# HW8

- Due Apr 9 by 5pm
- Points 100
- Submitting a file upload
- File Types docx, pdf, asm, and lst
- Available Apr 3 at 8am - Apr 14 at 5pm

This assignment was locked Apr 14 at 5pm.

## 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. Show your work.

```
mov cl, 1
mov al, 1Ah
rol al, cl
shl al, cl
mov al, 0D4h
mov cl, 1
ror al, cl
shr al, cl
stc
mov al, 83h
mov cl, 1
rcl al, cl
stc
mov al, 34h
mov cl, 1
rcr al, cl
```

2. (a) Write a program which calculates EBX*10 $_{10}$ using binary multiplication.

   (b) Consider the following value: B56CA2E9h. 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) .

3. (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, 80B0h
mov cx, 3h
mul cx
```

   (b) When does an IDIV instruction cause an overflow? Provide an example.

   (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, 5h
sbb dx, 0
```

4. 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?

```
.data
        CaseTable BYTE 'A'       ;lookup value
                  DWORD Process_A  ;address of procedure
        EntrySize = ($ - CaseTable)
```

```
                BYTE 'B'
                DWORD Process_B
                BYTE 'C'
                DWORD Process_C
                BYTE 'D'
                DWORD Process_D
        NumberOfEntries = ($ - CaseTable) / EntrySize
        msgA BYTE “Process_A”, 0
        msgB BYTE “Process_B”, 0
        msgC BYTE “Process_C”, 0
        msgD BYTE “Process_D”, 0
        prompt BYTE “Press A, B, C or D:”, 0
.code
Main Proc
        Mov edx, offset prompt
        Call writestring
        Call readchar
        mov ebx,OFFSET CaseTable
        mov ecx, NumberofEntries
L1:     cmp al,[ebx]
        jne L2
        call NEAR PTR [Ebx + 1]
        call WriteString
        call Crlf
        jmp L3
L2:     add ebx, EntrySize
        loop L1
L3:     exit
main EndP

Process_A Proc
         Mov edx, offset msgA
         Ret
Process_A EndP

Process_B Proc
         Mov edx, offset msgB
         Ret
Process_B EndP

Process_C Proc
        Mov edx, offset msgC
        Ret
Process_C EndP

Process_D Proc
        Mov edx, offset msgD
        Ret
Process_D EndP
END main
```