HW #4 | COMP 3350 | 02/03/2021

1. A. Program fragment that will clear the sign flag

```
mov bx,0 ; bx=0
sub bx,2 ; bx=-2 , SF=1
add bx,3 ; bx=1 , SF=0
```

B. Program fragment that will clear the carry flag

```
mov al, 0FFh
add al, 1  ; CF=1, AL=00
sub al, 1  ; CF=0, AL=0FFh
```

carry out ①

ff : al
+ 1
00 : al

C. Program fragment that will clear the over flow flag.

```
mov al, 7Fh
add al, 1  ; OF=1 , AL=80h
sub al, 1  ; OF=1 , AL=7Fh
```

D. What will be the value of the parity flag after the following lines execute, show your work

```
mov al, 17 ; AL=17
add al, 10 ; AL=27, PF=1
```

17
+10
27_{10}

Converting $27_{10} \rightarrow$?$_2$ (binary)

2 | 27		LSB ↓
2 | 13	r.1	
2 | 6	r.1	
2 | 3	r.0	
2 | 1	r.1	
0	r.1	↑ MSB

$27_{10} \rightarrow 11011_2$

27_{10} → 4 1's, meaning an even parity or even number of 1's

Since therefore, parity flag (PF) will be set (1)

2.

```
.data
My Array BYTE 4Eh, 64h, 9Ah, 7Fh, 3Ch
Total WORD ?
.code
```

2.
```
.data
MyArray BYTE 4Eh, 64h, 9Ah, 7Fh, 3Ch
Total WORD ?
.code
mov edi, OFFSET MyArray ; address of MyArray
mov ecx, LENGTHOF MyArray ; loop counter ecx = 5
mov ax, 0 ; zero the accumulator
L1:
add ax, [edi]
add edi, TYPE MyArray
loop L1
mov Total, ax
```

3.
```
.data
myBytes BYTE 21h, 67h, 8Ch, 0BAh
myWords WORD 4518h, 49A3h, 04C32h, 257Bh, 0DF30h
myDoubles DWORD 0E1D24h, 0C273h, 67F2h, 0B34Ch, 96
myPointer DWORD myDoubles

.code
mov esi, OFFSET myBytes
mov ax, WORD PTR [esi+1] ; AX = 8C67h
mov eax, DWORD PTR myWords ; EAX = 49A34518h
mov esi, myPointer
mov ax, WORD PTR [esi+2] ; AX = 0000h
mov ax, WORD PTR [esi+3] ; AX = 7300h
mov ax, WORD PTR [esi-2] ; AX = DF30h
```

Memory Map on page (3)

3. Memory Map (address-data table)
Little-Endian

MyBytes	21
MyBytes+1	67
MyBytes+2	8C
MyBytes+3	BA
MyWords	A8
MyWords+1	45
MyWords+2	A3
MyWords+3	49
MyWords+4	32
MyWords+5	AC
MyWords+6	7B
MyWords+7	25
MyWords+8	30
MyWords+9	DF
MyDoubles	D4
MyDoubles+1	E1
MyDoubles+2	00
MyDoubles+3	00
MyDoubles+4	73
MyDoubles+5	E7
MyDoubles+6	00
MyDoubles+7	00
MyDoubles+8	F7
MyDoubles+9	67
MyDoubles+10	00
MyDoubles+11	00
MyDoubles+12	46
MyDoubles+13	B3
MyDoubles+14	00
MyDoubles+15	00
MyDoubles+16	79
MyDoubles+17	96
MyDoubles+18	00
MyDoubles+19	00

4. .data
myArray DWORD 2 DUP(9), 2130, 65, 0CDE, 6DUP(2)

.code
mov ax, TYPE myarray ; AX = 4 ; the size in bytes of a single element in myArray
mov ax, sizeof myarray ; AX = 44 ; (TYPE myArray * LENGTHOF myArray)
mov ax, lengthof myarray ; AX = 11 ; number of elements in myArray

side work:
myArray DWORD 2DUP(9), 2130, 65, 0CDE, 6DUP(2)

myArray DWORD 9, 9, 2130, 65, 0CDE, 2, 2, 2, 2, 2, 2

---

5. (Sign Extension)

.code
1: mov bx, 0B62Eh
2: movzx eax, bx ; EAX = 0000B62Eh
3: movzx edx, bh ; EDX = 000000B6h
4: movzx cx, bl ; CX = 002E

5: mov bx, 0C142h
6: movsx eax, bx ; EAX = FFFFC142h
7: movsx edx, bl ; EDX = 00000042h
8: movsx cx, bh ; CX = FFC1h

1: mov bx, 0B62Eh ; bx = B62Eh = 1011 0110 0010 1110₂
Convert B62Eh to binary:

Hex:
A = 10₁₀ = 1 0 1 0₂
B = 11₁₀ = 1 0 1 1₂
C = 12₁₀ = 1 1 0 0₂
D = 13₁₀ = 1 1 0 1₂
E = 14₁₀ = 1 1 1 0₂
F = 15₁₀ = 1 1 1 1₂

bx = B62Eh =

1 0 1 1 (8 4 2 1) 0 1 1 0 (8 4 2 1) 0 0 1 0 (8 4 2 1) 1 1 1 0 (8 4 2 1)₂ = bx

2: movzx eax, bx ; EAX = 0000B62Eh
(eax: destination operand; bx: source operand)
zero extension; 32-bit register; 16-bit register

