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**