Clang::CompilerInstance定义：用来管理clang编译器实例。包括预处理器，目标平台信息，AST上下文，编译选项，源代码管理器，语义分析对象，

常用操作：

getCodeGenOpts():

Clang::PresumedLoc定义：表示源文件中代码位置。

常用操作：

getFilename()：获得本位置所在的文件名字。、

getLine()：获得对应源文件中的行号。

getIncludeLoc()：获得include语句的位置。

Clang::Token定义：提供了与lexed token有关的所有信息。

常用操作：

getLocation：

getName()：

Clang::Preprocessor定义：包含了语言选项，目标平台，源文件管理器，头文件搜索路径，builtin的宏，pragma处理函数，当前的lexer（lexer只能处理单个文件，不能处理Include文件栈），include栈信息，macro展开信息。

常用操作：

getCurrentLexer

Llvm::Module定义：是其他LLVM IR对象的最上层容器，模块包含了全局变量列表、函数列表、模块所依赖的库列表、符号表等。

常用的模块操作包括：

getSourceFileName()获得源文件名称。

getTargetTriple()获得目标平台信息。

GlobalValue \*getNamedValue(StringRef Name)：获得全局变量的值。

Function \*getFunction(StringRef Name) ：

Dump():打印出模块的内容。

LLVM：：Function定义：包含了参数列表、基本块列表、函数属性列表。

常用操作：

getArgumentList():

getBasicBlockList():

getEntryBlock():

viewCFG():打印出函数的CFG图，还包含了基本块的代码。需要ghostview的支持。

viewCFGOnly()：仅仅打印出函数的CFG，不包含代码。

Dump()：打印出函数内容。

Llvm::PassRegistry定义：管理遍的注册与初始化，并且帮助PassManager解析遍的依赖关系。

常用操作：

addRegistrationListener：注册回调函数。当一个遍注册时，就会调用这个函数。

registerPass:

getPassInfo:

Llvm::PassManager定义：

常用操作：

Add:最后调用的是PMTopLevelManager::schedulePass函数，会先调用遍的preparePassManager函数，然后检查遍是否是分析型的遍，如果是则将其删除（因为分析型的遍是当有遍用到时才调用），然后分析遍的依赖关系，然后将该遍加到对应PM（从PM栈中搜索出合适的PM，PM有模块级别的，函数级别，基本块级别，循环级别的等）中。

Llvm::PMDataManager定义：用来管理分析的数据analysis data。也是由遍组成的管理器。

Llvm::PMStack：PMDataManager结构的栈。

Llvm::ImmutablePass：主要是那些用来提供信息，而不是需要被执行的遍，如分析遍等。

PrintPreprocess过程：

FrontendAction::BeginSourceFile：初始化CompilerInstance的相关对象，如源代码管理器，预处理器等。

FrontendAction::Execute：

代理执行实际的PrintPreprocessedAction::ExecuteAction函数。

Preprocessor::EnterMainSourceFile：进入主要源文件。主文件只会进入一次。

Preprocessor::EnterSourceFile：将源文件加入到include栈顶，并且从它开始lexing token。处理include及macro语句。

回调的是PrintPPOutputPPCallbacks::FileChanged函数，当预处理器进入或退出一个#include文件时回调本函数，当然最开始处理的主文件也会触发。

然后会加入predefines的头文件。

使用clang::Preprocessor::Lex(Tok)来读取tokens，目的是消费掉来自predefines的tokens？？。然后打印出所有tokens。

LLVM code gen过程：

clang::ASTFrontendAction::ExecuteAction：为compilerInstance创建Code Complete Consumer与Sema对象，并调用clang::ParseAST函数。

使用clang::Parser::ParseTopLevelDecl函数来解析声明语句，在这里根据tok的类型分别进行处理，不能处理的类型再调用ParseExternalDeclaration函数（opencl的扩展就是在这个函数中处理的）。