CPSC 359 – Winter 2017
Assignment 5
Microprogramming
25 points (Weight: 8%)
Due April 12 @11:59 PM

Objective: To work with Mic-1 MMV, extending the IJVM ISA

1. **A JAS program (10 points total)**
   a. **Adding a new ISA instruction (1 point):**
   Add a new JAS instruction to the IJVM instruction set INEG by modifying its microprogram. INEG negates (in 2's complement) the value at the top of the stack. INEG is part of the JVM instruction set.

   b. **Write a JAS program that contains two methods:**
   - divCount(int1): returns the number of devisors for int1. **(4 points)**
   - isMul(int1, int2):returns 1 if int2 is a multiple of int1; 0 otherwise **(4 points)**

   Since IJVM ISA does not have instructions for multiplication and division, your JAS methods must use addition and subtraction instead. For instance, to multiply int1 and int2, assuming int1 > int2, add int1 to itself int2 times. The general algorithm is as follows, where abs(int1) is the absolute value of int1:

Here is a pseudo code for getting the number of divisors (pseudo code from: http://stackoverflow.com/questions/110344/algorithm-to-calculate-the-number-of-divisors-of-a-given-number):

```
int divisors(int1)
    {
      int limit = abs(int1);
      int numbOfDivisors = 0;

      if (abs(int1) == 1) return 1;
      for (int j = 1; j < limit; ++j)
      {
         if (abs(int1) % j == 0)
         {
            limit = abs(int1) / j;
            if (limit !=j)
            {
               numOfDivisors++;
            }
            numOfDivisors++;
         }
      }

     return numOfDivisors;
    }
```

Since JVM has no division instructions, you may want to implement division if needed through subtraction. The following is a simple algorithm (assuming int2 is non-zero).

```
idiv(int1, int2, return_type) {
        abs_int1 = abs(int1)
        abs_int2 = abs(int2)
        q = 0 \\ quotient
        r = abs_int1 \\ remainder
        while (r >= abs_int2) {
                r = r – abs_int2
                q++
        }
        If (exactly one of int1 or int2 is negative)
                q = - q
        if (return_type = 0) return q
        else return r
}
```

Demonstrate the use of these two methods (divCount & isMul) by using the IN and the OUT JAS instructions **(1 point)**. Assemble and test your program using the Mic-1 MMV. You should be able to take input from user and then display the result in the output console in Mic-1 MMV.

In the file *add.jas* contained in the examples folder of Mic-1 MMV, there are I/O methods to read and print numbers. You may use these methods for testing; however, they do not work with negative numbers. Hence, when testing your code with negative numbers, you have to do it the hard way, monitoring the stack and registers, or write your own. If you write your own getnum() and print() methods that work with negative numbers (print negative numbers with the minus sign), **2 points**.

2. Extending the IJVM ISA (10 points total)

Add two new instructions to the IJVM ISA:

- IIsMul: pops the top 2 values, a and b, on the stack and pushes 1 if b is multiple of a, 0 otherwise **(4.5 points)**.
- IIDIV: pops the top 2 values on the stack and pushes back the remainder and quotient. **(4.5 points)**

You need to ensure that these two JAS instructions work with negative numbers (**3 points**)

Modify the Mic-1 microprogram (in MAL) so that these two instructions can be interpreted by the IJVM hardware. Use the same algorithms for multiplication and division explained above.

Demonstrate the use of these two methods (IIsMul, IIDIV) by writing a JAS function that uses the IN and OUT instructions in Mic-1 MMV**.** Assemble and test your program using the Mic-1 MMV. You should be able to take input from user and then display the result in the output terminal. **(1 point)**

You need not worry about overflows, since the IJVM ALU does not have an overflow flag.

Up to **1.5 points** can be deducted from each implementation (IIsMul and IIDIV) for inefficiency.

Thoroughly document your programs (both JAS and MAL); programs that are not properly documented can lose up to **3 points**.

Programs that do not compile cannot receive more than **5 points**. Programs that compile, but do not run at all can receive a maximum of **6 points**.

Teams: You may work in teams of up to **two students** in order to complete the assignment, but you are not required to do so. **Your partner must be in a tutorial that is taught be the same TA.** Peer evaluation in teams may be conducted.

Demonstration & Submission: Submit a .tar.gz file of your entire project directory (including source code and header files) via the appropriate drop box on Desire2Learn. You will also need to be available to demonstrate your assignment during the tutorial.

Late Submission Policy: Late submissions will be penalized as follows:
-12.5% for each late day or portion of a day for the first two days
-25% for each additional day or portion of a day after the first two days
Hence, no submissions will be accepted after 5 days (including weekend days) of the announced deadline.

Academic Misconduct: Any similarities between assignments will be further investigated for academic misconduct. While you are encouraged to discuss the assignment with your colleagues, your final submission must be your own original work. Any re-used code of excess of 10 lines (10 JAS or MAL instructions) must be cited and have its source acknowledged. Failure to credit the source will also result in a misconduct investigation.

D2L Marks: Any marks posted on D2L are tentative and are subject to change (UP or DOWN).