Code Generation

A code generator has three primary tasks: instruction selection, register allocation and assignment, and instruction ordering.

Issues in the design of a code generator

While the details are dependent on the specifics of the intermediate representation, the target language, and the run-time system, tasks such as instruction selection, register allocation and assignment, and instruction ordering are encountered in the design of almost all code generators.

The most important criterion for a code generator is that it produce correct code. Correctness takes on special significance because of the number of special cases that a code generator might face

Input to the code generator

The input to the code generator is the intermediate representation of the source program produced by the front end, along with information in the symbol table that is used to determine the run-time addresses of the data objects denoted by the names in the IR.

The target program

The instruction-set architecture of the target machine has a significant impact on the difficulty of constructing a good code generator that produces high-quality machine code. The most common target-machine architectures are RISC,CISC, and stack based.

A RISC machine typically has many registers, three-address instructions, simple addressing modes, and a relatively simple instruction-set architecture.
CISC machine typically has few registers, two-address instructions, a variety of addressing modes, several register classes, variable-length instructions, and instructions with side effect.

Instruction Selection

The code generator must map the IR program into a code sequence that can be executed by the target machine. The complexity of performing this mapping is determine by factors such as:
-the level of the IR
-the nature of the instruction-set architecture
-the desired quality of the generated code

Id the IR is high level, the code generator may translate each IR statement into a sequence of machine instructions using code templates.Such statement -by-statement code generation, however, often produces poor code that need further optimization. If the IR reflects some of the

low-level details of the underlying machine, then the code generator can use this information to generate more efficient code sequences.

Register Allocation

Registers are the fastest computational unit on the target machine, but we usually do not have enough of them to hold all values. Values not held in registers need to reside in memory. The use of registers is divided into two problems:

1.-Register allocation, during which we select the set of variables that will reside in registers at each point in the program.

2.- Register assignment, during which we pick the specific register that a variable will reside in.

Evaluation Order

The order in which imputations are performed can affect the efficiency of the target code. As we shall see, some computation orders require fewer registers to hold intermediate results than others. However , picking a best order in the general case is a difficult NP-complete problem.