Comp 3350: Computer Organization & Assembly Language
HW # 6: Theme: Arithmetic and Procedures
All main questions carry equal weight.
(Credit awarded to only those answers for which work has been shown.)

1. [Arithmetic Expression] Write a program that computes the following arithmetic expression:
   EAX = -Val3 + Val1 – (Val2 + Val4)

   Use the following data definitions:

```
Val1 SWORD  -8
Val2 SWORD  -21
Val3 SWORD   36
Val4 SWORD   93
```

```
        TITLE COMPUTE ARITHMETIC EXPRESSION
        include Irvine32.inc
        .data
        Val1 SWORD -8
        Val2 SWORD -21
        Val3 SWORD 36
        Val4 SWORD 93
        .code
        main PROC
                movsx eax, Val2
                movsx ebx, Val4
                add eax, ebx
                neg eax
                movsx ebx, Val3
                neg ebx
                push eax
                movsx eax, Val1
                add ebx, eax
                pop eax
                add eax, ebx
                exit
        main ENDP
END main
```

2. [Arrays] Write a program that:

   1) Prompts the user for integer input a dozen times
   2) Stores these inputs in an array
   3) Displays the stored array values on the screen using WriteInt (not DumpRegs).

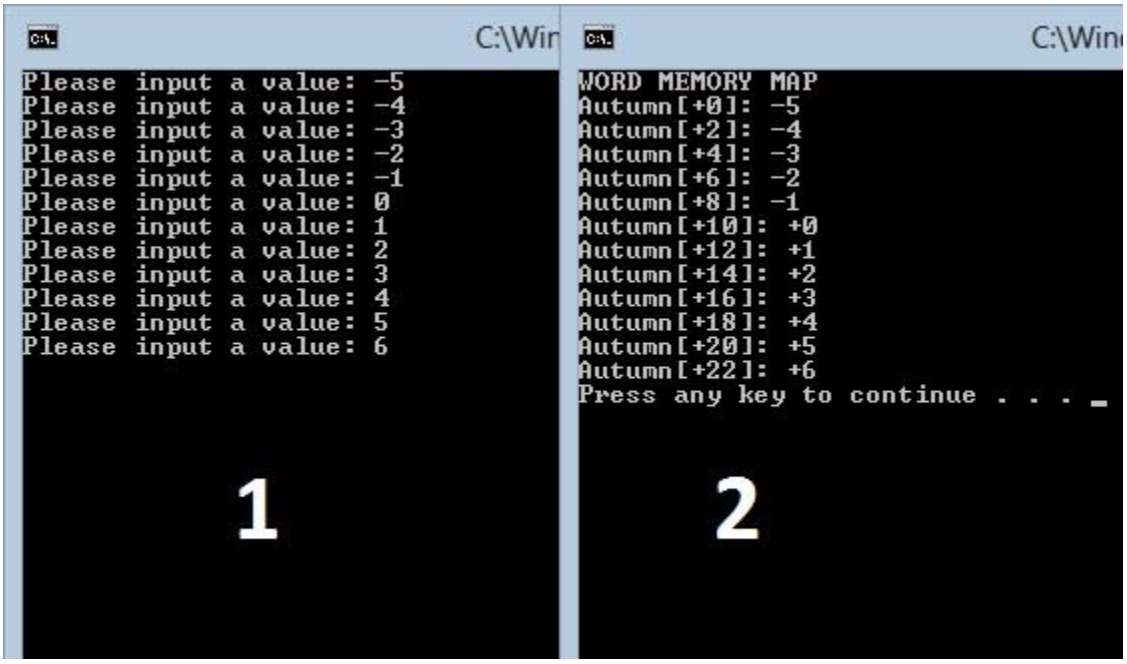
   In your submission, please embed the full program (.lst file) and one screen shot with at least one positive and one negative input value.

