Topics

We will...

- learn how compilers are constructed,
- learn how programming languages are designed,
- learn how to compile procedural and object-oriented languages,
- learn how interpreters work,
- learn how garbage collectors work,
- learn how debuggers work.

Syllabus

lexing

- The Chomsky hierarchy,
- regular expressions,
- DFAs,
- scanner implementation.

parsing

- Context-free grammars, BNF,
- parse trees,
- abstract syntax trees,
- Recursive Descent parsing.
Syllabus...

- Advanced topics will be covered as time permits:

**code optimization**
- data-flow analysis,
- common sub-expression elimination,
- inlining.

**object-orientation**
- compiling Java-like languages,
- run-time class templates,
- inheritance.

Syllabus...

**semantic analysis**
- Attribute grammars,
- environments,
- type-checking.

**intermediate representations**
- stacks,
- tuples,
- trees,
- intermediate-code generation from abstract syntax trees.

Syllabus...

**garbage collection**
- reference counting,
- pointer-maps,
- mark-and-sweep.

**systems software**
- debuggers,
- profilers.

Syllabus...

**code generation**
- control-flow graphs,
- code generation for arithmetic expressions, data-structure access, control-flow, and procedure calls.

**code optimization**
- survey of techniques,
- peephole optimization.
Assignment 4 Interpretation.
- Luca interpreter.
  input: Luca stack code
  output: none
- Implementation language: gcc.
- Algorithm: indirect threaded code.
- Out: Nov 1. Due: Nov 18. 100 points

Assignment 5 Code generation.
- Generate SPIM code for a subset of the Luca language.
  input: Luca control-flow graph
  output: MIPS assembly code
- Algorithm: any.
- Out: Nov 18. Due: Dec 6. 100 points

Assignment 6 Peephole optimization.
- Perform local optimization on the Luca IR and/or the generated MIPS code.
  input: Luca control-flow graph
  output: Luca control-flow graph
- Algorithm: any.
- Out: Dec 6. Due: Dec 16. 100 points
- You may work alone or in teams of 2.
- If your graded score for an assignment is $g$ and you handed in $k$ days late, then your computed score for this assignment will be
  \[
  \begin{cases} 
  \max(0, g(1 - 0.1k)) & \text{if } k \leq 5 \\
  0 & \text{otherwise}
  \end{cases}
  \]
Grading

- Grading (Subject to change)
  1. One midterm exam (10%)
  2. One final exam (30%)
  3. Seven programming assignments (60%)

- Midterm exam: Friday, October 18, in-class.
- Final exam: Friday, December 20, 11:00–13:00.

1. All exams are closed book.
2. Without prior arrangement, missed exam ⇒ grade of zero.
3. Fail the exams ⇒ you might fail the course.

Schedule (Subject to change)

1: Aug 26: Administrivia
2: Aug 28: Introduction
3: Aug 30: The TINY Language
4: Sep 4: The TINY Language
5: Sep 6: Lexical Analysis
6: Sep 9: Lexical Analysis
7: Sep 11: ?
8: Sep 13: Lexical Analysis
9: Sep 16: Grammars
10: Sep 18: Grammars
11: Sep 20: Grammars
12: Sep 23: Syntactic Analysis
13: Sep 25: Syntactic Analysis
14: Sep 27: Syntactic Analysis
15: Sep 30: Syntactic Analysis
16: Oct 2: Semantic Analysis
17: Oct 4: Semantic Analysis
18: Oct 7: Semantic Analysis
19: Oct 9: Semantic Analysis
20: Oct 11: Semantic Analysis
21: Oct 14: Semantic Analysis
22: Oct 16: Review
23: Oct 18: Midterm exam
24: Oct 21: Semantic Analysis
25: Oct 23: Semantic Analysis
28: Oct 30: Interpretation
29: Nov 1: Interpretation
30: Nov 4: Code Generation
31: Nov 6: Code Generation
32: Nov 8: Code Generation
33: Nov 11: Code Generation
34: Nov 13: Optimization
35: Nov 15: Optimization
36: Nov 18: OO Languages
37: Nov 20: OO Languages
38: Nov 22: Garbage Collection
39: Nov 25: Garbage Collection
40: Nov 27: Extra
41: Dec 2: Extra
42: Dec 4: Extra
43: Dec 6: Extra
44: Dec 9: Extra
45: Dec 11: Extra
46: Dec 13: Extra
47: Dec 16: Review

August

<table>
<thead>
<tr>
<th>Su</th>
<th>Mo</th>
<th>Tu</th>
<th>We</th>
<th>Th</th>
<th>Fr</th>
<th>Sa</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

September

<table>
<thead>
<tr>
<th>Su</th>
<th>Mo</th>
<th>Tu</th>
<th>We</th>
<th>Th</th>
<th>Fr</th>
<th>Sa</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

Grading

- Grading (Subject to change)
  1. One midterm exam (10%)
  2. One final exam (30%)
  3. Seven programming assignments (60%)

- Midterm exam: Friday, October 18, in-class.
- Final exam: Friday, December 20, 11:00–13:00.

1. All exams are closed book.
2. Without prior arrangement, missed exam ⇒ grade of zero.
3. Fail the exams ⇒ you might fail the course.
Handouts & Other Material


2. I always make copies of my transparencies available to students. Note that
   - I do this to relieve you of having to take notes during lectures,
   - they are not substitutes for reading the textbook,
   - their primary purpose is to remind you of what you need to study for the exam.

