添加debug信息

debug信息是一个很难解决的问题，原因主要在于优化代码上。首先，优化使得保持源位置更加困难。再来，优化会通过优化、与其他变量共享内存或难以跟踪的方式来移动变量。

在LLVM IR中，会保留指令上每个IR级指令的原始源位置。优化过程需要保留新创建的指令的源位置，但是合并后的指令只能保留一个位置，可能会导致在遍历优化程序时发生跳跃。在LLVM中，使用DWARF格式，它是一种紧凑的编码，代表类型、源位置和可变位置。

提前编译模式。突出向源语言添加调试信息的各个方面，并减少JIT调试的复杂性，先更改以支持将前端发出的IR编译成一个简单的独立程序，从而可以执行、调试和查看结果。

编译单元。DWARF中代码段的顶层容器是编译单元，包含单个翻译单元的类型和功能数据。

矮化发射装置。所拥有的DIBuilder类，可以为LLVM IR文件构建调试元数据，使用这个类来构建所有的IR级别描述。因此需要一个模块，在构建模块后不久构建它，并将其作为一个全局静态变量来便于使用。接下来创建一个小容器来缓存经常使用的数据。

功能。在有了编译单元和源代码位置后，将函数定义添加到调试信息中。

源位置。调试信息最重要的是准确的源代码位置，这可以将源代码映射回去。但Kaleidoscope在lexer或解析器中没有任何源位置信息，因此需要添加。先添加了一些关于如何跟踪“源文件”的行和列的功能，当将每一个标记都进行lex时，将当前的“词汇位置”设置为标记开头的分类行和列。用跟踪信息的新advance ( )覆盖了以前对getchar ( )的所有调用，然后在所有AST类中添加一个源位置。而为了确保每条指令都获得正确的源位置信息，需要将新的源位置传递给IRBuilder。

变量。有了函数后需要可以打印出我们范围内的变量。设置函数参数可以得到回溯。首先创建变量，给它范围( SP )、名称、源位置、类型，因为它是一个参数索引，再来创建一个LVM.dbg.declare调用，从而在IR级别指示在alloca中有一个变量，并在declare上为范围的开始设置一个源位置。需要注意各种调试器都有基于过去如何为它们生成代码和调试信息的假设，因此需要修改，以避免为函数序言生成行信息，以便调试器可以在设置断点时跳过这些指令。