CS212 - Solutions to Part 1

CS212 Staff

September 13, 2005

General Notes

In general, for each of the first five problems, not commenting every line resulted in a 2 point deduction. Using a return variable or memory storage for the first three problems (which was disallowed) was a five point deduction.

Problem 1 (8 pts)

The most common mistake for this problem was using ADDSP 1 to create a zero on the stack through memory allocation (which was disallowed).

```
PUSHIMM 1 // Push 1 onto the stack
DUP       // Duplicate 1 (now two copies of 1)
DUP       // Duplicate 1 again (now three copies of 1)
SUB       // Subtract top two copies.  Zero now on top
SUB       // Subtract 0 from third copy of 1
STOP      // Halt execution
```

Problem 2 (18 pts)

```
//!(T || ((!T && F) && (T || F)))


// evaluate (!True && False)
PUSHIMM 1 // push true
NOT       // !True
PUSHIMM 0 // Push false
AND       // Take and of !True and False

// evaluate (True || False)
PUSHIMM 1 // Push true
PUSHIMM 0 // Push false
OR        // ... and take the or

// evaluate lhs && rhs
AND
```

```
// evaluate !(True || RHS)
PUSHIMM 1 // push final true
OR        // take or with previous computation
NOT       // negate everything



STOP  // Halt execution
```

Problem 3 (18 pts)

```
PUSHIMM 1 // Push 1 onto the stack
PUSHIMM 8 // Push 8
PUSHIMM 3 // Push 3
PUSHIMM 4 // Push 4
ADD // 3+4 = 7
DIV // 8/7 = 1 (Integer division)
ADD // 1+1 = 2
PUSHIMM 2 // Push 2
PUSHIMM 1 // Push 1
PUSHIMM 2 // Push 2
DIV // 1/2 = 0
PUSHIMM 3 // Push 3
ADD // 0+3 = 3
TIMES // 2*3 = 6
DIV // 2/6 = 0
STOP
```

Problem 4 (18 pts)

```
//allocate 4 variables and rv
ADDSP 5

//assign values to variables
PUSHIMM 0  // value of a
STOREABS 4 // assign to a's location
PUSHIMM 1  // value of b
STOREABS 5 // assign to b's location
PUSHIMM 2  // new value of b
STOREABS 5 // assign to b's location
PUSHABS 4  // push value of a onto stack
PUSHABS 5  // push value of b onto stack
ADD        // add top two elements (a + b)
STOREABS 6 // assign to c's location
PUSHIMM 1  // value of d
STOREABS 7 // assign to d's location
```

```
//check if a+b <= c
PUSHABS 4  // push a
PUSHABS 5  // push b
ADD        // add them
PUSHABS 6  // push c
//GREATER NOT implies "less than or equal to"
GREATER
NOT

// check if (d || ~d && false) is true or false
PUSHABS 7 // push d
PUSHABS 7 // push second copy of d for negation
NOT       // negate d
PUSHIMM 0 // push false
AND       // !d and false
OR        // d or (!d and false)

// compare the two expressions and store it in rv
AND       // and of two bug expressions
STOREABS 3     // store into space for return value

//deallocate the 4 variables
ADDSP -4  // deallocate all other variables on the stack,
          // leaving only the return value
```

Problem 5 (18 pts)

```
//public boolean function()
//{
// a = 0;
// b = 1;
// b = 2;
// c = a + b;
// d = true;
// return (( a + b) <= c ) && (d || (~d && false)));
//}

// declare a, b, c, d (and return)
ADDSP 5

// a = 0
PUSHIMM 0
STOREOFF 2

// b = 1
PUSHIMM 1
STOREOFF 3
```

```
// b = 2
PUSHIMM 2
STOREOFF 3

// c = a + b
PUSHOFF 2
PUSHOFF 3
ADD
STOREOFF 4

// d = true (1)
PUSHIMM 1
STOREOFF 5


// (a + b) <= c
PUSHOFF 2
PUSHOFF 3
ADD
PUSHOFF 4
SUB
ISPOS
NOT

// (d || (~d && false))
PUSHOFF 5
NOT
PUSHIMM 0
AND
PUSHOFF 5
OR

// lhs && rhs
AND

// returnValue = current value on the stack
STOREOFF 1

// remove a, b, c, d from the stack but leave the return value!
ADDSP -4
```

Problem 6 (20 pts)

Many people had one or two points deducted for a lack of comments on Problem 6. In general, there were four points assigned for comments: 1 point for including the comment block with your name and netID, 1 point for a comment outside of the function or class describing the purpose of

the class, and 2 points for useful comments within the body of the function (where "useful" was defined very broadly)

```
// To compile, rename file SAM_DMULT.java

import edu.cornell.cs.sam.core.SystemException;
import edu.cornell.cs.sam.core.instructions.SamInstruction;

// This instruction takes in three integers(a,b,c) and
// adds/multiplies them in the form a*(b+c)
public class SAM_DMULT extends SamInstruction {
    public void exec() throws SystemException {
        // pop first int off stack
        int c = mem.popINT();
        // pop second int off stack
        int b = mem.popINT();
        // pop third int off stack
        int a = mem.popINT();

        // Compute the solution and push it into stack
        mem.pushINT(a*(b+c));

        // increment the PC
        cpu.inc(PC);
    }
}
```