Handouts & Other Material...

3. Various manuals and papers will be handed out during class. Extra copies can be picked up from the boxes outside my office.

4. Various information regarding the course (including postscript files of the handouts) can be found on the info-bahn: http://www.cs.arizona.edu/~collberg/Teaching/453/2002/index.html

Academic Integrity

- You will not
  1. turn in another student’s work as your own.
  2. use material from the web, textbooks, journals, etc. without giving the appropriate attribution.
  3. accept solutions from other students.
  4. give solutions to other students.
  5. tamper with graded papers or exams.
  6. collaborate with students outside your own team.

- Sanctions typically include:
  – grade reduction, course failure, suspension, expulsion.

- I take this stuff seriously.

Academic Integrity...

- Students who violate the Code are also subject to possible sanctions imposed by the Dean of Students office.

- Submitted solutions will be compared with each other, as well as with solutions from previous semesters.

- All students involved in collusion are equally culpable:
  1. Do not give another student access to your account.
  2. Do not leave printouts in the recycling bin.
  3. Pick up your printouts promptly.
  4. Do not leave your workstation unattended.

If you suspect that your work has been compromised notify me immediately.
Why Am I a Compiler Jock?

From: Dwight VandenBerghe, dwight@pentasoft.com
Newsgroups: comp.compilers

I was seventeen and had a lot of time on my hands. I was a programmer for the US Marines, stationed in Da Nang, Vietnam, in the mid sixties, and during the long nights I would go through the IBM microfiche - they kept a full set in the computer room, all the manuals and all the source code for all their tools. I got interested in the COBOL compiler, and I read through the assembler code for it. I still remember reading the instructions that scanned in an integer. It just thrilled me, seeing how that magic was done.

Writing a compiler, you go back and forth, from the infinite to the practical, over and over again. And now, with the advent of functional programming into my life, there is an elegance to the code that I haven’t seen before. The functional style seems to me to be made for us compiler jockeys.

And the advances that have been made in this field: BURS theory, the SUIF system, transformational tree rewrite systems, higher-order attribute grammars, the exciting new work with LL(infinite) parser generators ... it’s simply thrilling. Nothing else compares, at least, for me. Applications suck; device drivers are tedious; operating systems and file management and databases are boring.

After 31 years, I’ve pretty much done it all, and if it wasn’t for compilers I’d be bored stiff. Yet here it is, the wee hours yet again. Last night I finally hauled myself to sleep at a little after 2AM, because I found some great papers on polymorphic type theory on some server in the UK. So it goes.

Dwight

I stayed up late into the early morning, trying to figure it all out.

It’s 3:10AM here now, and I’ve just finished downloading some papers from Norway on attribute grammars in ML. I have to go print them out. I’m dead tired, but this stuff is so exciting that sleep will have to wait. I’m 48 now, and my youngest boy is almost seventeen. My grandkids are great, and my wife is a gem, but you know, I hope that I am still cheating sleep for another couple of decades, because I feel seventeen, not nearly fifty. Compilers are magical. Compilers are like the two opposing mirrors in the funhouse - you see infinity, yet you are clearly finite.
What's available in Compiler Jobs?

From: Clifford Click, cliff.click@Eng.Sun.COM
Newsgroups: comp.compilers

> 2. What do you think is the future for this market?
> 3. Is the demand growing?

Skilled optimizing compiler writers are in high demand. The market is small, but the labor pool is even smaller. (The demand for people who make functioning parsers is much smaller, probably due to the ease of using lex & yacc).

The future for this market? Welll... as long as chip architects and language designers are alive I got a job. Merced looks like full time employment for 20+ people for 5+ years (OUTSIDE of Intel & HP; they probably have double that number already and are always looking for more). [Again, I'm talking about optimizing compiler writers; big optimizers grow big support groups that hire more people in nearby fields]

And yes, I believe the demand is growing.
<table>
<thead>
<tr>
<th><strong>Job Title</strong></th>
<th>Compiler Designer - Permanent</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Job Location</strong></td>
<td>Ottawa, Ontario, CANADA</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Working with our core technology team you will develop compilers for our Tamper-Resistent Software (TRS).</td>
</tr>
<tr>
<td><strong>Our website</strong></td>
<td><a href="http://www.cloakware.com">http://www.cloakware.com</a></td>
</tr>
<tr>
<td><strong>Qualifications</strong></td>
<td>Cloakware is seeking expert Compiler Writers, with knowledge any or a combination of the following: Optimization, Program Transformation, Combinatorics, Obfuscation Techniques, Program slicing Techniques, Numerical Analysis, Semetric Representations for Control- and Data-Flow, Control- and Data-Flow Dependencies</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Job Title</strong></th>
<th>Post-Doctoral Researcher</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Several postdoctoral research positions are available, working on new approaches to mobile code; this is compiler-related research.</td>
</tr>
<tr>
<td><strong>Qualifications</strong></td>
<td>Applicants must have attained a Ph.D. in Computer Science or related field and have prior experience with software systems, possess demonstrated familiarity with code generation for modern RISC architectures, and the Ada and Java programming languages.</td>
</tr>
<tr>
<td><strong>Website</strong></td>
<td><a href="http://www.ics.uci.edu/~franz">www.ics.uci.edu/~franz</a></td>
</tr>
</tbody>
</table>