend. The projects are to be done in C on Linux/Unix environments. The project components are assigned in roughly increasing order of size and difficulty.

Late submissions will incur a 10% penalty per day. These are meant to be your own work; anyone found to be collaborating will be disciplined in accordance to University policy. Cheating means (but is not limited to): using code from previous terms, other universities, your friends, finding it on the Internet, getting help from unapproved forums, or outsourcing it. We will be using Moss, a tool from Stanford for determining inappropriate collaboration.

Homeworks: There will be around two written homeworks that are meant to check your understanding of the course material.

Participation: Attendance will be taken at random intervals at my discretion. Please let me know in advance if you are going to miss a class.

Grading

Your grade will be based upon 2 exams, 4 projects, and a number of homeworks:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Midterm Exam</td>
<td>25%</td>
</tr>
<tr>
<td>Final Exam</td>
<td>25%</td>
</tr>
<tr>
<td>5 Projects</td>
<td>30% (7.5% each)</td>
</tr>
<tr>
<td>Homeworaks</td>
<td>10%</td>
</tr>
<tr>
<td>Participation</td>
<td>10%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100%</strong></td>
</tr>
</tbody>
</table>
Disability Resources and Services
If you have a disability for which you are requesting an accommodation, you are encouraged to contact both your instructor and Disability Resources and Services, 216 William Pitt Union, (412) 648-7890, as early as possible in the term. DRS will verify your disability and determine reasonable accommodations for this course.

Academic Integrity
Students are expected to comply with the University of Pittsburgh’s Policy on Academic Integrity. Any student suspected of violating this obligation for any reason during the semester will be required to participate in the procedural process as outlined in the University Guidelines on Academic Integrity. For further information see: http://www.pitt.edu/~provost/ai1.html

Term Schedule
The daily topics are subject to change depending on our pace. They are there to assist you in the readings so you can focus on those concepts prior to class.

<table>
<thead>
<tr>
<th>Week 1: 1/8/2018 - 1/10/2018</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>READINGS:</strong> <em>Textbook</em>, Chapters 1.1 – 1.5, 3.1 – 3.5</td>
</tr>
<tr>
<td><strong>TOPICS:</strong></td>
</tr>
<tr>
<td>• Intro to the Course</td>
</tr>
<tr>
<td>• Lexical Analysis</td>
</tr>
<tr>
<td>o Tokens</td>
</tr>
<tr>
<td>o Regular Languages</td>
</tr>
<tr>
<td>o Regular Expressions</td>
</tr>
<tr>
<td>o Lex Tool</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Week 2: 1/15/2018 - 1/17/2018</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>READINGS:</strong> <em>Textbook</em>, Chapters 3.6 – 3.8, 3.9.6 – 3.9.7</td>
</tr>
<tr>
<td><strong>TOPICS:</strong></td>
</tr>
<tr>
<td>• Lexical Analysis (continued)</td>
</tr>
<tr>
<td>o Deterministic Finite Automata</td>
</tr>
<tr>
<td>o Nondeterministic Finite Automata</td>
</tr>
<tr>
<td>o Regular Expression to NFA</td>
</tr>
<tr>
<td>o NFA to DFA</td>
</tr>
<tr>
<td>o DFA Minimization</td>
</tr>
<tr>
<td>o DFA to table-driven implementation</td>
</tr>
</tbody>
</table>
### Week 3: 1/22/2018 - 1/24/2018

**READINGS**: *Textbook*, Chapters 4.1 – 4.2

**TOPICS**:

- Syntax Analysis
  - Grammars
    - Noam Chomsky Classification
  - Languages
  - Context Free Grammars
    - Parse Trees
    - Leftmost / Rightmost Derivations
    - Ambiguity
    - Associativity / Precedence
    - Abstract Syntax Tree

### Week 4: 1/29/2018 - 1/31/2018

**READINGS**: *Textbook*, Chapters 4.3 – 4.5

**TOPICS**:

- Top-down Parser
  - Recursive descent using backtracking
    - Removing left-recurion
  - Predictive
    - Left factoring
    - LL(k) parser / grammar / language
    - Recursive
    - Nonrecursive (table driven)
    - Look ahead
  - Parse Table Construction
    - First Set
    - Follow Set
    - Dealing with conflicts

### Week 5: 2/5/2018 - 2/7/2018

**READINGS**: *Textbook*, Chapters 4.5 – 4.6

**TOPICS**:

- Bottom-up Parser
  - Handles
  - Shift-reduce parsing
  - Conflicts and ambiguity
- LR Parsing
  - Viable Prefix
### Week 6: 2/12/2018 - 2/14/2018

**READINGS**: *Textbook*, Chapters 4.7 – 4.9

**TOPICS**:
- More powerful LR Parsers
  - Look ahead
  - SLR(k)
  - LR(k)
    - State splitting
  - LALR(k)
    - State merging
- Comparison between different parsers
- YACC Tool
  - Shift-reduce conflicts
  - Reduce-reduce conflicts
  - Semantic Actions
  - Error Recovery
  - LR Table Compaction

