

Overview

We will cover the following topics:

1. Lexical Analysis (*scanning*)
2. Syntax Analysis (*parsing*)
3. Context-sensitive Analysis
4. Intermediate Representations
5. Code Generation
6. Code Improvement Techniques

The textbook for this course is “*Compilers: Principles, Techniques, and Tools*” by Aho, Sethi, and Ullman.

Overhead transparencies are available in postscript format on the World Wide Web from <http://www.cs.tamu.edu/rwenger/>

Class-taking technique

I use overhead transparencies

- all transparencies are on the Web
- you should still take some notes

I'll tell you where we are in the book

- I don't lecture directly from the book
- *You* need to read the book
- you need to start by refreshing your C++ skills

Compilers

What is a compiler?

- a program that translates an *executable* program in one language into an *executable* program in another language
- the compiler typically *lowers* the level of abstraction of the program
- we expect the program produced by the compiler to be *better*, in some way, than the original

Motivation

Why build compilers?

Why study compiler construction?

Why attend class?

Reasons:

- compilers provide an essential interface between applications and architectures
- compilers embody a wide range of theoretical techniques
- compiler construction teaches programming and software engineering skills

Role of Compilers

High-level programming languages

- increase programmer productivity
- better maintenance
- portable

Low-level machine details

- instruction selection
- addressing modes
- pipelines
- registers & cache
- instruction-level parallelism

Compilers are needed to efficiently bridge the gap!

Isn't it a solved problem?

Machines have continued to change since they have been invented

Changes in architecture \Rightarrow changes in compilers

- new features present new problems
- changing costs lead to different concerns
- must re-engineer well-known solutions

Significant differences in performance

Interest

Compiler construction shows us a microcosmic view of computer science.

<i>artificial intelligence</i>	greedy algorithms learning algorithms
<i>algorithms</i>	graph algorithms union-find network flows dynamic programming
<i>theory</i>	<i>dfa's</i> for scanning parser generators lattice theory for analysis
<i>systems</i>	allocation and naming locality synchronization
<i>architecture</i>	pipeline management memory hierarchy management instruction set use

Inside a compiler, all these things come together.

As a result, compiler construction is challenging and fun.

Compiler Construction

Compilers are large, complex pieces of software.

By working on compilers, you'll learn to use

- programming tools (compilers, debuggers)
- program-generation tools (flex, yacc, bison)
- software libraries (sets, collections)
- simulators (spim)

Hopefully you will also enhance your software engineering skills.

Experience

You have used several compilers.

What qualities do you want in a compiler?

Here is a list:

1. Correct code
2. Output runs fast
3. Compiler runs fast
4. Compile time proportional to program size
5. Support for separate compilation
6. Good diagnostics for syntax errors
7. Works well with the debugger
8. Good diagnostics for flow anomalies
9. Cross language calls
10. Consistent, predictable optimization

My Biases

My own research:

- compiling for parallel architectures
- parallelizing C++
- exploring the language/compiler/runtime/architecture interface
- adaptive parallel processing: customizing compilation process to application

Some Advice

Compiler construction is one of the most time-consuming undergraduate level programming courses.

Be prepared to spend a lot of time on projects if you don't have much programming experience in UNIX and C++. Be prepared to lose a lot of sleep if you don't start projects early!

Next Class

I'll try to give you an overview of how a compiler works and what I think is important.

Things to do:

- read Aho, Sethi, and Ullman, Chapters 1, 2, 3
- read the manual page for *flex*
- learn to use a Web browser such as Netscape