32. The Compiler Design Handbook - Chapter 11 (SSA Form)

Authors: Končický Jaromír, Sokl Karel

Abstract

During translation a source program code into a target machine code, we usually want to optimize the code. Simply we want the final program to be smaller and run faster. So the target code shall not contain any dead code, redundant or unnecessary code and so on. To achieve this, many optimization techniques were developed. Our main goal here is to make the optimization methods faster and more efficient, so the compilation does not take too much time, and the performed optimizations significantly improve the program performance.

One possible way how to make the optimizations faster and easier is to modify the form of internal code representation in some appropriate way. The internal form of code we are using and discussing here is called SSA (Static Single Assignment) form. Simply described, in this form, each variable is assigned a value only once, so for each assignment in the program, a new individual variable is created.

In this presentation we will talk more about the SSA form, and we will briefly describe how a code is converted from a traditional form into SSA form. We will also state the variations of SSA form as Minimal SSA, Pruned SSA and Semi-pruned SSA. The main goal of our presentation is to describe the main advantages of optimizations of a code in SSA form, talking about some common optimizations like Conditional constant propagation, Dead code elimination etc.

During the presentation we will show some demonstration examples, and discuss the differences between normal-form and SSA-form code optimizations and their effectiveness.