1.1中间代码生成

1.1.1. LLVM 中间代码

参考地址: https://llvm.org/docs/LangRef.html#instruction-reference

1.1.2. 华为方舟中间代码 MAPLE

见附件文档

1.1.3. 实验目的

（1）了解编译器中间代码表示形式和方法;

（2）掌握中间代码生成的相关技术和方法，设计并实现针对某种中间代码的编译器模块;

（3）掌握编译器从前端到后端各个模块的工作原理，中间代码生成模块与其他模块之间的交互过程。

1.1.4. 实验内容

以自行完成的语义分析阶段的抽象语法树为输入，或者以 BIT-MiniCC 的语义分析阶段的抽象语法树为输入，针对不同的语句类型，将其翻译为中间代码序列。例如下面的输入语句:

```
int main() {
    int a, b, c;
    a = 0;
    b = 1;
    c = 2;
    c = a + b + (c + 3);
    return 0;
}
```

其对应的四元式形式的输出为:

```
(=,0,,a)
(=,1,,b)
(=,2,,c)
(+,a,b,%1)
(+,c,3,%2)
(+,%1,%2,c)
```

对应的 MAPPLE IR 的输出为:

```
func &main() i32{
var %a i32
var %b i32
var %c i32
dassign %4(constval i32 0)
dassign %a(regread i32 %4)
dassign %5(constval i32 1)
dassign %b(regread i32 %5)
dassign %6(constval i32 2)
dassign %c(regread i32 %6)
dassign %7(
    add i32(dread i32 %a,dread i32 %b))
dassign %8(regread i32 %7)
dassign %9(constval i32 3)
dassign %10(
    add i32(dread i32 %c,regread i32 %9))
dassign %11(regread i32 %10)
dassign %12(
    add i32(regread i32 %8,regread i32 %11))
dassign %13(regread i32 %12)
dassign %c(regread i32 %13)
dassign %14(constval i32 0)
return (regread i32 %14)}
```

其中%开始的表示临时变量或者伪寄存器。

1.1.5. 实验过程与方法

BIT-MiniCC 内置 MAPPLE IR 生成器，示例代码 ExampleICGen 输出的是教材上的四元式。示例代码采用了 visitor 模式，必要时将一个节点的中间计算结果与一个伪寄存器相关联，代码中给出了双目运算“+”的实现供大家参考。同学们可以在自己已有的代码基础上实现，也可以自定义类实现。

示例代码中只是部分代码，大家注意多个函数时应将不同函数的中间代码放在一起。中间代码的操作码可以自行定义，没有标准答案，具体可以参考 MAPPLE IR 和 LLVM 中间代码的定义。

1.1.6. 实验提交内容

本实验要求提交中间代码生成实现源码（项目 src 下所有文件），以及对应的 config.xml 和 classpath 两个文件; C/C++需提供对应的可执行程序（不需要编译的中间文件），每个人提交一份实验报告。

实验报告放置在 doc 目录下，应包括如下内容:

- 实验目的和内容
- 实现的具体过程和步骤
- 运行效果截图
- 实验心得体会