编译原理 Lab4 实验报告

姓名：熊丘桓

学号：201250127

邮箱：eaglebear@smail.nju.edu.cn

1. 实现功能

本次实验完成了以下功能：

1. 翻译主函数的定义为中间代码
2. 翻译返回语句为中间代码
3. 计算整形字面常量表达式并翻译为中间代码

2. 实验设计

```java
@Override
public LLVMValueRef visitFuncDef(SysYParser.FuncDefContext ctx) {
    LLVMTypeRef functionType = LLVMFunctionType(i32Type, LLVMVoidType(), 0, 0);
    String functionName = ctx.IDENT().getText();
    LLVMValueRef function = LLVMAddFunction(module, functionName, functionType);
    LLVMBasicBlockRef mainEntry = LLVMAppendBasicBlock(function, "mainEntry");
    LLVMPositionBuilderAtEnd(builder, mainEntry);
    super.visitFuncDef(ctx);

    return function;
}
```

笔者按照实验文档设计了 `visitFuncDef` 方法，并根据语法规则设计了 `visitUnaryExp`，`visitParenExp`，`visitAddExp`，`visitMulExp`，`visitReturnStmt` 等方法。

3. 实验困难

笔者初始设计方案是将整形字面常量取值并在 java 中计算表达式求值，但该方案在 OJ 上仅得到了 1300 分（满分 3100）。

修改为调用 `LLVMBuildSDiv` 等 API 计算代替算术计算后得到了满分。

```java
@Override
public LLVMValueRef visitMulExp(SysYParser.MulExpContext ctx) {
    LLVMValueRef valueRef1 = visit(ctx.exp(0));
    LLVMValueRef valueRef2 = visit(ctx.exp(1));
    if (ctx.MUL() != null) {
        return LLVMBuildMul(builder, valueRef1, valueRef2, "tmp_");
    } else if (ctx.DIV() != null) {
        return LLVMBuildSDiv(builder, valueRef1, valueRef2, "tmp_");
    } else {
        return LLVMBuildSRem(builder, valueRef1, valueRef2, "tmp_");
    }
}
```