Comp 3350: Computer Organization & Assembly Language
HW # 4: Theme: Debugging, Flags, Data Declarations

All main questions carry equal weight.
(Credit awarded to only those answers for which work has been shown.)

Note: For Problems 1 to 6, you must NOT program on computer. Only handwritten answers are required. For Problem 7, you must code the program inside the Visual Studio/MASM environment and submit a screenshot showing the output of you running your code. You must also provide your code in the Word/PDF file you are submitting. The above-mentioned screenshots should also be embedded in the same file.

1. A. Write a program fragment that will reset the sign flag
   B. Write a program fragment that will reset the overflow flag.
   C. Write a program fragment that resets the zero flag.
   D. What will be the value of the parity flag after the following lines execute?
      Mov al, 8
      Add al, 5 3.5 points * 4

   Ans:
```
A.  .CODE
       mov al, FFh
       add al, 0                   ; the sign flag = 1
       add al, 5h                  ; reset the sign flag = 0
B.  .CODE
       mov al, 7Fh
       mov bl, 7Fh
       add al, bl                  ; the overflow flag = 1
       add al, 0                   ; reset the overflow flag = 0
C.  .CODE
       mov al, 0h
       add al, 1h                  ; the zero flag = 1
       sub al, 1h                  ; reset the zero flag = 0
D.
       mov al, 8h                  ; al = 0000 1000
       add al, 5h                  ; al = 0000 1011
```
   The number of ones is not even, so the parity flag = 0

2. Given the following data declarations:

```
.DATA
Alpha  WORD 1Ah, 2Bh, 3Ch, 4CH, 5C00, 6D03, 7F1A
SUM    WORD ?

.CODE
;Write instructions that sum the odd elements of the array into AX and then
save the resultant sum in the location SUM.
```

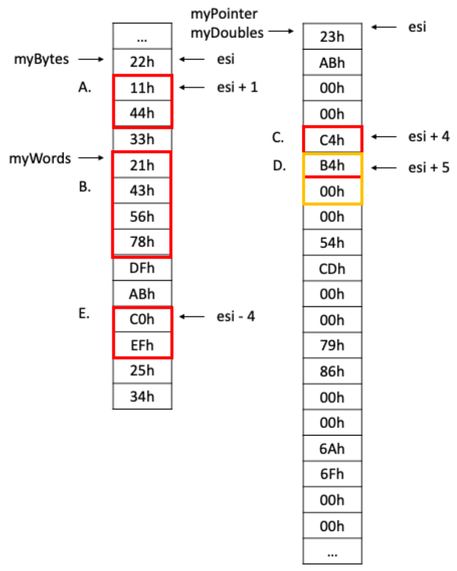
```
; sum of odd index elements
mov ax, 0               ; ax = 0          2 points
add ax, [Alpha]         ; ax += 1Ah     2.5 points
add ax, [Alpha + 4]     ; ax += 3Ch     2.5 points
add ax, [Alpha + 8]     ; ax += 5C00h   2.5 points
add ax, [Alpha + 12]    ; ax += 7F1Ah   2.5 points
mov SUM, ax             ; SUM = ax        2 points
```

```
; sum of odd numbers
mov ax, 0               ; ax = 0           2 points
add ax, [Alpha + 2]     ; ax += 2Bh        5 points
add ax, [Alpha + 10]    ; ax += 6D03h      5 points
mov SUM, ax             ; SUM = ax         2 points
```

3. Fill in the requested register values after executions of the instructions:
   Show the memory map using an address-data table. 2 points * 5

```
.data
myBytes      BYTE   22h, 11h, 44h, 33h
myWords      WORD   4321h,7856h, ABDFh, EFC0h, 3425h
myDoubles    DWORD  AB23h, B4C4h, CD54h, 8679h, 6F6Ah
myPointer    DWORD  myDoubles

.code
mov esi, OFFSET myBytes
mov ax, WORD PTR [esi+1]          ; A.    AX =   4411h
mov eax, DWORD PTR myWords        ; B.    EAX = 78564321h
mov esi, myPointer
mov ax, WORD PTR [esi+4]          ; C.    AX =   B4C4h
mov ax, WORD PTR [esi+5]          ; D.    AX =   00B4h
mov ax, WORD PTR [esi-4]          ; E.    AX =   EFC0h
```

memory map – 4 points



4. What is the value of ax after the following instructions?

```
.data
myArray WORD 4 DUP (5), 21, 4, 65, 0CDE
.code
mov ax, TYPE myarray
mov ax, lengthof myarray
mov ax, sizeof my array
```

