SP22 CSCI 113 Assignment 5 25 pts due: 03/17 (Th)

Simulation programming for Booth's Multiplier

Build a simulator for the Booth's 2's complement number multiplier.
In this practice, we use 16-bit operands for the multiplication, i.e., 16-bit multiplicand, 16-bit multiplier, and 32-bit product. This simulator should have a subcomponent, 16bit_ALU, which performs addition and subtraction operations.

Suggested steps of building the simulator:
1. Implement a simulator for 1-bit ALU, which manipulates addition and subtraction operations;
2. Implement a simulator for 16-bit ALU, which uses 1-bit ALU as a subcomponent;
2. Implement a simnulator for the Booth's multiplier, which uses 16bit ALU as a subcomponent;
   This module should have the following subcomponents too:
      registers MD, AC, MQ (16 bits each), and 4-bit cycle_counter (initialized to binary "1111")
   This module is responsible for displaying the contents of the registers at each cycle and returns product.
3. You should write a driver (e.g., main function), which accesses two input operands (16 bit binary each) from keyboard or data file, calls the Booth's multiplier by passing the operands as arguments, and displays the product returnd from the Booth's multiplier.

In the Booth's multiplier module, you should implement the followings:

```
Initialization step:
    Set cycle_counter to binary "1111";
    Clear AC (16-bit) with binary "0000000000000000";
    Put multiplier into register MQ (16-bit);
    Put multiplicand into register MD (16-bit);

Repeat while cycle_counter ≥ 0
    1.  MQ0  MQ-1   Action on MQ0  //initial MQ-1 = 0
         0    0     None    (AC ← AC + 0)    //by calling 16-bit ALU for add
         0    1     Add MD  (AC ← AC + MD)   //by calling 16-bit ALU for add
         1    0     Sub MD  (AC ← AC – MD)   //by calling 16-bit ALU for sub
         1    1     None    (AC ← AC + 0)    //by calling 16-bit ALU for add
    2.  arithmetic-shift AC/MQ/MQ-1 >> 1;
    3.  cycle-counter -- ;
```

Input:
   16-bit multiplicand and 16-bit multiplier (2's complement binary number) from keyboard or data file;
   Please use (have to use) the following three sets of testing data, i.e., run your program three times.

```
1. MD= 0000 0010 0001 1001,  MQ= 1111 1111 1110 0110
2. MD= 1111 1111 1110 0110,  MQ= 0000 0010 0001 1001
3. MD= 1111 1111 1010 1000,  MQ= 1111 1111 1011 0100
```

Output:
   Show contents of registers step by step, e.g.,

```
cycle-counter     MD                  AC                  MQ              MQ-1
  1111      0000001000011001    0000000000000000    1111111111100110    0    // initial
  1111      ……                  ……                  ……                  ..   // step1 of 1st itr
  1111      ……                  ……                  ……                  ..   // step2 of 1st itr
  1110      ……                  ……                  ……                  ..   // step1 of 2nd itr
  1110      ……                  ……                  ……                  ..   // step2 of 2nd itr
  ……                                                                           …..
```

Submission (to Canvas):
- Source code; please include a good global documentation and each function head documentation in your source code (before compilation).
- Output (screenshots); run your program three times with the given data (1, 2, 3).