Compiling Techniques
Lecture 1: Overview

Christophe Dubach
Overview

- Course Organisation
- Syllabus
- Context
Essential Facts

- Lecturer: Christophe Dubach (christophe.dubach@ed.ac.uk)
- Office hours: Thursdays 11am-12pm
- Textbook:
  - Keith Cooper & Linda Torczon: Engineering a Compiler
    Elsevier, 2004
  - Textbook can be reused in UG4 Compiler Optimisation
- Website: http://www.inf.ed.ac.uk/teaching/courses/ct/
- Will include slides a day before the lecture
Exam & Coursework

- Exam (75%)

- Coursework (25%)
  - Small C Frontend written in Java
  - Java Bytecode Backend (executable in JVM)
Syllabus

- Overview
- Scanning
- Parsing
- Context Sensitive Analysis
- Inner Workings of Compiled Code
- Code Generation
  - Instruction Selection
  - Register Allocation
- Selected Advanced Topics
Class-Taking Technique

- Extensive use of projected material
- Interaction encouraged!
- Read the textbook alongside the course
- Complements the lectures
- In-depth coverage of material
- Not a programming course!
- Start the practical early
Compilers

What is a compiler?

- A program that translates an executable program in one language into an executable program in another language
- The compiler might improve the program, in some way

What is an interpreter?

- A program that directly execute an executable program, producing the results of executing that program

Examples

- C is typically compiled
- R is typically interpreted
- Java is compiled to bytecodes, then interpreted within a Java Virtual Machine (JVM) or a hybrid strategy is used: Just-in-time compilation
A Broader View

- Compiler Technology = Off-Line Processing
  - Goals: improved performance and language usability
  - Making it practical to use the full power of the language
  - Trade-off: preprocessing time versus execution time (or space)
  - Rule: performance of both compiler and application must be acceptable to the end user

- Examples
  - Macro expansion / Preprocessing
  - Database query optimisation
  - Javascript just-in-time compilation
  - Emulation acceleration: TransMeta “code morphing”
Why study compilation?

- Compilers are important system software components
  - They are intimately interconnected with architecture, systems, programming methodology, and language design

- Compilers include many applications of theory to practice
  - Scanning, parsing, static analysis, instruction selection

- Many practical applications have embedded languages
  - Commands, macros, formatting tags ...

- Many applications have input formats that look like languages
  - Matlab, Mathematica

- Writing a compiler exposes practical algorithmic & engineering issues
  - Approximating hard problems; efficiency & scalability
Intrinsic Interest

Compiler construction involves ideas from many different parts of computer science.

<table>
<thead>
<tr>
<th>Artificial intelligence</th>
<th>Greedy algorithms, Heuristic search techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td>Graph algorithms, union-find, Dynamic programming</td>
</tr>
<tr>
<td>Theory</td>
<td>DFAs &amp; PDAs, pattern matching, Fixed-point algorithms</td>
</tr>
<tr>
<td>Systems</td>
<td>Allocation &amp; naming, Synchronization, locality</td>
</tr>
<tr>
<td>Architecture</td>
<td>Pipeline &amp; hierarchy management, Instruction set use</td>
</tr>
</tbody>
</table>
Compiler construction poses challenging and interesting problems:

- Compilers must do a lot but also run fast
- Compilers have primary responsibility for run-time performance
- Compilers are responsible for making it acceptable to use the full power of the programming language
- Computer architects perpetually create new challenges for the compiler by building more complex machines
- Compilers must hide that complexity from the programmer
- Success requires mastery of complex interactions
Making Languages Usable

It was our belief that if FORTRAN, during its first months, were to translate any reasonable "scientific" source program into an object program only half as fast as its hand coded counterpart, then acceptance of our system would be in serious danger.

[...]

I believe that had we failed to produce efficient programs, the widespread use of languages like FORTRAN would have been seriously delayed.

John Backus
Preview

- The View from 35000 Feet
- How a compiler works
- What I think is important
- What is hard and what is easy