* Each hex digit is 4 binary bits

3: movzx eax, bh ; EDX = 0x000000B6

bx = B6 2Eh
bh = B6 (8-bit register)
bl = 2E (8-bit register)

4: movzx cx, bl ; CX = 0x002E

bl = 2E

5. (cont)

5: mov bx, 0C142h ; bx = 0xC142

6: movsx eax, bx ; EAX = 0xFFFFC142

movsx → sign extension; eax → 32-bit register; bx → 16-bit register

Converting C142h to binary:

1100 0001 0100 0010$_2$

Performing sign extension on 1100 0001 0100 0010$_2$ (1 = MSb)

1111 1111 1111 1111 1100 0001 0100 0010$_2$

Converting 1111 1111 1111 1111 1100 0001 0100 0010$_2$ to hexadecimal:

(1111) (1111) (1111) (1111) (1100) (0001) (0100) (0010)$_2$

F F F F C 1 4 2$_{16}$

7: movsx edx, bl ; EDX = 0x00000042

edx → 32-bit

bx = 0xC142 (bh = C1, bl = 42)

bl = 0x42 8-bit

0x42 = 0100 0010$_2$ (0 = MSb)

8: movsx cx, bh ; cx = 0xFFC1

cx → 16-bit; bh → 8-bit

bh = 0xC1

0xC1 = 1100 0001$_2$ (1 = MSb)

After performing sign extension:

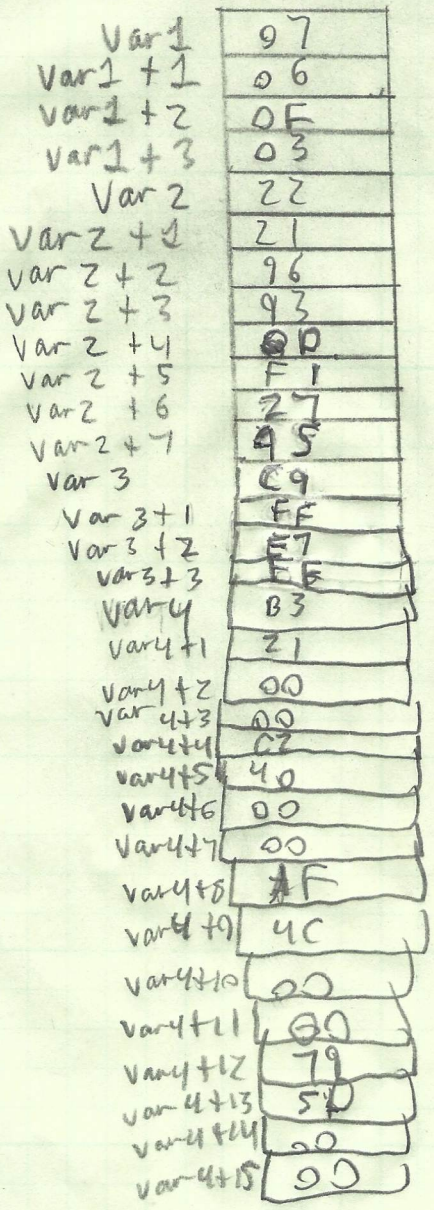
1100 0001$_2$ => 1111 1111 1100 0001$_2$

F F C 1$_{16}$

6 (In-direct Little Endian)

```
.data
var1 BYTE  7, 6, 0Fh, 3
var2 WORD  2122h, 9396h, 0F10Dh, 9527h
var3 SWORD -55, -25
var4 DWORD 21B3, 40C2, 4CAF, 5D79
```

memory map (Little Endian):



Converting -55_{10} to 2's comp hex

Lsb

16 | 55

16 | 3 r. 7

0 r. 3

↑ Msb

37_{16}

ff

− 37

C8

+ 1

$ffC9_{16} = -55_{10}$

converting -25_{10} to 2's comp hex

16 | 25 (Lsb ↓)

16 | 1 r. 9

0 r. 1 (↑ Msb)

19_{16}

ff

− 19

E6

+ 1

$ffE7_{16} = -25_{10}$

6. (cont)

Answers

```
.code
mov ax, [var1+1]   ; AX = 0F06h
mov ax, [var2+2]   ; AX = 9396h
mov ax, var3       ; AX = FFC9h
mov ax, [var3-2]   ; AX = 9527h
```

* var3-2 ≡ var2+6 according to Little Endian Memory Map on page 6.

7. Write a program that prints your <FirstName Lastname> on your screen. You can use the template provided. Assemble and generate the output using MASM and Visual Studio. Embed your output in your submission.

```
TITLE My first assembly program
INCLUDE Irvine32.inc
.DATA

message BYTE "Daniel Benjamin", 0

.CODE
main PROC
      mov edx, offset message
      Call WriteString
exit
main ENDP
END main
```