深入浅出让你理解什么是LLVM

Coffee_LaFa (/u/533b59db5047) ＋关注
3.0 2018.08.12 21:55* 字数 1609 阅读 13987 评论 2 喜欢 26
(/u/533b59db5047)

什么是LLVM (https://llvm.org/)

> LLVM项目是模块化、可重用的编译器以及工具链技术的集合。

美国计算机协会 (ACM) 将其2012 年软件系统奖项颁给了LLVM，之前曾经获得此奖项的软件和技术包括:Java、Apache、 Mosaic、the World Wide Web、Smalltalk、UNIX、Eclipse等等
创始人:Chris Lattner，亦是Swift之父

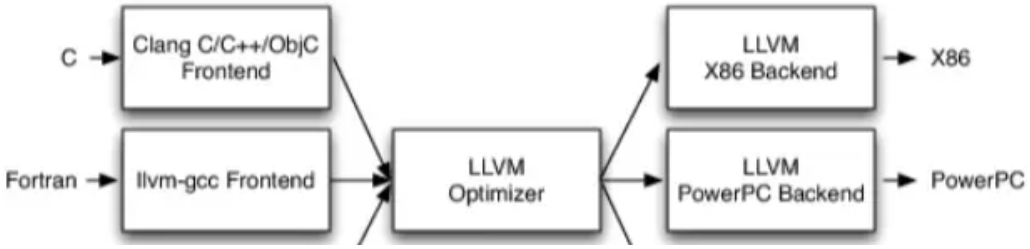
备注：有些文章把LLVM当做Low Level Virtual Machine(低级虚拟机)的缩写简称，官方描述如下
The name "LLVM" itself is not an acronym; it is the full name of the project. “LLVM”这个名称本身不是首字母缩略词; 它是项目的全名。

传统的编译器架构

传统编译器架构

- **Frontend:前端**
  词法分析、语法分析、语义分析、生成中间代码
- **Optimizer:优化器**
  中间代码优化
- **Backend:后端**
  生成机器码

LLVM架构



LLVM架构

- 不同的前端后端使用统一的中间代码LLVM Intermediate Representation (LLVM IR)
- 如果需要支持一种新的编程语言，那么只需要实现一个新的前端
- 如果需要支持一种新的硬件设备，那么只需要实现一个新的后端
- 优化阶段是一个通用的阶段，它针对的是统一的LLVM IR，不论是支持新的编程语言，还是支持新的硬件设备，都不需要对优化阶段做修改
- 相比之下，GCC的前端和后端没分得太开，前端后端耦合在了一起。所以GCC为了支持一门新的语言，或者为了支持一个新的目标平台，就 变得特别困难
- LLVM现在被作为实现各种静态和运行时编译语言的通用基础结构(GCC家族、Java、.NET、Python、Ruby、Scheme、Haskell、D等)

什么是Clang (http://clang.llvm.org/)

LLVM项目的一个子项目，基于LLVM架构的C/C++/Objective-C编译器前端。

相比于GCC，Clang具有如下优点

- 编译速度快:在某些平台上，Clang的编译速度显著的快过GCC(Debug模式下编译OC速度比GGC快3倍)
- 占用内存小:Clang生成的AST所占用的内存是GCC的五分之一左右
- 模块化设计:Clang采用基于库的模块化设计，易于 IDE 集成及其他用途的重用
- 诊断信息可读性强:在编译过程中，Clang 创建并保留了大量详细的元数据(metadata)，有利于调试和错误报告
- 设计清晰简单，容易理解，易于扩展增强

Clang与LLVM关系

Clang与LLVM

LLVM整体架构，前端用的是clang，广义的LLVM是指整个LLVM架构，一般狭义的LLVM指的是LLVM后端（包含代码优化和目标代码生成）。

源代码（c/c++）经过clang--> 中间代码(经过一系列的优化，优化用的是Pass) --> 机器码

OC源文件的编译过程

这里用Xcode创建一个Test项目，然后cd到main.m的上一路径。
命令行查看编译的过程:$ clang -ccc-print-phases main.m

```
$ clang -ccc-print-phases main.m

0: input, "main.m", objective-c
1: preprocessor, {0}, objective-c-cpp-output
2: compiler, {1}, ir
3: backend, {2}, assembler
4: assembler, {3}, object
5: linker, {4}, image
6: bind-arch, "x86_64", {5}, image
```

0.找到main.m文件
1.预处理器，处理include、import、宏定义
2.编译器编译，编译成ir中间代码
3.后端，生成目标代码
4.汇编
5.链接其他动态库静态库
6.编译成适合某个架构的代码

查看preprocessor(预处理)的结果:$ clang -E main.m
这个命令敲出，终端就会打印许多信息，大致如下：

```
# 1 "main.m"
# 1 "<built-in>" 1
# 1 "<built-in>" 3
# 353 "<built-in>" 3
# 1 "<command line>" 1
# 1 "<built-in>" 2
# 1 "main.m" 2
.
.
.
  int main(int argc, const char * argv[]) {
@autoreleasepool {
    NSLog(@"Hello, World!");
}
return 0;
}
```