### Week 7: 2/19/2018 - 2/21/2018

**READINGS**: *Textbook*, Chapters 2.7, 6.4.3, 6.5 – 6.5.2

**TOPICS**:
- Semantic Analysis
  - Static scope / Dynamic scope
  - Symbol Table
  - Semantic of Arrays
  - Type Checking

### Week 8: 2/26/2018 - 2/28/2018

**READINGS**: *Textbook*, Chapters 5.1 – 5.2.4, 5.5

**TOPICS**:
- Syntax Directed Translation
  - Attribute Grammar
  - Synthesized Attribute / Inherited Attribute
  - Syntax Directed Definition
Week 9: 3/5/2018 - 3/7/2018

March 4–11, 2018: Spring Break, University Closed

Week 10: 3/12/2018 - 3/14/2018

Monday, March 12, 2018: Midterm Review

Wednesday, March 14, 2018: Midterm Exam

Week 11: 3/19/2018 - 3/21/2018

Readings: Textbook, Chapters 6 intro, 6.2, 6.3.4 – 6.3.6, 6.4

Topics:

- Code Generation
  - Multi-level IRs / frontends / backends
  - Low-level IRs
    - Three Address Code and its representation
    - SSA
  - Storage layout
    - Offset
    - Endianness
- Code Generation Through Syntax Directed Translation
  - Processing Declarations
  - Processing Statements
  - Processing Array references

Week 12: 3/26/2018 - 3/28/2018

Readings: Textbook, Chapters 1.6.6, 6.6 – 6.7, 7 intro, 7.1 – 7.3, 12.2.6

Topics:

- Code Generation Through Syntax Directed Translation (cont’d)
  - Processing Boolean Expressions
    - Short-Circuiting and non-L attributes
  - Handling non-L attributes
    - Two-pass based approach
    - One-pass base approach using backpatching
- Runtime Management
  - Concept of runtime environment / runtime code
  - Static / stack / heap data management
  - Stack data management

MARCH 4–11, 2018: SPRING BREAK, UNIVERSITY CLOSED
• Advantages / Disadvantages compared to static management
• Activation records (ARs)
• Frame pointer / stack pointer
• Calling convention and AR layout

• Runtime Code Generation
  • Code for global / local / non-local variable access
  • Nested procedure declarations
  • Code of function calls and calling conventions
  • Call by value, by reference, by name
  • Example MIPS assembly
  • Expressions, function calls, variable accesses
  • Buffer Overflow Attack
  • Classes / Objects
  • Member variables / member methods
  • Virtual methods / non-virtual methods
  • Dynamic dispatch / static dispatch

---


**READINGS:** *Textbook*, Chapters 7.4 – 7.7

**TOPICS:**

• Garbage Collection
  • Motivation
  • Reachability and garbage
  • Two GC schemes
    • Reference Counting
    • Tracing
  • Reference Counting
    • Circular data structures
    • Advantages / Disadvantages
  • Tracing
    • Root objects
    • Advantages / Disadvantages
  • Comparison of different Tracing GC
    • Mark-Sweep
    • Semispace
    • Incremental
    • Generational

---


**READINGS:** *Textbook*, Chapters 7.4.2. – 7.4.4, 8.4, 8.5.4, 9.2.1 – 9.2.3, 9.2.5, 9.3, 9.4

**TOPICS:**

• Code Optimization
  • Data-related optimization
• Data layout, code layout
  ▪ Code-related optimization
    ▪ Different categories
  ▪ Local optimization
    ▪ Strength reduction
    ▪ Constant folding
  ▪ Global optimization
• Control flow analysis
  ▪ Basic block
  ▪ Control flow graph
  ▪ Conservatism due to control flow
• Data flow analysis
  ▪ Data flow framework components
    ▪ Direction
      ▪ Forward analysis
      ▪ Backward analysis
    ▪ Set of values
    ▪ Meet operator
      ▪ Semilattice
      ▪ Top value / bottom value
    ▪ Transfer function
  ▪ Data flow framework implementation
    ▪ Iterative refinement of values
    ▪ Termination guarantee
  ▪ Global Constant Propagation
  ▪ Global Liveness Analysis

Week 15: 4/16/2018 - 4/18/2018

READINGS: Textbook, Chapters 8.6, 8.8, 8.9.1 – 8.9.2

TOPICS:

• Register Allocation
  ▪ Local register allocation
    ▪ Register spilling
  ▪ Global register allocation
    ▪ Graph coloring
      ▪ Live ranges
      ▪ Register interference graph
      ▪ Chaitin’s algorithm
        ▪ Optimality
        ▪ Spilling and retry
    ▪ Linear scan
      ▪ Live intervals
      ▪ Linear code layout
    ▪ ILP
  ▪ Instruction selection
○ Importance of CISC
○ Tree tiling

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MONDAY, APRIL 23, 2018:</strong> Final Review</td>
</tr>
<tr>
<td><strong>WEDNESDAY, APRIL 25, 2018:</strong> Final Exam</td>
</tr>
</tbody>
</table>