**HW # 8. Theme: Integer Arithmetic**

*All main questions carry equal weight.*

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

* 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, 1   
            mov al, 12h              ; al = 00010010h = 12h  
            rol al, cl ; al = 00010100h = 14h  
 shl al, cl ;al = 00101000h = 28h  
            mov al, 34h ;al =00110100h = 34h  
            mov cl, 2   
            ror al, Cl ; al = 00001101 = 0Dh  
 shr al, cl ; al = 00000011 = 03h  
            stc CF= 1  
            mov al, 56h al = 56h = 01010110h  
            mov cl, 1  
            rcl al, cl al = 10101101 = ADh  
            stc  
            mov al, 78h al = 01111000 = 78h  
            mov cl, 1   
            rcr al, cl al = 10111100 = BCh

* 1. (a) Write a program which calculates EBX\*11*10* using binary multiplication.

(b) Let the Consider the following value: A1BC23D4h.  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 screenshot of **the console output** of the program and the **asm/lst file**.  Note that you are writing a program for this specific example where the letter and digit positions are known to you (you are NOT writing a generic program to separate the letters and digits) .

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

            mov dx, 0  
      mov ax, FFFAh  
      mov cx, 5h  
      mul cx

ax \* cx = 5 \* FFFAh

dx:ax = 0004FFFAh

If dx wasn’t cleared, then it would effect the dx part of the product

(b) When does an IDIV instruction cause an overflow? Provide an example. It produces an overflow when the quotient is too large for the operand.

Mov ax, 900

Mov bl 2

Idiv bl;

(c) 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, 3h ; FFFDh CF = 1   
      sbb dx, 0

dx = FFFDh

DX:AX will act as a 32 bit integer

* 1. Enter, assemble and execute the following program which implements a case table.  Write a paragraph explaining how the code works.  Expand the program to work with inputs 'A', 'B', 'C', 'D' and similarly 'E'.  Test execute it.  What is the disadvantage of manually putting a value for EntrySize and NumberofEntries instead of the way it is done in the program?

A case table is made containing the addresses of each procedure, the entry size is equal to the current address minus the address of the case table therefore equaling the size of a byte. The number of entries is equal to the current address – the case table address divided by the entry size. ECx is equal to NumberOfEntries. Then the program compares to the input to the value of ebx, if its not A, then it iterates the pointer in the case table until the value at the case table equals the input. This solution is more flexible, robust and general instead of just having a finite number that must be changed everytime a new proess is added.