Lab.7 中间代码生成实验报告

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。

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。

实验过程

实验中的中间代码采用四元式的形式，代码块使用Label来标记。通过对`visit`函数的重载，遍历AST树的所有节点，递归地生成四元式。

所有变量均未分配寄存器，一律视为全局变量。

部分语句的四元式定义：

BreakStatement

用`Break`标识，其四元式定义形如`(Break, , , )`

ContinueStatement

用Continue标识，其四元式定义形如(Continue，，，)

FunctionCall

参数用param标识，其四元式定义形如(param，/*参数名*/，，)；
在0条或多条表示参数的四元式后，以(call，，，/*函数名*/)结尾。

FunctionDefine

参数用param标识，其四元式定义形如(param，/*参数名*/，，)；
在0条或多条表示参数的四元式后，以(func，，，/*函数名*/)结尾。

GotoStatement

用goto标识， 其四元式定义形如(goto，，，/*跳转的标签*/)

IterationStatement & IterationDeclaredStatement

将循环代码块翻译为如下形式：

```
(Label，，，@loopStart)
... .../*循环计数器的初始值*/
(Label，，，@loopCheck)
... .../*循环条件语句*/
(JF，，，@loopEnd)
... .../*中间循环体语句*/
(Label，，，@loopNext)
... .../*末尾循环体语句*/
(Label，，，@loopEnd)
```

LabeledStatement

用Label标识， 其四元式定义形如(Label, , , /*跳转的标签*/)

ReturnStatement

用Ret标识， 其四元式定义形如(Ret, , , /*返回值，可能为空*/)

SelectionStatement

将分支代码块翻译为如下形式：

```
(Label, , , @1If)
... .../*分支条件*/
(JT, , , @1Then)
... .../*分支语句*/
(Label, , , @1Else)
(Label, , , @2If)
... .../*分支条件*/
(JT, , , @2Then)
... .../*分支语句*/
(Label, , , @1Endif)
```

运行截图

用BITMiniCC中test/ic_test/test.c进行测试。

```
(=, 1, , b)
(param, x, , )
(param, y, , )
(func, , , f1)
(+, x, y, %1)
(=, %1, , z)
(RET, , , z)
(func, , , f2)
(param, "in f2\n", , )
(call, , , Mars_PrintStr)
(RET, , , )
(func, , , main)
(=, 1, , a1)
(=, 2, , a2)
(!, , a1, a1)
(!, a1, , res)
(~, , a1, a1)
(~, a1, , res)
(+, a1, a2, res)
(%, a1, a2, res)
(<<, a1, a2, res)
(++, , a1, a1)
(++, , a1, res)
(++, , a1, a1)
(++, a1, , res)
(Label, , , @1If)
(&&, a1, a2, %2)
(JT, , , @1Then)
(param, a1, , )
(param, a2, , )
(call, , , f1)
(=, , , res)
(Label, , , @1Else)
(Label, , , @2If)
(!, , a1, a1)
(JT, , , @2Then)
(param, b, , )
(param, a2, , )
(call, , , f1)
(=, , , res)
(Label, , , @2Else)
(call, , , f2)
(Label, , , @1Endif)
(Label, , , @1loopStart)
(=, 0, , i)
(Label, , , @1loopCheck)
(<, i, a1, %3)
(JF, , , @1loopEnd)
(Break, , , )
(Continue, , , )
(+=, 1, , res)
(Label, , , @1loopNext)
(++, , i, i)
(Label, , , @1loopEnd)
(Label, , , @k)
(goto, , , @k)
(RET, , , 0)
```

实验心得

本次实验的原理并不难于理解，只要能够理解map在递归中的作用就能顺利地利用递归生成四元式。主要还是一些细节问题，包括自己定义的中间代码的合理性和生成器代码实现过程中的诸多奇奇怪怪的bug。