Computer Architecture
CS 325 - ON40
Department of Physics and Computer Science
Medgar Evers College
Exam 4

Direction: Modify the file "exam04.cpp" and submit your work in the Exams directory of your github repository and/or as an attachment on Google classroom under the Exam04 assessment.

Problem	Maximum Points	Points Earned
1	5	
2	5	
3	5	
4	5	
Total	20	

A SM computer has 20-bit words and instructions in the format

`[opcode|operandX|operandY]`

where opcode is 4 bits and each operand is a memory reference that is 8 bits. The instruction commands list for the computer are

Opcode	Description
0	Halts the program
1	Adds M(X) and M(Y) and puts the result in X
2	Subtracts M(Y) from M(X) and puts the result in X
3	Multiples M(X) and M(Y) and puts the least significant bits of the result in X and most significant bits in AC
4	Divides M(X) by M(Y) and stores the quotient in X and the remainder in AC
5	Transfers M(X) to AC
6	Transfers contents from AC to X
7	Transfers M(X) to Y
8	Makes next instruction X
9	If contents of AC $\geq$ M(X), makes next instruction Y
A	If contents of AC $\leq$ M(X), makes next instruction Y
B	If contents of AC ≥ 0, makes next instruction X; otherwise, makes next instruction Y
C	Increments M(X) by 1
D	Decrements M(X) by 1

where instructions that work with a single operand sets the second operand to 00, and the function M(X) implies the contents of memory address X.

1. Write a program that stores triple the sum of n consecutive integers in the memory address 32 where the value of n is stored in the memory address 64.
   Hint:
$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

2. Write a C++ void function named `ArithmeticShift()` whose header is

```
void ArithmeticShift(bool word[],int bits)
```

Given that *word* has a size of 32, if *bits* is positive, the function performs an arithmetic left shift of the bits (elements) of *word* by *bits* number of bits; otherwise, it does nothing.

3. For each problem, perform the operation provided on the twos complement operand(s) provided in the order they are provided and state if there is overflow. You must show work to receive full credit.

   a. **Negative**; `(R1) = 1011001001011110`;

   b. **Addition**; `(R1) = 101111010001`; `(R2) = 01010011`

   c. **Subtraction**; `(R1) = 1110101011001011`; `(R2) = 001010110011`

   d. **Multiplication**; `(R1) = 1101`; `(R2) = 0110`

4. Write a C++ void function named `UnsignedMultiplication()` whose header is

```
void UnsignedMultiplication(bool op1[],bool op2[],bool r[])
```

Given that *op1*, *op2* both have a size of 4 and represent unsigned binary numbers, and *r* has a size of 8, the function assigns the product of *op1* and *op2* to *r*. For instance, if

$$\begin{aligned} op1 &= \{\texttt{true,true,false,false}\} \\ op2 &= \{\texttt{false,false,true,true}\} \end{aligned}$$

then after the call of `UnsignedMultiplication()`,

$$r = \{\texttt{false,false,true,false,false,true,false,false}\}$$

To help define `UnsignedMultiplication()`, you can use the helper function `Adder()` whose header is

```
void Adder(bool a[],bool b[],bool r[],const int n)
```

given that *n* represents the size of *a*, *b* and *r*, the function assigns the sum of *a* and *b* to *r*.