# ****LLVM简介：****

LLVM是构架编译器(compiler)的框架系统，以C++编写而成，用于优化以任意程序语言编写的程序的编译时间(compile-time)、链接时间(link-time)、运行时间(run-time)以及空闲时间(idle-time)，对开发者保持开放，并兼容已有脚本。

  在理解LLVM时，我们可以认为它包括了一个狭义的LLVM和一个广义的LLVM。广义的LLVM其实就是指整个LLVM编译器架构，包括了前端、后端、优化器、众多的库函数以及很多的模块；而狭义的LLVM其实就是聚焦于编译器后端功能（代码生成、代码优化、JIT等）的一系列模块和库。

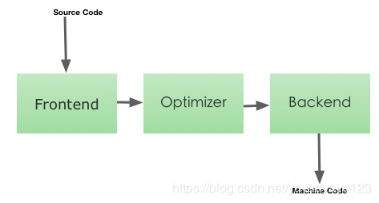
# ****LLVM优势:****

传统编译器分三个阶段：

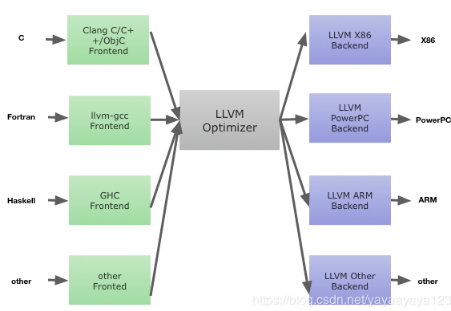
前端（Frontend）-- 优化器（Optimizer）-- 后端（Backend）

前端负责分析源代码，可以检查语法级错误，并构建针对语言的抽象语法树（AST）；抽象语法树可以进一步转换为优化，最终转为新的表示方式，然后再交给让优化器和后端处理；

最终由后端生成可执行的机器码。

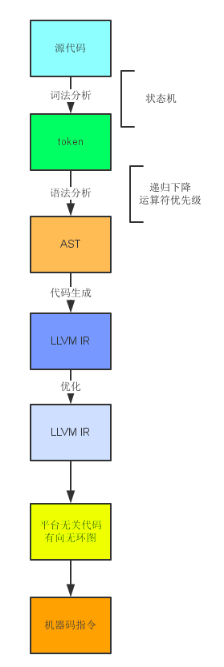


llvm也分三个阶段，但是设计上略微的有些区别， LLVM不同的就是对于不同的语言它都提供了同一种中间表示：

  
前端可以使用不同的编译工具对代码文件做词法分析以形成抽象语法树AST，然后将分析好的代码转换成LLVM的中间表示IR（intermediate representation）；中间部分的优化器只对中间表示IR操作，通过一系列的pass对IR做优化；后端负责将优化好的IR解释成对应平台的机器码。LLVM的优点在于，中间表示IR代码编写良好，而且不同的前端语言最终都转换成同一种的IR。

为什么使用三段式设计？优势在哪里？首先解决一个很大的问题：假如有N种语言（C、OC、C++、Swift...）的前端，同时也有M个架构（模拟器、arm64、x86...）的target，是否就需要N\*M个编译器？三段式架构的价值就提现出来了，通过共享优化器的中转，很好的解决了这个问题。

# ****LLVM编译工具链编译流程：****



LLVM编译一个源文件的过程：预处理 -> 词法分析 -> Token -> 语法分析 -> AST -> 代码生成 -> LLVM IR -> 优化 -> 生成汇编代码 -> Link -> 目标文件

完全需要我们手工，或者依靠其他工具如lex, yacc来做的事情，是从源代码到token的词法分析和从token到AST的语法分析；词法分析的输出是将源代码解析成一个个的token。这些token就是有类型和值的一些小单元，比如是关键字，还是数字，还是标识符，从AST转LLVM开始，LLVM就开始提供一系列的工具帮助我们快速开发。从IR(中间指令代码)到DAG(有向无环图)再到机器指令，针对常用的平台，LLVM有完善的后端。也就是说，我们只要完成了到IR这一步，后面的工作我们就享有和Clang一样的先进生产力了。

# ****Clang和LLVM关系：****

    Clang是一个C++编写、基于LLVM、发布于LLVM BSD许可证下的C/C++/Objective-C/Objective-C++编译器。那么为什么已经有了GCC还要开发Clang呢？Clang相比于GCC有什么优势呢？   其实，这也是Clang当初在设计开发的时候所主要考虑的原因。Clang是一个高度模块化开发的轻量级编译器，它的编译速度快、占用内存小、非常方便进行二次开发。

CodeGen负责将语法树从顶至下遍历，翻译成LLVM IR，LLVM IR是Frontend的输出，也是LLVM Backerend的输入，桥接前后端。

# ****LLVM命令：****

1. ****可以使用 llc 将 LLVM 字节代码转换成特定于平台的汇编代码****
2. ****lli 可以通过解释器或使用高级选项中的即时 (JIT) 编译器执行此工作****
3. ****llvm-gcc 是 GNU Compiler Collection (gcc) 的修改版本，可以在使用 -S -emit-llvm 选项运行时会生成 LLVM 字节代码。****
4. ****编译指令：****

clang -c -emit-llvm test1.c -o test1.bc  编译产生字节码

clang -S -emit-llvm test.c -o test.ll     编译产生可视化字节码

llvm-dis test1.bc test1.ll                        bc字节码转为可视化字节码ll

llvm-as test1.ll test1.bc                         可视化字节码转为字节码bc