Table of Contents

Introduction

Code generation can be considered as the final phase of compilation.

Through post code generation, optimization process can be applied on the code, but that can be seen as a part of code generation phase itself.

The code generated by the compiler is an object code of some lower-level programming language.

For example, Assembly language.

We have seen that the source code written in a higher-level language is transformed into a lower-level language that results in a lower-level object code, which should have the following minimum properties:

- It should carry the exact meaning of the source code.
- It should be efficient in terms of CPU usage and memory management.

Design of code generator

Definition

The target program: The output of code generator is target program.
Memory management: Mapping names in the source program to addresses of data objects in Design done by run time memory is front end & code generator

A code generator is expected to have an understanding of the target machine's runtime environment and its instruction set.

The code generator should take the following things into consideration to generate the code:

1. Target language

The code generator has to be aware of the nature of the target language for which the code is to be transformed. That language may facilitate some machine-specific instructions to help the compiler generate the code in a more convenient way. The target machine can have either CISC or RISC processor architecture.

2. IR type

Syntax Tree (AST) Intermediate representation has various forms. It can be in Abstract structure, Reverse Polish Notation, or 3-address code.

3. Selection of instruction

The code generator takes Intermediate Representation as input and converts (maps) it into target machine's instruction set. One representation can have many ways (instructions) to

convert it, so it becomes the responsibility of the code generator to choose the appropriate instructions wisely.

4. Register allocation

A program has a number of values to be maintained during the execution. The target machine’s architecture may not allow all of the values to be kept in the CPU memory or registers. Code generator decides what values to keep in the registers. Also, it decides the registers to be used to keep these values.

5. Ordering of instructions

At last, the code generator decides the order in which the instruction will be executed. It creates schedules for instructions to execute them.