**Table of Contents:**

# Comp 3350: Computer Organization & Assembly Language

# HW # 8. Theme: Integer Arithmetic

*All main questions carry equal weight.*

*Points will be awarded to only those answers which have work clearly shown*

* Question 1 is answered on page 1
* Question 2 is answered on page 2
* Question 3 is answered on page 3
* Question 4 is answered on page 4
* Question 5 is answered on page 5

1. In the following code sequence, show the value of AL after each shift or rotate instruction has executed. This question is to be done by hand, not by running a program.

mov cl, 2

mov al, 25h ; al = 00100101b

rol al, cl ; al = 10010100b = 94h

mov al, 4Ah ; al = 01001010b

mov cl, 1

ror al, cl ; al = 00100101b = 25h

stc ; CF = 1

mov al, 3Eh ; al = 00111110b

mov cl, 2

rcl al, cl ; al = 11111010b = FAh

stc ; CF = 1

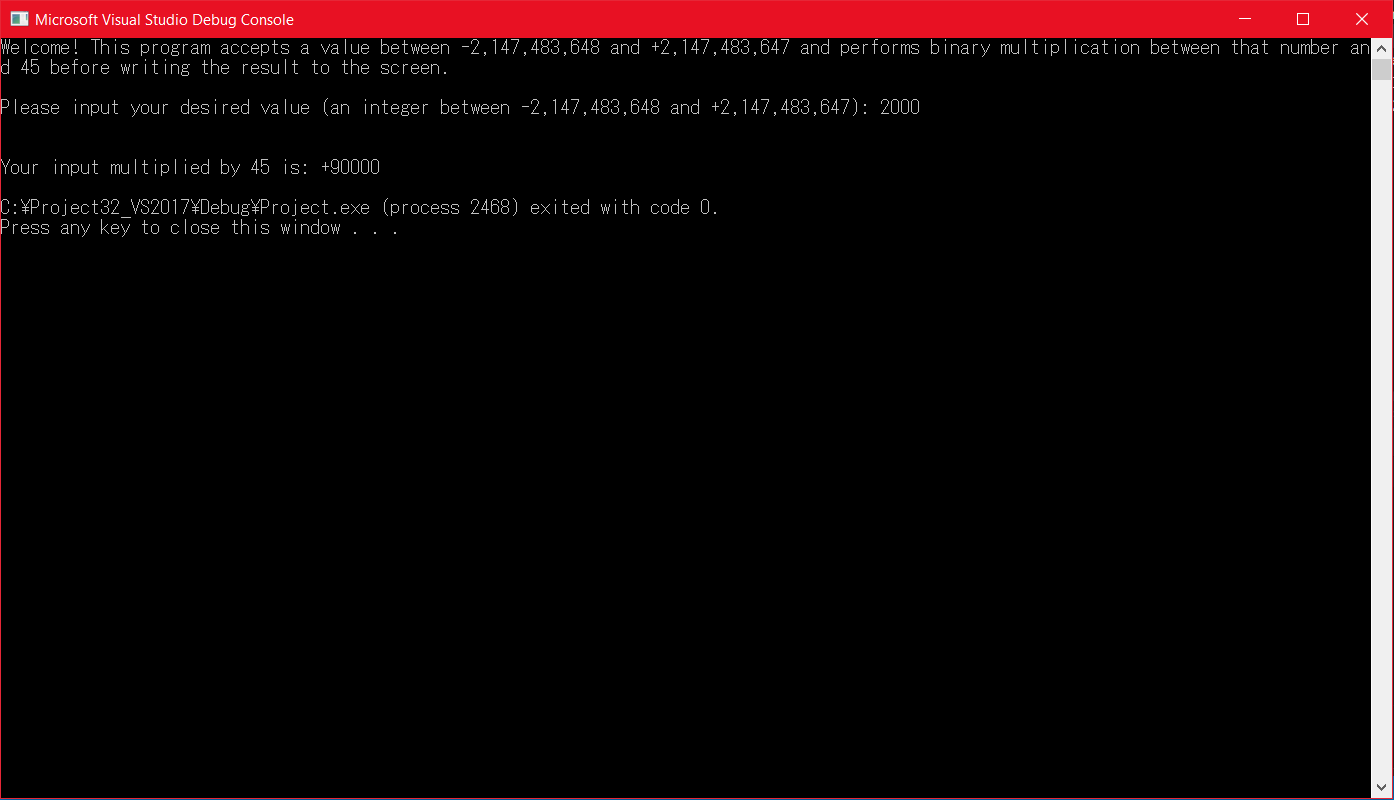
mov al, EAh ; al = 11101010b

mov cl, 1

rcr al, cl ; al = 11110101b = F5h

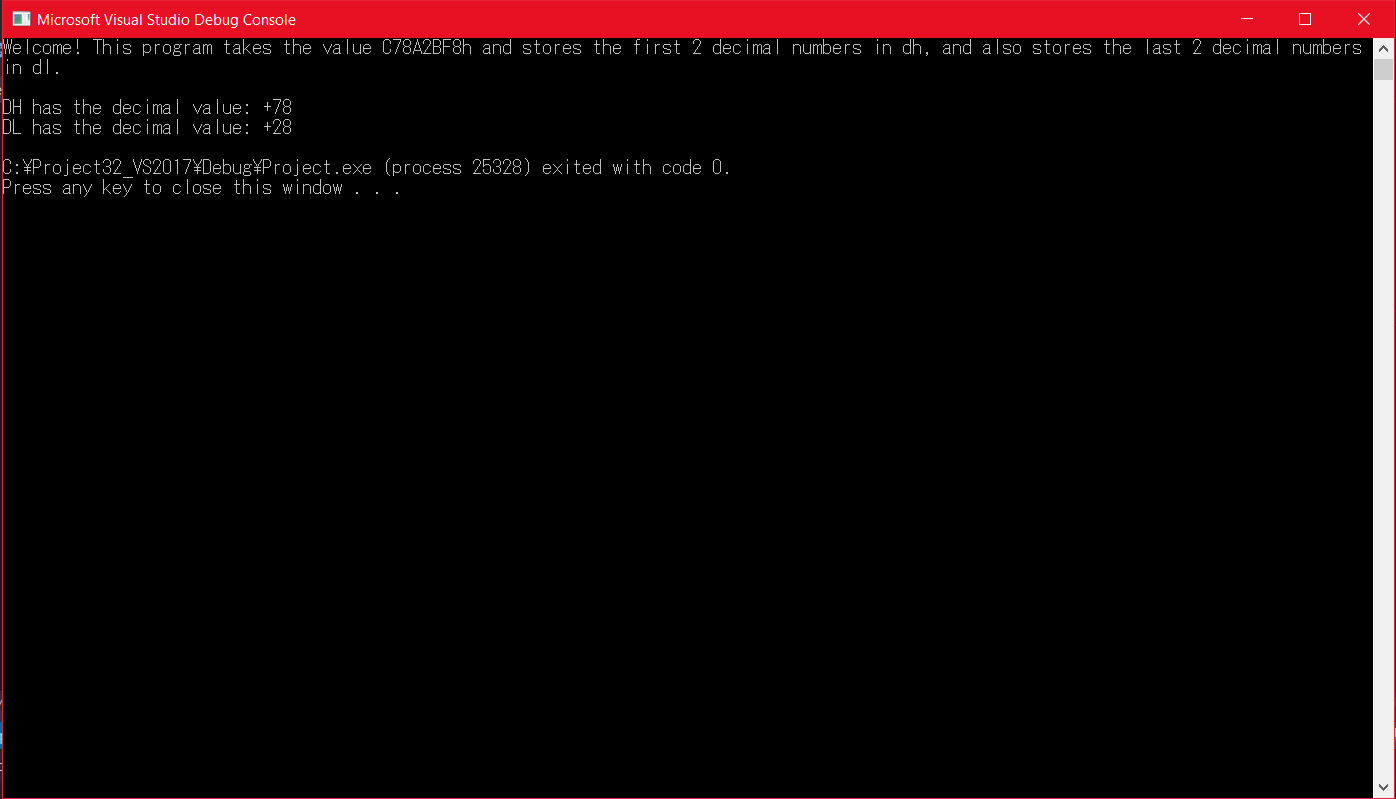
1. (a) Write a program which calculates EAX\**4510* using binary multiplication.

**Screenshot:**



(b) Consider the following value: C78A2BF8. Let this value be stored in register EAX. Write a program that will extract the decimal digits from this value using shifts and logical instructions. Place the first two **decimal numeric** digits in DH and the other two into DL. Submit a print out of the run of the program and the list file.

