CSO Assignment #1

Instructor: Dr. Avinash Sharma *Due date: 01/02/2020*

Welcome to Assignment 1 of the Computer Systems Organisation Course. The aim of this assignment is to familiarize you with writing x86 code. On completion of this assignment you should be able to successfully write arithmetic, conditional, and looping components in x86-64. You will also be able to convert a given program in a higher level programming language into its equivalent assembly code.

Instructions: Read all the instructions below carefully before you start working on the assignment.

- There are 3 problems for this assignment.
- Writing complete code with successful execution guarantees full marks. Failure on test cases will result in penalisation. Therefore ensure all edge cases are handled.
- The assignments will be manually evaluated for plagiarism. Any and all forms of plagiarism will result in 0 credit for this assignment.
- Comment every line of your code and justify why you write that statement. Well-commented code will be awarded extra credit.

Submission Format: Strictly adhere to the following submission format. Failure to do so may result in an erroneous evaluation of your assignment.

- The following directory structure is expected,

```
./<roll_number>
    |__ Q1
    |   |__ Q1.pdf
    |__ Q2
    |   |__ Q2.s
    |   |__ Q2.pdf
    |__ Q3
        |__ Q3.s
        |__ Q3.pdf
```

- Zip the ./ <roll_number> folder and name the zipped folder as <roll_no_assign1.zip>

Problem 1: Integer Representation (10 points)

Consider the following C code that runs on a 32-bit machine that uses two's complement arithmetic:

Code Snippet:

```
1  int x = your_rollnumber%100;
2  int a = -1*(x);
3  unsigned int b = (unsigned int)a;
4  unsigned int c = UINT_MAX-x;
5  int d = (int)c;
6  int p = 65490 + x;
7  short int e = (short int)p;
8  unsigned short f = (unsigned short)a;
9  printf("%d %u %u %d %hi %hu\n",a,b,c,d,e,f);
```

What will be the output of the above code snippet? Write clear steps to show how each value is calculated.

Problem 2: x86 assembly code (40 points)

Given a number x, your task is to find first natural number i whose factorial is divisible by x.
Instructions

- You have to write the code in x86-64 assembly language
- The number x will be stored in register *%rax* using the *mov* instruction.
- The final result should be stored in register *%rdi*.
- Assume x is chosen such that overflow does not occur.

(a) Report 5 results for different values of n. Your code should be saved as Q2.s
(b) What value of x, results in the factorial calculation to overflow? Report your answers in Q2.pdf

Problem 3: Pseudo-code to x86 (50 points)

Write the equivalent x86 assembly code for the given pseudo-code. Make use of any number of registers of your choice. Final answer should be stored in register *%rdi*. Note: *g.t* here refers to *greater than operator(>)*. a, b, n are chosen such that they do not produce an overflow.

Algorithm 1 Modular exponentiation

Input: a, b, n
Output: $a^b \bmod n$

```
1: Let acc = 1.
2: a = a mod n.
3: while b g.t 0 do
4:    if b is odd then
5:       acc = (acc * a) mod n
6:    end if
7:    b = b/2.
8:    a = (a * a) mod n.
9: end while
10: return acc
```

(a) Report 5 results for different values of a, b, n in Q3.pdf