Code Generation: Intermediate Languages

Abstract of presentation

(VYPe course 2011/2012)

Jiří Matička, xmatic00
Michal Minář, xminar06

The major purpose of compilers is to transform the input source program written in language „A“ to output target program in language „B“ with the same semantic. An assembly language is commonly used as the output program language. It is too difficult to compile the input program to assembly language directly. That is the reason why compilers often generate an intermediate language at first and then translate it to the assembly language. There are several advantages of this approach. First, we can design an intermediate language to ease the subsequent optimalization. It is extremely advantageous for the final binary image, because it can be optimized in a better manner. Second, the translation of an intermediate language to a binary code is usually done by a separate compilation phase called back end. As a result, we can provide several back ends for different target languages. There are typically three forms of intermediate languages. Their advatages and disadvantages will be discussed further below.

One specific intermediate language is called C-code. It is a subset of the C language with some restrictions and relaxed syntax. It is possible to think of the C-code as of a model assembly language that is optimized for virtual machine. This is close to an ideal (but also nonexistent) computer that deals with the code. The virtual machine has a set of registers, a stack and a memory. The stack represents memory that can be used by the running program. The memory is partitioned into several segments, one of them can be the stack segment. All segments and their particular meaning will be discussed in detail later on. The talk will also focus on a relocatable object module which is a code containing place holders. It makes this code multiplatform. The next memory issue is called alignment. Particular data types do not allow to place their instances on arbitrary positions in the memory and they need to be aligned. This is very important when it comes to structures. Wrong order of elements in structure will cause a multiple memory occupation. We will show address modes involving direct and indirect mode and address arithmetic that involves pointers. The last topic will be about stack frame. It explains handling of subroutine arguments and local variables.

The first part of the presentation explains what is an intermediate code and why it is used. Then, we focus on one specific intermediate language called C-code. After processing, the code is served by a virtual machine. The architecture and functionality of this machine is discussed in the last part of the presentation.