**Screenshot:**



1. (a) What will be the contents of AX and DX after the following operation? You must work this problem by hand, not by a program run. What may happen if you do not set dx to 0 in the beginning?

mov dx, 0

mov ax, 3333h

mov cx, 4444h

mul cx ; dx = 0DA7h, ax = 258Ch

**Explanation:** If DX was not set to 0 at the beginning, and if some previous operation modified the existing value within the EDX register, then the 0 within the DX value may have a different value that didn’t get cleared.

1. When does an IDIV instruction cause an overflow? Provide an example.

**Answer:** An overflow is caused by a quotient produced by the instruction that is too large (or too small if working with signed integers) for the destination operand. When an overflow happens, the program abruptly stops and an error code is provided.

**Example:** mov ax, 700

mov bl, 2

idiv bl ; 350 cannot be stored in AL – overflow error

1. What will be the values of DX:AX after the following instructions execute? What might be the use of such a sequence of instructions in a 16-bit computer?

mov ax, 0h

mov dx, 0h

sub ax, 1h ; ax = FFFFh, CF = 1

sbb dx, 0 ; dx = FFFFh

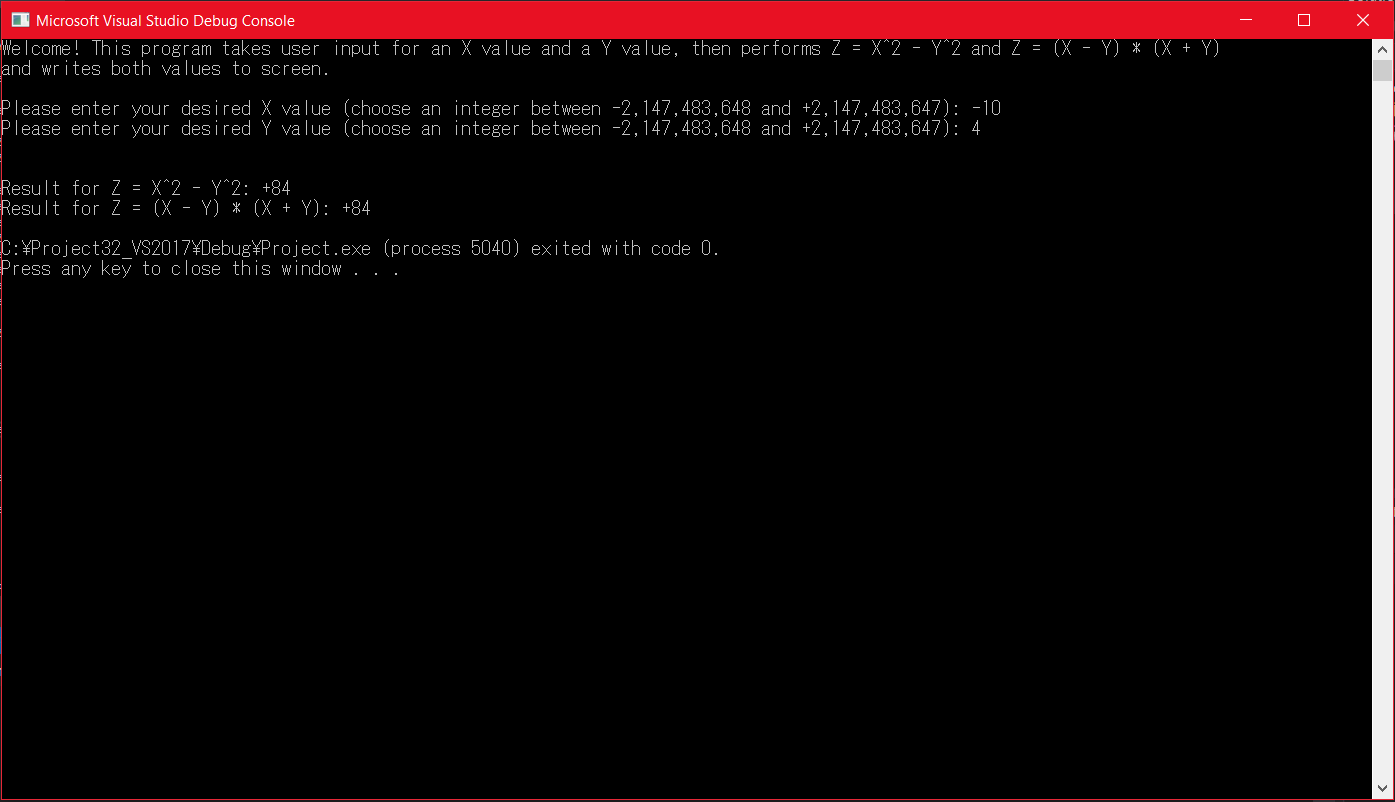
**Explanation:** These instructions, along with the use of DX and AX, can emulate 32-bit integers and perform 32-bit operations, as 16-bit computers cannot naturally use any 32-bit operations/registers/integers.

