编译原理实验报告
PA3-JVM

姓名：谢兴宇
学号：2017011326

2019 年 12 月

1 工作简述

在阅读了实验指导书，对于 JVM 和 ASM 的概念及运行过程有了基本了解之后，我开始根据实验说明尝试完成本次 PA。

1.1 抽象

为抽象类和抽象方法添加`ACC_ABSTRACT`的标签。对于抽象方法，还需添加一个特殊判断，不再为其函数体生成代码。

1.2 First-class Functions

我采用了类似实验说明文档中的实现方式，将所有函数视为一个拥有`apply`方法的函数对象。对于成员方法或静态方法，其`apply`中的内容便是调用那个成员方法或静态方法；对于 lambda 表达式，其`apply`中的内容便是 lambda 表达式的语句块。

为了在不知道函数类的类名情况下调用函数对象，我为每一个出现的函数类型构造了一个也拥有`apply`方法的函数基类。这样，通过第二阶段计算出的函数类型，便可以调用一个函数对象了。

例如如下代码：

```
class Main{
    static void main() {
```

```
        fun(int x) {
            return "abc";
        };
    }
}
```

会被视作:

```
class _69=Dvoid {
    void apply() {
    }
}

class _int=Dstring {
    string apply(int a1) {
        return "";
    }
}

class Main$main extends _69=Dvoid {
    void apply() {
        Main.main();
    }
}

class Main$f extends _69=Dvoid {
    class Main _self;
    void apply() {
        _self.f();
    }
}

class Lambda$0 extends _int=Dstring {
    class Main _self;
    string apply(int x) {
```

```
        return "abc";
    }
}

class Main{
    void f() {
        new Lambda$0();
    }
    static void main() {
    }
}
```

2 问题回答

1. - 总体来看，PA3 基于函数闭包来实现。但由于 JVM 中无法实现指向指令内存的指针，所以依据实验说明文档，我采用了将所有函数视为一个拥有`apply`方法的对象的策略，将对函数的调用视为对函数对象的`apply`方法的调用。
   - 在 PA3 的实现中，出于简便，所有的函数对象中都留有`this`；在 PA3-JVM 的实现中，仅对于需要使用`this`的函数对象（成员函数和成员函数中的 lambda 表达式）我才保留了`this`。

2. JVM 字节码支持无条件跳转和有条件跳转，框架中将其翻译成了类似如下伪代码的 JVM 字节码。

```
        if (x >= y) goto true
        push 0
        goto exit
    true:
        push 1
    exit:
```