Introduction to Compiler Construction

Compilation: Preliminaries

Outline

Compilers

Compilers

A compiler translates a source language program into a target language program

Compilers

A compiler translates a source language program into a target language program

Examples (source language): C, Java, *j--*, *iota*

Compilers

A compiler translates a source language program into a target language program

Examples (source language): C, Java, *j--*, *iota*

Examples (target language): Intel x86 instructions, JVM instructions, Marvin Machine instructions

A programming language specification consists of

- Syntax of tokens
- Syntax of constructs such as classes, methods, statements, and expressions
- Semantics (ie, meaning) of the constructs

A machine's instruction set along with its behavior is referred to as its architecture

A machine's instruction set along with its behavior is referred to as its architecture

Examples

- Intel x86
- Java Virtual Machine (JVM)
- Marvin Machine

Interpreters

Interpreters

An interpreter executes a source language program directly

Interpreters

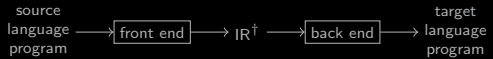
An interpreter executes a source language program directly

Examples: Bash, Python

Phases of Compilation

Phases of Compilation

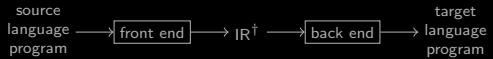
A compiler can be broken into a front end and a back end



† Intermediate Representation

Phases of Compilation

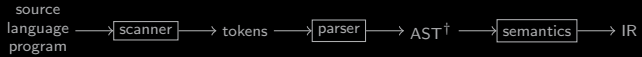
A compiler can be broken into a front end and a back end



† Intermediate Representation

Separating the front end from the back end enables code re-use

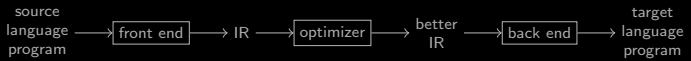
The front end can be decomposed into a sequence of analysis phases



† Abstract Syntax Tree

The back end can be decomposed into a sequence of synthesis phases

A compiler sometimes has an optimizer between the front end and the back end



Why Study Compilers?

Why Study Compilers?

Compilers are larger programs than the ones you have written so far

Why Study Compilers?

- Compilers are larger programs than the ones you have written so far
- Compilers make use of things you have learned about earlier

Why Study Compilers?

- Compilers are larger programs than the ones you have written so far
- Compilers make use of things you have learned about earlier
- You learn a lot about the source language and the target machine

Why Study Compilers?

- Compilers are larger programs than the ones you have written so far
- Compilers make use of things you have learned about earlier
- You learn a lot about the source language and the target machine
- Compilers are still being written for new languages and targeted to new architectures

Why Study Compilers?

- Compilers are larger programs than the ones you have written so far
- Compilers make use of things you have learned about earlier
- You learn a lot about the source language and the target machine
- Compilers are still being written for new languages and targeted to new architectures
- There is a good mix of theory and practice

Why Study Compilers?

- Compilers are larger programs than the ones you have written so far
- Compilers make use of things you have learned about earlier
- You learn a lot about the source language and the target machine
- Compilers are still being written for new languages and targeted to new architectures
- There is a good mix of theory and practice
- Compiler writing is a case study in software engineering

Why Study Compilers?

- Compilers are larger programs than the ones you have written so far
- Compilers make use of things you have learned about earlier
- You learn a lot about the source language and the target machine
- Compilers are still being written for new languages and targeted to new architectures
- There is a good mix of theory and practice
- Compiler writing is a case study in software engineering
- Compilers are programs and writing programs is fun