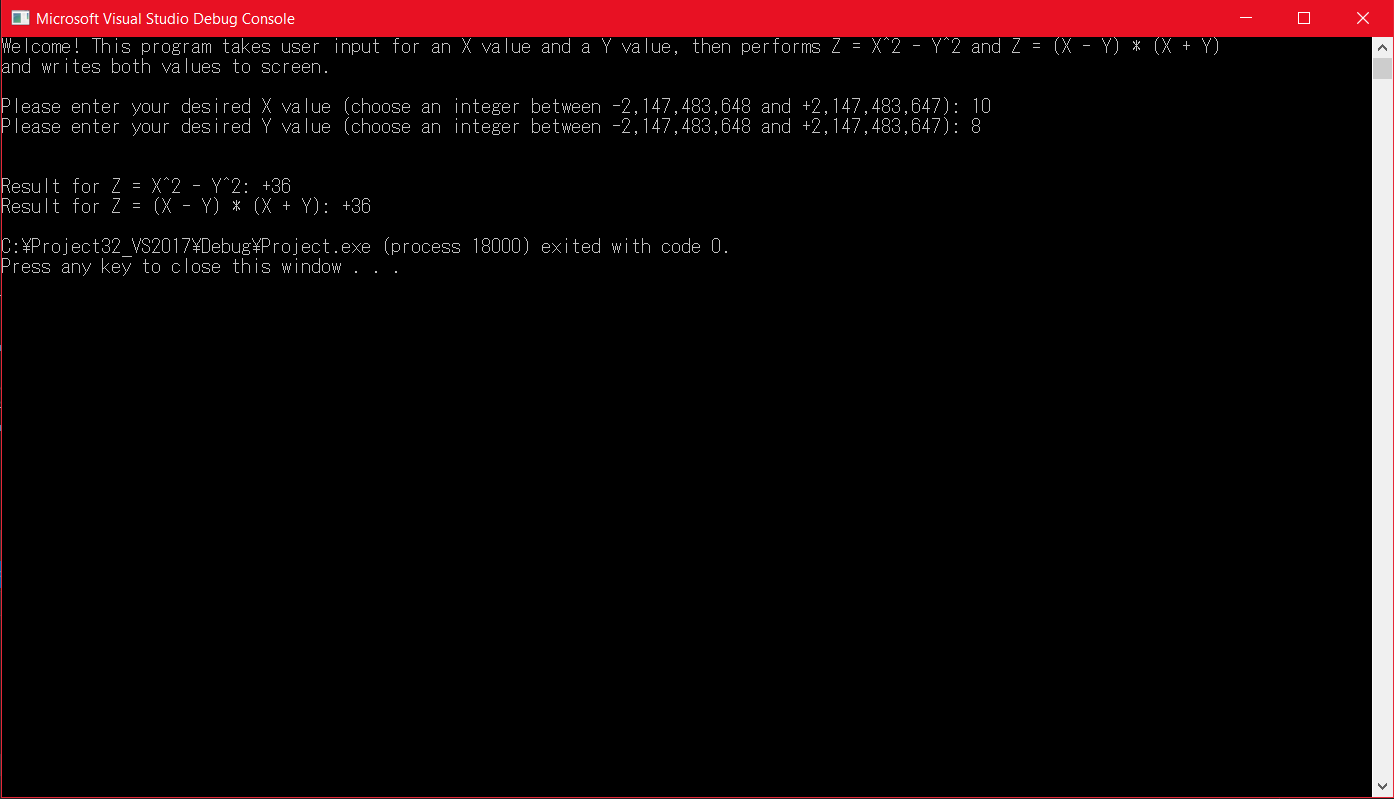
1. Implement the following two expressions in assembly language, using 32-bit signed operands. Demonstrate the equivalence of the two using some test values for X and Y. Show the runs of the programs using both positive and negative test values. Which of the two implementations is preferable?

