Overview of Compilers

Part 1

What is a Compiler?

- A **compiler** is a computer program that translates a program in a **source language** into an equivalent program in a **target language**.

- A **source program/code** is a program/code written in the source language, which is usually a high-level language.

- A **target program/code** is a program/code written in the target language, which often is a machine language or an intermediate code.

Process of Compiling

- **Stream of characters**
- **Stream of tokens**
- **Parse/syntax tree**
- **Annotated tree**
- **Intermediate code**
- **Intermediate code**
- **Code optimization**
- **Code generator**
- **Target code**
Scanning

- A scanner reads a stream of characters and puts them together into some meaningful (with respect to the source language) units called tokens.
- It produces a stream of tokens for the next phase of compiler.

Parsing

- A parser gets a stream of tokens from the scanner, and determines if the syntax (structure) of the program is correct according to the (context-free) grammar of the source language.
- Then, it produces a data structure, called a parse tree or an abstract syntax tree, which describes the syntactic structure of the program.

Semantic analysis

- It gets the parse tree from the parser together with information about some syntactic elements
- It determines if the semantics or meaning of the program is correct.
- This part deals with static semantic.
  - semantic of programs that can be checked by reading off from the program only.
  - syntax of the language which cannot be described in context-free grammar.
- Mostly, a semantic analyzer does type checking.
- It modifies the parse tree in order to get that (static) semantically correct code.
Intermediate code generation

- An intermediate code generator
  - takes a parse tree from the semantic analyzer
  - generates a program in the intermediate language.
- In some compilers, a source program is translated into an intermediate code first and then the intermediate code is translated into the target language.
- In other compilers, a source program is translated directly into the target language.

Intermediate code generation (cont’d)

- Using intermediate code is beneficial when compilers which translates a single source language to many target languages are required.
  - The front-end of a compiler – scanner to intermediate code generator – can be used for every compilers.
  - Different back-ends – code optimizer and code generator– is required for each target language.
- One of the popular intermediate code is three-address code. A three-address code instruction is in the form of \( x = y \text{ op } z \).

Code optimization

- Replacing an inefficient sequence of instructions with a better sequence of instructions.
- Sometimes called code improvement.
- Code optimization can be done:
  - after semantic analyzing
    • performed on a parse tree
  - after intermediate code generation
    • performed on an intermediate code
  - after code generation
    • performed on a target code
Code generation

• A code generator
  – takes either an intermediate code or a parse tree
  – produces a target program.