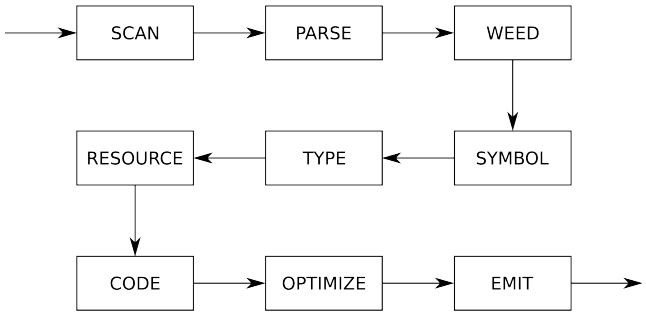
Course Summary

COMP 520: Compiler Design (4 credits)

Alexander Krolik

alexander.krolik@mail.mcgill.ca

MWF 13:30-14:30, MD 279



McCompiley

Announcements (Friday, Apr 7/Monday, Apr 10)

- Final Report due **Tuesday, April 11th 11:59PM** on GitHub
- Peephole Optimizer due **Tuesday, April 11th 11:59PM** on GitHub/myCourses
- Course evaluations! Please let us know what you think of the course!
- Project meeting signup in myCourses (under groups tab). 1 person per team signup

Review classes:

- **Friday:** Scanner – bytecode generation
- **Monday:** Optimization – native code generation

Why did we learn about Compilers?

How does learning about compilers change your view of Programming Language Design?

If you were to select a compiler/language toolkit for another compiler project, what would you choose?

Structure of Final Exam (100 points)

1. Scanners and Parsers (20 points)
2. Typechecking (10 points)
3. Bytecode Generation (15 points)
4. Optimization (15 points)
5. Native Code Generation (10 points)
6. Garbage Collection (10 points)
7. GoLite Project (20 points)

- Emphasis on application
- Virtual machines and tinylang example included

Tips

- Review Vincent's midterm review.
- Review the midterm, if you got something wrong, go back to the notes and figure out the right answer.
- Organize your answers - make is easy for the grader to find your answers.
- Write neatly.
- Start each question on a new page.
- Don't squish in your answers to make a lot fit on one page.
- Be precise.

All the midterm Material

- All the topics from the midterm will also be possible on the final.
- Review, scanners, parsers, weeders, type checking and symbol tables.

Scanners

- You should be comfortable implementing a scanner in flex or SableCC
- Know the limitations and languages that be recognized by regular expressions/DFAs/NFAs

Parsers

- Know what makes a grammar ambiguous, and how to write unambiguous context-free grammars
- You should know what limitations there are on context-free grammars that are handled by weeders

Typechecking

- Know how to read and write type rules and symbol tables
- Know how type rules are implemented in compilers
- Know how static scoping is implemented in symbol tables

Bytecode Generation

- Know how to generate bytecode: the different sections, types, bytecodes, labels, etc.
- Know how the baby stack works to compute output (some internals of the JVM)

Optimization

- Understand why and how generated code is inefficient
- Know how and when to optimize code, while maintaining the original semantics (soundness)
- Peephole optimization (not static analysis)

Garbage Collection

- Understand workings of garbage collection techniques
- Know the rules for reference counting, mark and sweep, and stop and copy
- Know what makes record live and dead

Native Code Generation

- Know the 3 different register allocation schemes covered in this class (naïve, fixed, and basic block)
- Understand the register allocations for each scheme
- Know the advantages/disadvantages to each technique, and the situations in which you would notice the difference

Thanks...

- To Hanfeng and Prabhjot, they worked hard as TAs
- To the class - you worked hard all term
- To Laurie and Vince, for help and support all semester