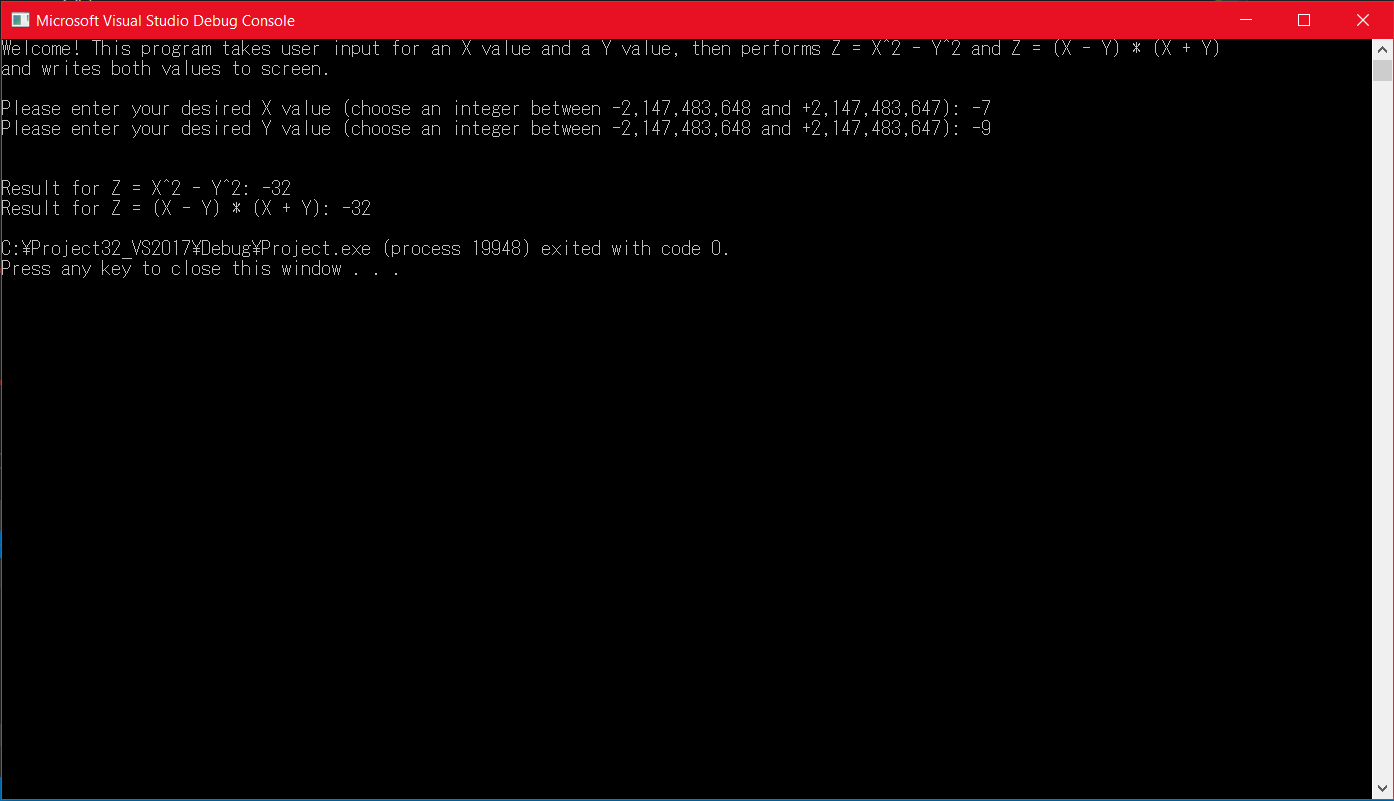
*Z = X2 - Y2*

*Z = (X-Y)\*(X+Y)*

**Screenshots:**







**Explanation:** The first implementation is the most preferable, as it took 6 lines while the second took 10 lines. Both implementations will yield the same result, but the first implementation is the easiest and fastest to execute in Assembly programming language.

1. Write a program that performs C = A + B using extended addition. See textbook pg. 270-271.

Use the following:

Apple QWORD 123456789ABCDE12h

Berry QWORD 0ABCDEF123456789Ah

Cherry QWORD ?

You may only use16-bit registers to perform the addition, e.g. AX, BX etc.

Submit the list file and a display of the contents of all the arrays after the run.

**Screenshot:**

