Compiler Construction

Compilation Strategies

Objectives

The main goal is to execute high level programming languages.

- How to obtain an efficient (optimized) executable?
- How to have an easy to debug and deploy language?

Remember! Any programming language can be either interpreted or compiled!

Interpreters

An interpretor reads code and immediately executes that code. It works by fetching, analyzing, and executing one instruction at a time.

- **Portable:** no need to compile the code for a targeted architecture
- **Impact on the input language**: first-order eval function, dynamic typing, dynamic scoping

Compilation

Programs are compiled (translated) into native code during a compilation process.

- **Ahead-of-the-time (AOT) compilation**: produce a binary file that can be run in the targeted architecture.
- **Just-in-time (JIT) compilation**: involves compilation during execution of a program (at runtime) rather than before execution.

Bytecode strategy

Bytecode is a portable low level code. Contrary the assembly language, there is no existing physical machine that understand this language.

Bytecode can be:

- compiled
- interpreted
- executed by a virtual machine

Virtual Machine

A virtual machine is a platform-independent programming environment that abstracts away details of the underlying hardware or operating system. It allows a program to execute in the same way on any platform.

Transpilation strategy

Transpile the input language into an existing language. Then use the existing compiler for this language.

Summary