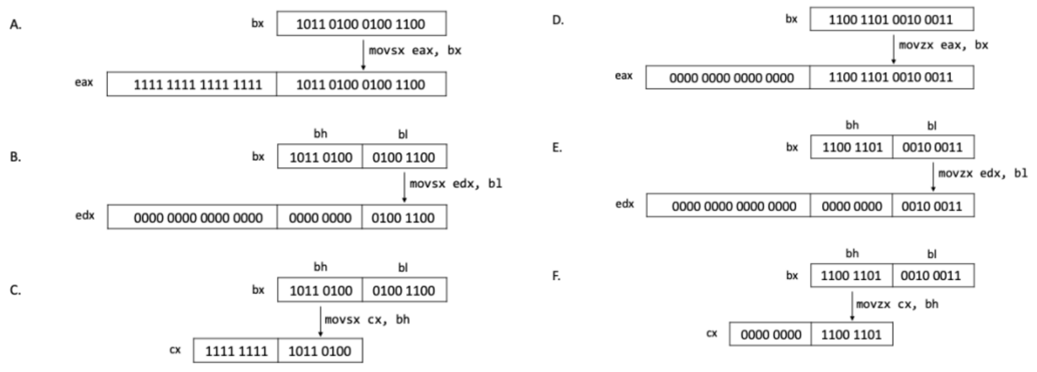
Ans: 4.5 points * 3

```
mov ax, TYPE myarray            ; TYPE returns the size of a WORD
                                ; ax = 2
mov ax, lengthof myarray        ; lengthof returns the number of
                                ; elements in myArray
                                ; ax = 4 + 1 + 1 + 1+ 1= 8
mov ax, sizeof my array         ; size of = lenghtof × TYPE
                                ; ax = 2 × 8 = 16
```

5. Fill in the requested register values after executions of the instructions (Do not let your eyes deceive you. There are some mov**S**x instructions and some mov**Z**x instructions.): 2.5 points * 5

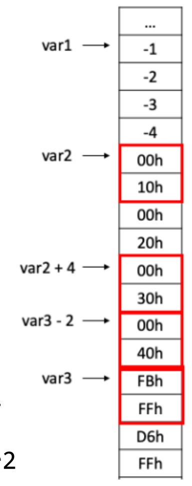
```
.code
mov bx, 0B44Ch
movsx eax, bx                        ; A.   EAX = FFFF B44Ch
movsx edx, bl                        ; B.   EDX = 0000 004Ch
movsx cx, bh                         ; C.   CX = FFB4h

mov bx, 0CD23h
movzx eax, bx                        ; D.   EAX = 0000 CD23h
movzx edx, bl                        ; E.   EDX = 0000 0023h
movzx cx, bh                         ; F.   CX = 00CDh
```

Ans:



6. What will be the value of the destination operand after each of the following instructions execute? 3.5 points * 4

```
.data
var1 SBYTE   -1, -2, -3, -4
var2 WORD    1000h, 2000h, 3000h, 4000h
var3 SWORD   -21, -42
var4 DWORD   10A0, 20B0, 30C0, 40D0
.code
mov ax, var2
mov ax, [var2+4]
mov ax, var3
mov ax, [var3-2]
```



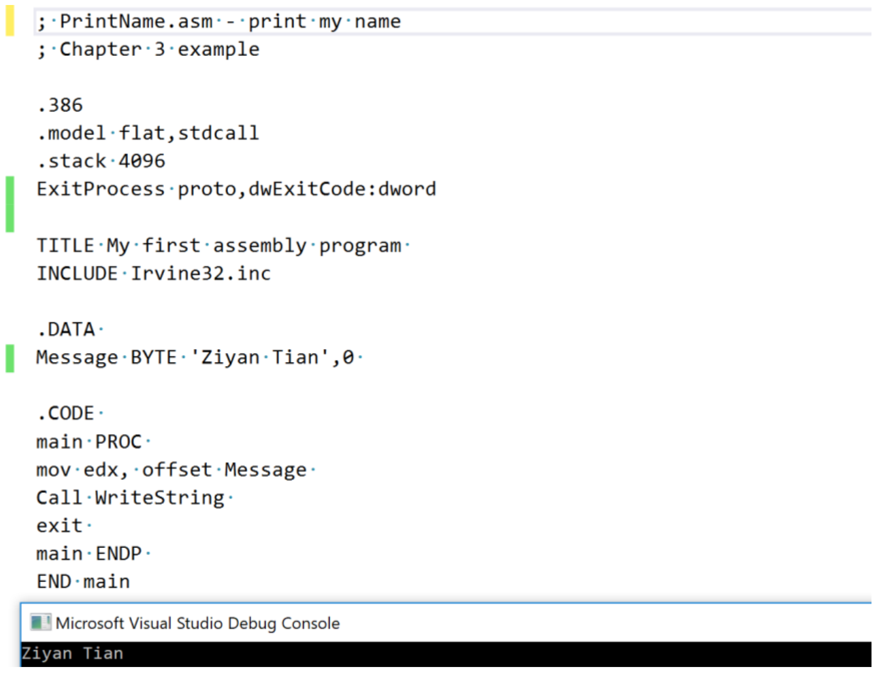
Ans:

```
mov ax, var2              ; get the first element in var2
                          ; ax = 1000h
mov ax, [var2+4]          ; get the second element in var2
                          ; ax = 3000h
mov ax, var3              ; get the first element in var3
                          ; -21 16-bit 2'complement = 1111 1111 1110 1011
                          ; ax = FFEBh
mov ax, [var3-2]          ; var3 – 2 points to the last element in var2
                          ; ax = 4000h
```

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 “FirstName Lastname”,0
.CODE
main PROC
        mov edx, offset message
        Call WriteString
        exit
main ENDP
END main
```

Ans:



```
; PrintName.asm - print my name
; Chapter 3 example

.386
.model flat,stdcall
.stack 4096
ExitProcess proto,dwExitCode:dword

TITLE My first assembly program
INCLUDE Irvine32.inc

.DATA
Message BYTE 'Ziyan Tian',0

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

Microsoft Visual Studio Debug Console

```
Ziyan Tian
```