   Use the following:

```
.data
Prompt1  BYTE "Please input a value:", 0
Autumn   WORD 12 DUP(?)
```

```
TITLE PROMPT INPUT AND DISPLAY INTS
include Irvine32.inc
LengthOfArray =  LENGTHOF Autumn
TypeOfArray = TYPE Autumn
.data
Autumn    WORD     12       DUP(?)
.code
main PROC
CALL PopulateArray
CALL DisplayArray
          exit
main ENDP
;=====================================
;PopulateArray prompts the user for an integer and
; then stores the input upto 16bit into the Array
;=====================================
PopulateArray Proc
.data
Prompt1  BYTE "Please input a value: ",0
.code
mov edx, OFFSET Prompt1 ; places the offset of the message to edx reg
mov ecx, LengthOfArray  ; places the length of the array to ecx and this initalizes the loop
mov ebx, 0
LP1:
          CALL WriteString ;Writes whatever the address in edx points to
          CALL ReadInt ;Stores input into eax
          mov [Autumn + ebx], ax
          add ebx, TYPE Autumn
Loop LP1
ret
PopulateArray ENDP

;=====================================
;The DisplayArray displays each element of the
; array and shows the offset of where the element is stored
; in memory thus shows the memory map pertaining to the width of a word.
;=====================================
DisplayArray Proc
.data
DisplayMsg BYTE "WORD MEMORY MAP",0
ElementMsg BYTE "Autumn[",0
ElementMsg2 BYTE "]: ",0
.code
mov ecx, LengthOfArray
mov ebx, 0
          Call Clrscr ; Clears the Screen before displaying the array
          mov edx, OFFSET DisplayMsg
          Call WriteString   ; Displays "WORD MEMORY MAP"
          Call Crlf
LP2:
          mov edx, OFFSET ElementMsg
          Call WriteString ; displays Autumn[
```

```
          mov eax, ebx
          Call WriteInt       ; displays Autumn[# #-The displacement from the Array's Address thus the offset
          mov edx, OFFSET ElementMsg2
          Call WriteString ; displays Autumn[#]:
          movsx eax, [Autumn + ebx]
          Call WriteInt
          Call Crlf
          add ebx, TYPE Autumn
Loop LP2
ret
DisplayArray ENDP
end main
```



3. [Compares, Procedures] Write a procedure, *MinArray* that computes the minimum value stored in the array *Autumn*. Write a main program that calls *MinArray* and prints the minimum value found.

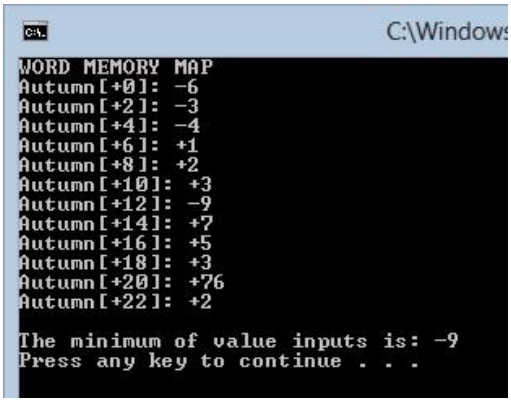
   Use the following:

```
.data
prompt      BYTE "Please input a value: ", 0
spacing     BYTE ", ",0;
result      BYTE "The minimum of value inputs is: ",0
Autumn      WORD 12 DUP(?)
```

   In your submission, please embed the full program (.lst file) and one screen shot showing the minimum value found.

```
TITLE PROMPT INPUT AND DISPLAY INTS AND LOWEST VALUE
include Irvine32.inc
LengthOfArray =  LENGTHOF Autumn
TypeOfArray = TYPE Autumn
.data
Autumn    WORD     12          DUP(?)
.code
main PROC
CALL PopulateArray
CALL DisplayArray
CALL MinArray
            exit
main ENDP
;====================================
;PopulateArray prompts the user for an integer and
; then stores the input upto 16bit into the Array
;====================================
PopulateArray Proc
.data
Prompt1    BYTE "Please input a value: ",0
.code
mov edx, OFFSET Prompt1 ; places the offset of the message to edx reg
mov ecx, LengthOfArray  ; places the length of the array to ecx and this initalizes the loop
mov ebx, 0
LP1:
            CALL WriteString ;Writes whatever the address in edx points to
            CALL ReadInt ;Stores input into eax
            mov [Autumn + ebx], ax
            add ebx, TYPE Autumn
Loop LP1
ret
PopulateArray ENDP

;====================================
;The DisplayArray displays each element of the
; array and shows the offset of where the element is stored
; in memory thus shows the memory map pertaining to the width of a word.
;====================================
DisplayArray Proc
.data
DisplayMsg BYTE "WORD MEMORY MAP",0
ElementMsg BYTE "Autumn[",0
ElementMsg2 BYTE "]: ",0
.code
mov ecx, LengthOfArray
mov ebx, 0
            Call Clrscr ; Clears the Screen before displaying the array
            mov edx, OFFSET DisplayMsg
            Call WriteString        ; Displays "WORD MEMORY MAP"
```

