CS212 Part 1

Fall 2005

Please be sure to read the material posted on the website prior to attempting this assignment. Also, make sure your submission follows all guidelines, as described in section 2.

1 Tasks

The following tasks will introduce you to SaM and the SaM Simulator. For tasks 1-3 you are not allowed to use `PUSHOFF`, `PUSHABS`, `STOREOFF`, and `STOREABS`. Also, do not use a return variable for those tasks.

1. Write a Samcode program `task1.sam` that evaluates the expression

   $$1 - 0$$

   without using the `PUSHIMM` instruction more than once.

2. Write a Samcode program `task2.sam` that evaluates the following expression

   $$\neg\left(T \vee \left(\left(\left(\neg T\right) \wedge F\right) \wedge \left(T \vee F\right)\right)\right)$$

   Remember that T is true and F is false. You do not need to consider any order of operations. If you are unfamiliar with logic symbols, please see a website such as http://en.wikipedia.org/wiki/Table_of_logic_symbols.

3. Write a Samcode program `task3.sam` that evaluates the following numerical expression

   $$\frac{1 + \frac{8}{3+4}}{2 \times \left(\frac{1}{2} + 3\right)}$$

   You should only use integer operations. Standard order of operations applies for this task.

4. Write Samcode that corresponds to the following Java function using only the absolute addressing approach

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

Note that the `&` and `|` operators in this code do *not* short-circuit. You are not required to write the code that will call your function (this is something that will be introduced in later parts of the project). You are required to use the following template:

```
ADDSP 1
JSR function
STOREOFF 0
STOP
function:
LINK

// ######### YOUR CODE HERE #############
// Reserve space for variables
// Perform computations
// Leave return value on top of stack
// ######### END YOUR CODE ##############

SWAP
UNLINK
SWAP
RST
```

Include *only* the code you write in a file called `task4.sam`. Make sure to allocate space for all four variables and the return variable. The variable slots you create will have addresses ≥ 3. After your code finishes running, the return value should be at address 3 (you will see that the code we provide uses addresses 0, 1, and 2).

5. Write Samcode that corresponds to the Java code in Task 4 using only the relative addressing approach. For this part of the assignment you should assume that the FBR register to which addresses are relative is set to SP $-$ 1 when your code begins execution (immediately below the first free stack location). Therefore, the first free location is at offset 1, the second at offset 2, etc.... Offset 0 is reserved and you should not write to it. You should again use the template provided for task 4 and include only the code you write in a file called `task5.sam`.

6. Write a Java class that would enhance SaM's instruction set with the `DMULT` instruction. The `DMULT` instruction implements distributive property for multiplying three numbers:

$$\begin{aligned} \text{DMULT}(a, b, c) &= a(b + c) \\ &= ab + ac \end{aligned}$$

   The `DMULT` instruction should take as input the integers a, b, and c and produce an integer result. Thus, running the following Samcode should leave the integer 9 on the stack:

```
PUSHIMM 3
PUSHIMM 2
PUSHIMM 1
DMULT
STOP
```

   You do not need to account for the possibility that the three input values are not integers. The instructions included with SaM are located in the *edu.cornell.cs.sam.core.instructions* package. You may look at the code for instructions like `ADD` for hints on how to accomplish this.
   You should submit your implementation of `DMULT` in a file called `task6.java` (we will change the name of your file when grading to allow it to successfully compile). Your file should include all necessary import statements. In order to test your file, you may use the *Load Instruction* feature of the Simulator. After launching the simulator, go to *File→Load Instruction* and select the correct class file. Watch the status bar for a confirmation that the `DMULT` instruction has been loaded. You can then open a sample program and test your solution.

7. This semester we will be using a special Assignment Management System for the submission of your project. The major benefit to you is that the submission application will check your submitted code to make sure it will be correctly loaded by our grading applications and that you have included all files in your submission. Furthermore, we anticipate including test case facilities in future versions. In order to make sure there are no issues with the submission part of the AMS, you will need to create a test submission as part of this assignment. The Bali section of the website contains a sample compiler (that essentially does nothing) called `BaliCompiler.java`. Use the submission system provided under the *Source* link in CMS to submit this file. The first time you create a submission, the submission application will prompt you to find a particular file. Please see the CS212 website (under the AMS section) for more information on creating the submission.

2 Submitting Your Work

Be sure to test your code before submitting it.

2.1 Partners

Although we allow groups of up to three people for future assignments, for part 1 you must work by yourself.

2.2 Submission Guidelines

For tasks 1-6, write each problem's solution in a separate text file, as described in each task decription. For example, if we gave a task 0 its solution might be as follows:

```
// +-----------------------------------------+
// | Part 1, Task x                          |
// | David I. Schwartz                       |
// | dis9                                    |
// | 123456                                  |
// +-----------------------------------------+

// Write a program that returns true
PUSHIMM 1 // push true
STOP      // stop execution
```

For this assignment, you should comment every single line of Samcode so that we know that you know what is happening. For general submission instructions, follow the CMS submission specifications on the CS211 Fall 2005 website. Each file you create will be submitted individually on CMS. If we cannot read your work in a standard text editor, the work is unacceptable and will receive no credit. For task 7, submit the package created by the submission application.

2.3 Grading

We will grade your assignment based on style (how well you followed our instructions, how neatly you arranged your solutions, how neatly you wrote your code, and how clear you made your comments) and correctness (how well the programs work).