HW8: Chap. 7, Integer Arithmetic

1. In the following code sequence, show the value of AL after each shift or rotate instruction has executed. ***Credit will NOT be given for answers that do not show work.***

```
mov al, 59h         ;       01011001 → 10110010 → 01100101 → 11001010 → 10010101 → 00101011
rol al, 5           ;       A: al = 00101011 = 2Bh

mov al, 0D5h        ;       11010101 → 11101010 → 01110101 → 10111010
ror al, 3           ;       B: al = 10111010 = BAh

stc
mov al, 3Bh         ;       00111011 → 01110111 → 11101110 →  11011100
rcl al, 3           ;       C: al = 11011100 = DCh

stc
mov al, 0A9h        ;       10101001 → 11010100 → 11101010 → 01110101 →  00111010
rcr al, 4           ;       D: al = 00111010 = 3Ah
```

2. Write instructions that calculate EAX*91 using binary multiplication.
   91 = 2^6 + 2^4 + 2^3 + 2^1 + 2^0

   ;assuming the result is to be stored in eax, overwriting the previous value...

```
MOV ebx, eax        ;copy n to ebx
SHL ebx, 6          ;ebx = n * 64
ADD eax, ebx        ;add ebx
SHR ebx, 2          ;ebx = n * 16
ADD eax, ebx        ;add ebx
SHR ebx, 1          ;ebx = n * 8
ADD eax, ebx        ;add ebx
SHR ebx, 2          ;ebx = n * 2
ADD eax, ebx        ;add ebx
```

3. The time stamp of a file uses bits 0-2 for hours, bits 3-7 for days, bits 8-11 for months, and bits 12-15 for years. Write instructions that extract the days and copy the value to a byte variable named “days”.
   ;using 'timestamp' to denote wherever timestamp is stored

```
.data
        days BYTE ?
.code
        MOVZX eax, timestamp        ;ax = hhhd dddd mmmm yyyy
        SHL ax, 3                   ;ax = dddd dmmm myyy y000
        SHR ax, 11                  ;ax = 0000 0000 000d dddd
        MOV days, al
```

4. What will be the contents of AX and DX after the following operation?

```
       mov dx, 0            ;dx = 0000 0000 0000 0000 = 0h
       mov ax, 1337h        ;ax = 0001 0011 0011 0111 = 1337h
       mov cx, 1000h        ;cx = 0001 0000 0000 0000 = 1000h
       mul cx               ;dx:ax = 0000 0001 0011 0011 0111 0000 0000 0000 = 01337000h

dx = 0000 0001 0011 0011 = 0133h
ax = 0111 0000 0000 0000 = 7000h
```

5. Implement the following C++ expression in assembly language, using 32-bit unsigned operands:
Alpha = (Alpha / Beta) + ((Gamma – Delta) * Iota)

```
INCLUDE Irvine32.inc
.data
       Alpha DWORD 18
       Beta DWORD 3
       Gamma DWORD 4
       Delta DWORD 2
       Iota DWORD 3
.code
main PROC
       MOV edx, 0
       MOV eax, DWORD PTR [Gamma]
       SUB eax, DWORD PTR [Delta]
       MUL DWORD PTR [Iota]

       MOV ecx, eax

       MOV EDX, 0
       MOV eax, DWORD PTR [Alpha]
       DIV DWORD PTR [Beta]

       ADD eax, ecx

       MOV Alpha, eax

exit
main ENDP
END main
```

6. 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, Dh    ;ax = 0000 0000 0000 1100 = 0Dh
       mov dx, Eh    ;dx = 0000 0000 0000 1101 = 0Eh
       sub ax, Fh    ;    0000 0000 0000 1100
                         -0000 0000 0000 1111
                     ax = 1111 1111 1111 1101 = 0FFFEh
       sbb dx, 0     ;dx – 1 = 0000 0000 0000 1100 = 0Dh
```

This sequence of instructions could be used to do higher than 16-bit subtraction on a 16-bit computer, as it will subtract from a second 16-bit value IFF the carry flag is set by the previous 16-bit subtraction.