Introduction to LLVM Compiler Infrastructure

Robert Baručák, Jakub Šulek

Abstract

Goal of this presentation is to introduce audience to the principles of LLVM (Low Level Virtual Machine) compiler. LLVM is modern, modular and easily modifiable compiling architecture, it is not a virtual machine in classic meaning of the word. Started as a research project at University of Ilinois, LLVM is now heavilly used not only in academic, but in commercial sphere as well. Main difference between LLVM and common used older compilers is, that LLVM was designed as the set of reusable libraries and well-defined interfaces rather than monolithic application.

Design of LLVM is based on classical three phase design, the basic components are front-end, optimizer, backend. Front-ends are used to transform input code to intermediate representation (LLVM IR).

This representation is independent from target machine and currently there are front-ends for many languages. Basically LLVM IR is static single assignment based representation in form of a low-level RISC-like instruction set. It serves as the only interface to optimizer. Designed to represent all high level languages, it emphasizes type safety and flexibility. Important feature of the design is, that IR is used in three different forms: as an in-memory compiler IR, as an on-disk bitcode representation (for usage in Just-In-Time compilers), and as an assembly language representation.

In the presentation we want to focus on the most important part of LLVM Infrastructure– versatility of this infrastructure. It can be easily used on any problem from building a C compiler for specialized architecture to building a JVM-like virtual machine.

The optimizer is organized as a pipeline of optimization passes, where each pass can modify the input. For example inliner, loop invariant code motion etc. This can be useful when we need to implement specific transformations of the input code.

The backend serves for generating target specific machine code. The code generation problem is split into subproblems like instruction selection, register allocation, etc..

We will present interesting projects utilising LLVM infrastructure as well.