编译原理 PA3 作业报告

黄家晖 2014011330

PA3 利用 Visitor 模式对 AST 进行访问，利用部分静态语义信息生成 TAC 代码。这种代码非常接近汇编代码，在此部分需要实现汇编语言的相关逻辑，增加新的语法特性。

1 新增加的数据结构和函数

1. 为了实现 switch 和 case 语句和?:操作符，需要为 TranslatorPass2 中遍历到相应节点时增加相应汇编语言生成。switch 使用的逻辑是先经过若干个 if 条件判断应该前往哪一个分支，再在下面依次定义这些分支；而三元运算符使用的逻辑是先做条件判断，根据条件的真假跳转到不同的标签继续执行。

2. 为了实现 <<操作符，为 Class 类增加 CopyFuncLabel 存储拷贝构造函数的 Label 信息，同时在 VTable 中存储对应的 Class 信息，将整个 VTable 的偏移量加 4，第三个位置放置一个伪非静态的拷贝构造函数。

3. 为了实现 repeat 语句和对应的 continue 语句，新加入栈，存储当前所在循环的开始点，每次遇到 continue 语句就跳转到这个开始点继续执行。

4. 为了实现运行时检查的除 0 非法错误，需要在每次进行除法或是取模操作的时候，先插入条件判断代码，如果除数为 0 则终止程序，输出错误信息。

2 遇到的问题和解决方法

本次 PA 我耗时比较长的是 <<操作符的实现，一开始误认为最小公共父类可以利用静态语义信息得到（即使用上次 PA 中的分析结果），但改好代码之后，发现测例中所有的 <<操作符都是运行时分析，而运行时分析的问题是不清楚新生成的对象大小是多少，应该如何拷贝信息。因此考虑定

义拷贝构造函数（技术上来讲，定义一个 getSize 函数也可以，但是不如这种方式简便），这个拷贝构造函数是在这个阶段加入的新函数，与所有的函数都不同，既不是静态函数，它需要存在 Vtable 中，也不是非静态函数，它的第一个形参不是 this 而是需要拷贝的源对象地址。这个拷贝构造函数的功能是建立自己对应类的 Vtable，并且拷贝源对象的部分函数到自己的地址空间来。这样，在遇到 <<操作符的时候，只需要在运行时分析出最小的公共 Vtable，即可查出所需拷贝构造函数进行隐式调用。