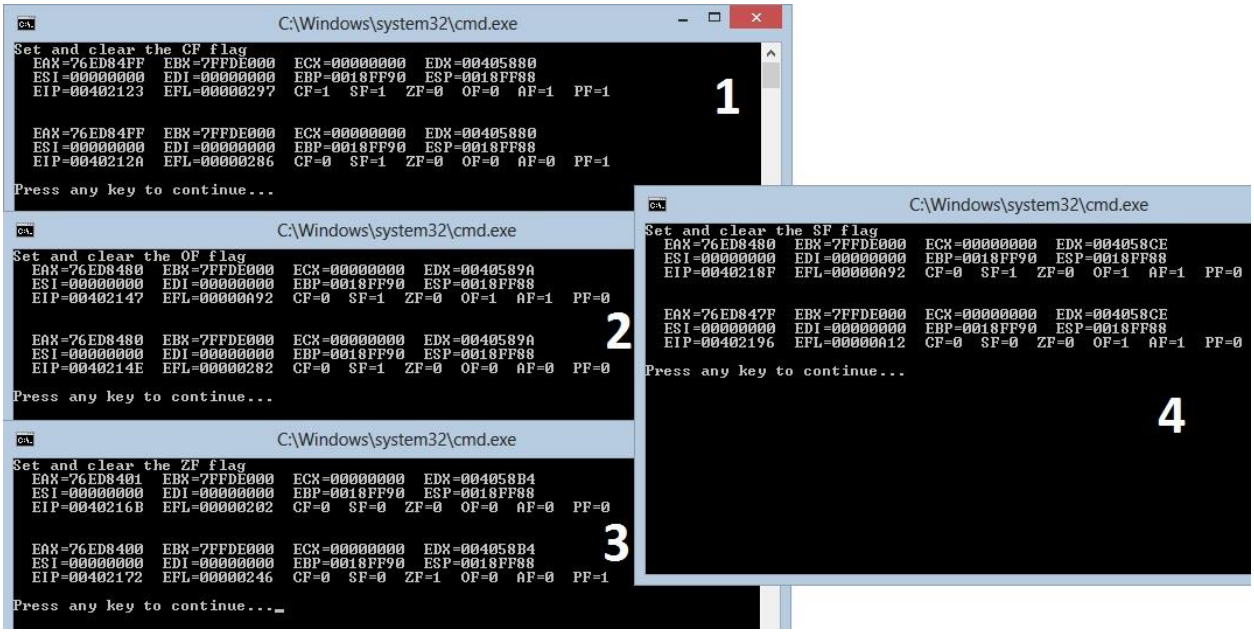
```
		Call Crlf
LP2:
		mov edx, OFFSET ElementMsg
		Call WriteString ; displays Autumn[
		mov eax, ebx
		Call WriteInt		; displays Autumn[# #-The displacement from the Array's Address thus the offset
		mov edx, OFFSET ElementMsg2
		Call WriteString ; displays Autumn[#]:
		movsx eax, [Autumn + ebx]
		Call WriteInt
		Call Crlf
		add ebx, TYPE Autumn
Loop LP2
ret
DisplayArray ENDP

;=======================================
;MinArray Find the minimum integer a array.
;=======================================
MinArray PROC
.data
result BYTE "The minimum of value inputs is: ",0
.code
mov edx, OFFSET result
mov ecx, LENGTHOF Autumn - 1 ;instalize the loop with the length -1
mov ax, [Autumn]
mov bx, TYPE Autumn

LP2:
		cmp [Autumn + bx], ax ; compares an element in the array with the next element
		JL SwitchVal ; Jump condition if the carry flag is set meaning that the destination is less than the source
		LPReturn:  ; A label to get back to the flow after exchanging values with a smaller value
		add ebx, TYPE Autumn ; adds the type to get to the next word value of the array
loop LP2

jmp quit ; returns to recommended flow of the program

SwitchVal: ; replaces the previous minimum value with the present minimum value and returns inside LP2
mov ax, [Autumn + bx]
jmp LPReturn

quit:
CALL Crlf
CALL WriteString
movsx eax, ax ; moves the 16bit portion back to the 32bit portion to bring with it the sign of the number
CALL WriteInt
CALL Crlf
ret
MinArray ENDP

end main
```



4. [Flags] Write a program which has the following parts:
    a. Use the Add and Sub instructions to set and clear the CF flag
    b. Use the Add and Sub instructions to set and clear the OF flag
    c. Use the Add and Sub instructions to set and clear the ZF flag
    d. Use the Add and Sub instructions to set and clear the SF flag

