Andreas Landgrebe
September 9, 2015

Lab 1 Part 3
In order to do the changes of the integer values of i, j, k and l, this is the changes that needs to be made
sipush -2
(This will change to 65534 so this is a bug in the software from changing 32 bit and 16 bit operators.)
istore_1
sipush 5000
istore_2
ldc 65000
istore_3
iconst_3
istore 4

Lab 1 Part 4

It seemed that there are different instructions to be able to change different values between doubles and integers. If I assign a int constant to a double, I will be able to save the change in the Java byte code editor. Despite I was able to make the change in the Java Byte Code Editor software, when I was running the program, a run tine error occurred due to the fact that there are different protocol to parse a double variable to a integer and vise versa.

Lab 1 Part 5

There are a couple of reasons why the Java compiler uses several different types of instructions for the assignment in the previous two questions. One of the reasons for having different instructions is to be able to use the least amount of memory when assigning variables. When declaring a integer variable, it would take a less amount of bits to declare and assign when declaring and assigning a double or a string. Another reason for the different types of instructions is being able to declare the most common numbers with the least amount of memory being used to do so.

Lab 1 Part 6

The java bytecode for adding two int variables is **iadd**. Before you do any addition, you would need to load two variables and then add. In order to load any varaibles, the command to do so is **iload**
The java bytecode for multiplying two int variables is **imul**. Before you do any multiplication, you would need to load two variables and then multiply. In order to load any variables, the command to do so is **iload**
The java bytecode for substracting two int variables is **isub**. Before you do any

substraction, you would need to load two variables and then substract. In order to load any variables, the command to do so is **iload**
The java bytecode for dividing two int variables is **idiv**. Before you do any division, you would need to load two variables and then divide. In order to load any variables, the command to do so is **iload**

Lab 1 Part 7

Here is the Java Code to do constant folding

```
/*
Andreas Landgrebe
Computer Science 220 Lab 1 Part 7

Honor Code: I pledge that this program represents my own program code.
*/


public class Lab1Part7Easier {

        public static void main(String[] args) {

        int x = 3 + 5;
        }

}
```

In this example, the step of constant folding should have the compiler just to declare this Java source code to say that x is equal to 8. Here is what the Java byte code looks like after compiling and Java program

```
bipush 8
istore_2
return
```

As you can see, the step of constant folding was successful in the way of instead of having the compiler to add the two integers of 3 and 5 together in the byte code, the compiler successfully just said that the variable is assigned to 8.