PA3 实验报告

姓名：杨雅儒 班级：无73 学号：2017011071

这部分主要是在PA2的基础上将其转换为TAC（三地址码）表示。

第一趟扫描计算每个类对应的类对象的大小，处理类成员方法的顺序，创建虚函数表，计算类成员变量偏移量，为函数创建 Functy 对象，并计算形参的偏移量。第二趟扫描完成所有语句和表达式的翻译工作。

## 代码阅读和个人理解

### decaf.tac

这部分是TAC有关的语句，基本不用修改。里面有 Functy.java, Label.java, Tac.java, Temp.java, VTable.java 五个文件。

#### Tac.java

用于表示TAC语句，里面含有各种用于构造Tac对象的静态方法，以及 Kind 的枚举类型用于表示是哪种类型的TAC语句。

#### Temp.java

用于处理临时变量，与实际机器中的寄存器相对应。函数的形式参数和局部变量均会被对应到一个Temp对象上。

在PA3第一趟扫描中会为函数的所有形式参数关联Temp对象，第二趟扫描中会为局部变量定义（VarDef）关联Temp对象。

可以通过Variable的getTemp()方法获得其关联的Temp对象；使用Temp的createTempI4()方法可以获得一个新的表示32位整数的临时变量。

#### Lable.java

Label 表示标号，一般调用其静态方法 createLabel(boolean target) 获得一个新的标号，在这个方法中会让 id 自增并将其名字取为 "\_L" + id。

函数的入口和跳转目标处都会存在标号。

#### VTable.java

表示虚表，定义非常短，只有String类型的name表示名字；VTable类型的parent表示父类的VTable；String类型的className表示类名；Label[]类型的entries表示虚表中各函数的入口标号。

#### Functy.java

表示函数体，定义也非常短。

Label类型的label表示函数入口标号；Tac类型的paramMemo，head，tail；Funtion类型的sym关联到Function符号上。

### decaf.translate

这部分用于将AST翻译为TAC，包含TransPass1类用于做第一趟遍历，TransPass2类用于做第二趟遍历，这两个类的实现同样使用Visitor 模式；Translator类用于辅助翻译以及输出。

输出时，首先打印VTABLE：内部依次为父类名字、类名、非静态的成员方法的名字；然后打印FUNCTION：内部依次为paramMemo、内部TAC指令（掺杂Mark）。注意Mark是标号，例如 “\_L15:”

#### TransPass1.java

主要用于处理类、方法、参变量等的定义，构建对应VTable，处理order、offset，以及建立对应Temp类对象等。

#### TransPass2.java

### 补充

#### Class.java

该类中增加了resolveFiledOrder()方法，用于将这个类（包括其继承的类）中所有的成员方法和成员变量按定义顺序排序，将顺序存放在符号类的order变量中。

另外还增加了VTable类型的vtable变量，用于存放类对应的VTable。

使用size存放了一个类实例对象的大小（字节数）。

## 工作记录

### 调试信息

在Tac类中的Kind类型增加DEBUG类型，增加genDebug(String)方法，并在toString中增加相应输出。在Translator.java中增加genDebug(String)方法以及genDebugInt(Temp)方法，前者用于在输出的tac中**直接**输出一个字符串，后者用于在输出的tac中使用指令使得在程序输出时输出一个数。

后来还加入了genDebugString(String)用于通过指令使得程序输出一个字符串。

### 特性1

Tree.java：在Scopy类中增加了Class类型的classSymbol变量，用于在Scopy语句中快速找到对应类。

TypeCheck.java：在visitScopy中增加对classSymbol变量的赋值。

TransPass2.java：新增visitScopy方法，其中利用store语句进行浅复制。

### 特性3

TransPass2.java：新增visitGuardedIf和visitIfSubStmt方法进行处理，其中在visitIfSubStmt中通过genBeqz进行跳转控制。

### 特性4

TransPass2.java：在visitIdent中增加visitVarDef类似代码，用于建立对应的Temp变量。

### 特性5

#### 1

TransPass2.java：在visitBinary中新增对于tag为Tree.DMOD的表达式的处理：使用tr.genCheckNewArraySize(expr.right.val)判定长度是否为正整数；另外通过判定类型是否为类来决定是否要进行浅复制。

Translater.java：增加成员变量public boolean inDMOD，用于判断是否在%%中创建数组，并且修改genCheckNewArraySize中给定的字符串进行判定是否非负。

RuntimeError.java：增加NEGATIVE\_ARR\_SIZE2表示DMOD的数组长度小于0错误

#### 2

TransPass2.java：重载visitDefaultArrayRef方法，利用布尔表达式（需判断超出length和小于零两种情况）判断将indexed还是deft赋值给val即可。

#### 3

TransPass2.java：重载visitForeach方法，在其中生成类似循环的TAC指令，并使用loopExits处理break语句；重载visitAttachedStmtBlock方法，对其中内容进行visit。

TypeCheck.java：更改checkTestExpr(foreach.expr2)的执行顺序到了调整var类型的后面

### 除以零

RuntiError.java：增加DIV\_ZERO常字符串。

### Debug

* 注意有的gen方法是有返回值的！不要忘记！
* 在实现Scopy时，对于Ident新建了一个Temp，而并没有与原来new这个Ident的联系起来。处理方式：Ident应当visitIdent，而不应自己新建Temp处理
* 在实现循环时，发现tr.genSub()是会新生成一个Temp，这样就不能用cnt=tr.genSub(cnt, one)实现自减（因为会变成形如\_T15 = \_T14 - \_T13，每次循环时\_T14都为原本的cnt，而\_T15只会变成cnt-1）。处理方式：使用tr.append(Tac.genSub()) 自行实现而绕过translater。同理其中的其它运算也应当注意此问题，**一般地，带Temp类型返回值的gen方法都会新建Temp，故都应当有所注意**