编译原理 lab4 实验报告

李维璇 201250058

实验思路

- 根据文档所给框架代码复现
  - 函数要生成返回值类型、生成参数类型、生成函数，并添加基本块，且后续生成的指令将追加于此。
- 接下来是计算返回值表达式的值。
  - visit*函数将会返回LLVMValueRef，由此我们可以获得某表达式的值。
  - 根据优先级，递归实现，详情见下部分。

精巧的设计

- 本次要解析的表达式，总是可以被匹配为以下框出的5种的某一种：
  -
```
exp  : L_PAREN exp R_PAREN #L_EXP_R
     | lVal   #EXP_LVAL
     | number #EXP_NUM
     | IDENT L_PAREN funcRParams? R_PAREN     #EXP_FUNC_ARGUMENT
     | unaryOp exp # UNARY_EXP
     | exp (MUL | DIV | MOD) exp #EXP_MUL_EXP
     | exp (PLUS | MINUS) exp  #EXP_PLUS_EXP
     ;
```
  - 所以拿到整个表达式后，从上至下匹配这5种表达式。visit*函数将会返回LLVMValueRef，由此我们可以获得某表达式的值。
    - 对于`(exp)`的形式，递归调用对`exp`的访问。
    - 对于`number`，这是递归的终点，返回`LLVMConstInt(i32Type, int, 0)`。
    - 对于其他3种，对单个exp进行访问，得到的`LLVMValueRef`转化为long后，再根据符号进行相应的计算。并返回`LLVMConstInt(i32Type, int, 0)`。
  - 整个过程较为简洁，过程也比较清晰。

遇到的困难

- 加载maven依赖时一直在resolving maven dependencies。换源后解决。
- 对于LLVMValueRef，它会有好几种形态，如函数/变量/constInt，这里我需要知道他是什么类型并在他为constInt时获得这个值。查阅了不少资料后得知如此做：

```
if (LLVMGetTypeKind(LLVMTypeOf(llvmValueRef)) == LLVMIntegerTypeKind)
            return LLVMConstIntGetSExtValue(llvmValueRef);
```