**Learning Objectives:**

This assignment is to illustrate the steps required for integer multiplication using the algorithm discussed in class.

**Overview:**

For this assignment you will write a program that will ask the user for two unsigned numbers between the values of 0 – 255. You will then multiply those two numbers showing each step as we discussed in class.

Description:

1. For this assignment you will print the steps for an 8-bit by 8-bit integer multiplication.
2. Your program should ask the user to enter two unsigned numbers the first being the multiplicand and the second being the multiplier. This should be a function.
3. Your program should write a function to verify both values are within the appropriate range. If the values were not within the appropriate limit, the function should print the appropriate error message and ask the user to enter the value again. Example, if the first value is incorrect tell the user the value is outside of the acceptable range and ask the user to re-enter the multiplicand.
4. Your program should initialize the simulated registers. This could be a function.
5. Print the input values in decimal and also in binary. This could be a function.
6. Next, your program should illustrate the steps required for multiplication using the same type of step diagrams discussed in class, as well as, shown in the class notes. This could be a function.
7. Finally, your program should print a check section that shows a summary of the multiplication in binary as well as in decimal. This could be a function.
8. Because you will need to print the binary representation of the numbers you will need to write a function to prints binary in C. You must use some form of bit manipulation in this function. Most of you wrote this function in PA1, therefore, you can reuse that function. However, several of you did not use bit manipulation in your print binary function. You lost points on that assignment and will lose points on this assignment if you fail to use bit manipulators in this assignment.
9. You should have at a minimal, 3 files; driver.c, functions.h, and functions.c. The driver should have a minimal amount of code. The functions.h file should have prototypes for all functions, etc. The functions.c should contain the implementation of all functions.
10. You will need to include the makefile from Lab 8. (your make file must have a make run and make clean)

**Formatting, Compiling, and Hand-in Requirements:**

Tar your file **PA2.tar.gz** and submit the file through **hand-in to the folder PA2. I will use a script to grade your assignment and if you do not follow the specific Hand-in instructions you will break my script, which will cost you points.**

**When tarring your file do not tar a PA2 folder. Please cd into the folder and tar the files in the folder. I will deduct points if you tar a folder.**

At a minimal, you should submit the following files:

* driver.c
* functions.c
* functions.h
* makefile
* a “readme” file that has the following information:
  + problems you encountered with this assignment
  + how you solved the problems
  + your thoughts about the assignment

Other considerations:

* With the exception of the makefile, you should have a header in each of your files that contains the following information. If you neglect to place a header in a file, you will receive a 5-point deduction.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

\*Your name \*

\*CPSC2310 Spring 20 \*

\*Lab Section: <Your section> \*

\*UserName: \*

\*Instructor: Dr. Yvon Feaster \*

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

* Your program should compile with no warnings and no errors on the School of Computing servers. There will be a substantial point deduction if your program has compile errors. If you made a substantial effort to complete the program, but you have compile errors, you will receive no more than 30 on the assignment. If the program compiles but has warnings, there will be a minimal of 10-point reduction. (The more warnings you have the higher the deductions.)
* You should use meaningful variable names throughout the program.
* Your code should be well documented. (comments – see example below)
* There should be no line of code longer than 80 characters.
* You should use proper and consistent indention.

Failure to do any of the above items will result in a deduction of points for each offense.

Here are some guide lines for documenting the code in this assignment.

Before each function**, in the .h files**, you should have a detailed description of what the overall function does. To borrow from another student’s code, here is an example of overall function description.

/\* Parameters: img - image\_t pointer array holding the image data for  
 \*                   each of the input files  
 \* Return:     output - image\_t struct containing output image data  
 \* This function averages every pixels rbg values from each of the   
 \* input images and puts those averages into a single output image  
 \*/

You are required to have this type of comment block before each function.

Also, if you include comments in the body of the function (and you should) they should be placed above the line of code not beside the code.

Example:

Bad

if(something) //This is a comment

{

do something;

}

Good

//This is a comment

if(something)

{

do something;

}

You should format your output as shown below, except yours should be an 8 bit output. 10% of the grade will be awarded for the following format.

Sample run is given below:

M = multiplicand = 11 (User specified)

Q = multiplier = 13 (User specified)

C = carry = 0

ACC = accumulator = 0

STEP 0 Initialize the data M = 1011 (11) C = 0 ACC = 0 Q = 1101 (13)

C ACC Q

STEP 1 0 0000 1101

+ 1011 lsb = 1 add M to the ACC

0 1011 1101

Shift >> 1

0 0101 1110

C ACC Q

STEP 2 0 0101 1110

+ 0000 lsb = 0 no add to ACC

0 0101 1110

Shift >> 1

0 0010 1111

C ACC Q

STEP 3 0 0010 1101

+ 1011 lsb = 1 add M to the ACC

0 1101 1111

Shift >> 1

0 0110 1111

C ACC Q

STEP 4 0 0110 1111

+ 1011 lsb = 1 add M to the ACC

1 0001 1111

Shift >> 1

0 1000 1111

Extra Credit:

10 points

Write your program such that it can accept any size of data up to 32 bits. Ex. Your program should be able to accept unsigned int, unsigned short, uint8\_t, uint16\_t, uint32\_t. When possible, use the limits.h file to determine the upper and lower limits when checking the input.

All other requirements are the same as the regular option.

If you are doing the extra credit option, you will need to submit to the to the PA2EC bucket in handin. Please DO NOT submit to both the regular and extra credit handin bucket.