CPSC 359 – Spring 2019
Assignment 2
Due: May 28thth @ 11:59PM

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:**
   - imul(int1, int2): multiplies the parameters and returns the result (int1*int2) on the stack **(4 points)**
   - idiv(int1, int2, return_type): integer divides int1 by int2 and returns either the the quotient or the remainder depending on return_type. If return_type = 0, the method returns the quotient; otherwise, it returns the reminder. **(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(x) is the absolute value of x:

```
imul(int1, int2) {
        m = 0
        c = min(abs(int1), abs(int2))
        o = max(abs(int1), abs(int2))
        for (i = 0; i < c; i++)
                m = m + o
        if (exactly one of int1 or int2 is negative)
                m = - m
        return m
}
```

   You need not worry about overflows.

   Integer division can also be accomplished through subtraction. For instance, int1/int2 is calculated as follows (assuming int2 != 0):

```
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 by calling them from the main method **(1 point)**. Assemble and test your program using the Mic-1 MMV.

2. Extending the IJVM ISA (10 points total)

Add two new instructions to the IJVM ISA:

- IMUL: pops the top two values on the stack and pushes their multiplication **(4.5 points)**.
- IDIV: pops the top two values on the stack and pushes their integer division remainder and then the quotient **(4.5 points)**

For this part, you can assume that both values that are being multiplied or divided are non-negative.
Modify the Mic-1 microprogram (in MAL) so that these two instructions can be interpreted by the IJVM hardware. Follow similar steps to the ones in part 1(a) above. Use the same algorithms for multiplication and division explained above, ignoring the handling of negative numbers.
Write a tester JAS program that uses these two new instructions. **(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 (IDIV and IMUL) for inefficiency.

Thoroughly document your programs (both JAS and MAL); programs that are not properly documented can lose up to **3 points**.
Assemble and test your programs using the Mic-1 MMV.

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**.

Bonus questions (to be answered individually on D2L surveys) **(2 bonus points)**

3. On a scale of 1 to 5 (1 = Strongly Disagree, 2 = Disagree, 3 = Neutral, 4 = Agree, 5 = Strongly Agree), answer each of the following questions directly on D2L:

a. This assignment and the microarchitecture module provided me with a better understanding of data paths.
b. This assignment and the microarchitecture module provided me with a better understanding of ALU operation.
c. This assignment and the microarchitecture module provided me with a better understanding of memory latency.
d. This assignment and the microarchitecture module provided me with a better understanding of microprograms in general.
e. This assignment and the microarchitecture module provided me with a better understanding of microarchitectures in general.
f. This assignment and the microarchitecture module provided me with a better understanding of JVM internal working.

4. In the space below indicate if there are other things you learned from this assignment and the microarchitecture module.

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. 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 dropbox on Desire2Learn. Each team need **only make one submission.**

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 20 lines (20 assembly language instructions) must be cited and have its source acknowledged. Failure to credit the source will also result in a misconduct investigation.

D2L Marks: Marks posted on D2L are subject to change (up or down).