   Submit the list file as well as appropriate trace of the runs.

```
TITLE set  and clear flags
include Irvine32.inc

.data
STR1 BYTE "Set and clear the CF flag",0
STR2 BYTE "Set and clear the OF flag",0
STR3 BYTE "Set and clear the ZF flag",0
STR4 BYTE "Set and clear the SF flag",0
.code
main PROC

;a.      Use the Add and Sub instructions to set and clear the CF flag
;b.      Use the Add and Sub instructions to set and clear the OF flag
;c.      Use the Add and Sub instructions to set and clear the ZF flag
;d.      Use the Add and Sub instructions to set and clear the SF flag

;Change the Carry Flag CY
mov edx, OFFSET STR1
CALL WriteString
mov al, 00h ;     , al=00
sub al, 01h ; CY=1, al=FF
CALL DumpRegs
add al, 00h ; CY=0, al=FF
CALL DumpRegs
CALL WaitMsg
CALL Clrscr

;Change the Overflow Flag
mov edx, OFFSET STR2
CALL WriteString
mov al, +127;       al=7F
add al, 1   ; OV=1, al=80
CALL DumpRegs
sub al, 0   ; OV=0, al=80
CALL DumpRegs
CALL WaitMsg
CALL Clrscr

;Change the Zero Flag
mov edx, OFFSET STR3
CALL WriteString
mov al, 0   ;       al=0
add al, 1   ; ZR=0, al=1
CALL DumpRegs
sub al, 1   ; ZR=1, al=0
CALL DumpRegs
CALL WaitMsg
CALL clrscr

;Change the Sign Flag
mov edx, OFFSET STR4
CALL WriteString
mov al, 7Fh  ;       al=7F
add al, 1h   ; PL=1, al=80
CALL DumpRegs
sub al, 1h   ; PL=0, al=7F
CALL DumpRegs
CALL WaitMsg
CALL clrscr

exit
main ENDP
END main
```


C:\Windows\system32\cmd.exe
Set and clear the CF flag
EAX=76ED84FF EBX=7FFDE000 ECX=00000000 EDX=00405880
ESI=00000000 EDI=00000000 EBP=0018FF90 ESP=0018FF88
EIP=00402123 EFL=00000297 CF=1 SF=1 ZF=0 OF=0 AF=1 PF=1
EAX=76ED84FF EBX=7FFDE000 ECX=00000000 EDX=00405880
ESI=00000000 EDI=00000000 EBP=0018FF90 ESP=0018FF88
EIP=0040212A EFL=00000286 CF=0 SF=1 ZF=0 OF=0 AF=0 PF=1
Press any key to continue...
1
C:\Windows\system32\cmd.exe
Set and clear the OF flag
EAX=76ED8480 EBX=7FFDE000 ECX=00000000 EDX=0040589A
ESI=00000000 EDI=00000000 EBP=0018FF90 ESP=0018FF88
EIP=00402147 EFL=00000A92 CF=0 SF=1 ZF=0 OF=1 AF=1 PF=0
EAX=76ED8480 EBX=7FFDE000 ECX=00000000 EDX=0040589A
ESI=00000000 EDI=00000000 EBP=0018FF90 ESP=0018FF88
EIP=0040214E EFL=00000282 CF=0 SF=1 ZF=0 OF=0 AF=0 PF=0
Press any key to continue...
2
C:\Windows\system32\cmd.exe
Set and clear the ZF flag
EAX=76ED8401 EBX=7FFDE000 ECX=00000000 EDX=004058B4
ESI=00000000 EDI=00000000 EBP=0018FF90 ESP=0018FF88
EIP=0040216B EFL=00000202 CF=0 SF=0 ZF=0 OF=0 AF=0 PF=0
EAX=76ED8400 EBX=7FFDE000 ECX=00000000 EDX=004058B4
ESI=00000000 EDI=00000000 EBP=0018FF90 ESP=0018FF88
EIP=00402172 EFL=00000246 CF=0 SF=0 ZF=1 OF=0 AF=0 PF=1
Press any key to continue...
3
C:\Windows\system32\cmd.exe
Set and clear the SF flag
EAX=76ED8480 EBX=7FFDE000 ECX=00000000 EDX=004058CE
ESI=00000000 EDI=00000000 EBP=0018FF90 ESP=0018FF88
EIP=0040218F EFL=00000A92 CF=0 SF=1 ZF=0 OF=1 AF=1 PF=0
EAX=76ED847F EBX=7FFDE000 ECX=00000000 EDX=004058CE
ESI=00000000 EDI=00000000 EBP=0018FF90 ESP=0018FF88
EIP=00402196 EFL=00000A12 CF=0 SF=0 ZF=0 OF=1 AF=1 PF=0
Press any key to continue...
4