# Assignment 1

Count how many **GOTO** instructions have been executed in my program? In bytecode or intermediate code (such as Jimple code), GOTO instructions have been widely used to jump to different addresses. Figure 1 and Figure 2 show two examples. In this assignment, you are going to count ***how many*** GOTO instructions have been triggered for ***each class*** in ***dynamic*** executions.

```
// Bytecode stream: 03 3b 84 00 01 1a 05
68 3b a7 ff f9
// Disassembly:

iconst_0        // 03
istore_0        // 3b
iinc 0, 1       // 84 00 01
iload_0         // 1a
iconst_2        // 05
imul            // 68
istore_0        // 3b
goto -7         // a7 ff f9
```

*Figure 1 Goto In Bytecode*

```
public int stepPoly(int)
{
    unknown op0, l0, l1;

    l0 := @this;
    l1 := @parameter0;
    if l1 >= 0 goto label0;

    op0 = java.lang.System.out;
    op0.println("error");
    return -1;

 label0:
    if l1 > 5 goto label1;

    op0 = l1 * l1;
    return op0;
```

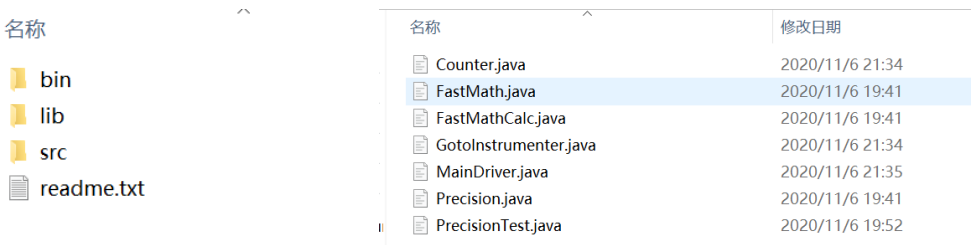
*Figure 2 Goto In Jimple Code*

## Objectives

a) Learn how to implement Java programs, including compiling, executing, and testing.
b) Learn how to instrument programs.
c) Learn how to use basic functionalities of Soot, which is a code static analysis and optimization framework for Java. More details could be found in https://github.com/soot-oss/soot

## Steps

1. Download the provided zip files ***assignment1.zip***, the structure of which is as follows:




Please go to this folder, and all the commands should be run under this folder. There are 7 source files under the `src` folder, and the meaning of them is as follows:

a) `FastMath.java; FastMathCalc.java; Precision.java.` These are target source files that we are going to instrument.

b) `PrecisionTest.java`. It is a test file which is designed to test the functionalities of the above three target files. You do not need to instrument or change this file. When finishing instrumentation, you will run this test file via JUnit (using our given command), and check how many GOTO instructions will be triggered in the above target files.

c) `MainDriver.java`. The main driver of our instrumentation. We have already implemented this class for you. You do not need to change it.

d) `Counter.java`. The helper class for instrumentation. You are required to implement necessary functionalities of this class (***those parts that have been marked with "TO DO"***).

e) `GotoInstrumenter.java`. The main class for instrumentation. You are required to implement necessary functionalities of this class (***those parts that have been marked with "TO DO"***).

2. Implement the required functionalities in `Count.java` and `GotoInstrumenter.java`. Please refer to the tutorial and PPT provided in the previous assignment.
3. After implementing the required two classes, test your implemented functionalities via run the following three commands:

   a) Compile:
   ```
   javac -cp lib/* -d bin src/*
   ```
   if successfully, you will see the compiled class file under folder "bin"

   b) Instrumentation:
   ```
   java -cp "bin;lib/*" MainDriver Precision FastMath FastMathCalc
   ```
   if successfully, you will see the instrumented class file under folder "sootOutput".

   c) Run Test:
   ```
   java -cp "sootOutput;bin/;lib/*" org.junit.runner.JUnitCore PrecisionTest
   ```
   We will check the results using this command. If you have successfully finished this assignment, we should see a file "result.txt" generated under the folder "report". We will then check the results in this file.

# Requirement

- Linux/Mac/Windows
- Java 1.8 (you need to install it on your computer)
- Soot 4.2.1 (we have provided it to you in "lib". You need to do nothing)
- Junit 4.1.1 (we have provided it to you in "lib". You need to do nothing)

# Assessment

Output the number of executions of the GOTO instruction for each class that has been executed to the file *"report/result.txt"*. The file should be generated after running command: *java -cp "sootOutput;bin/;lib/*" org.junit.runner.JUnitCore PrecisionTest*. In our assignment, please

output such numbers for class ***FastMath, FastMathCalc*** and ***Precision*** in the following format.

```
ClassName \t NumberOfGotoExecution
FastMath \t [number]
Precision \t [number]
…
```

We will compare the correctness of the outputted numbers to assess your assignment.

## Submission:

After implementing the required two classes, please zip the folder “assignment1” and submit it through “微助教”.