Assembly Language Lab 5 HandOn Solution

1- Write an assembly program that copies an array to another array. The array is entered by the user.

```
arrSize EQU 10

.data

arr1 SDWORD arrSize DUP(?)
arr2 SDWORD arrSize DUP(?)

strPrompet BYTE "Please enter the values of array 1",0
strPrompetResult BYTE "The values of array 2 are : ",0

.code
main PROC

mov edx, offset strPrompet
call writestring
call crlf

;read the values of array 1 from the user
mov esi, offset arr1
mov ecx, arrSize            ;loop counter

L1:
       call readInt
       mov [esi], eax              ; copy the read value to arr 1
       add esi, TYPE arr1

Loop L1

;copy the values from array 1 to array 2
mov esi, offset arr1 ; source pointer
mov edi, offset arr2 ; destination pointer
mov ecx, arrSize            ;loop counter

L2:
       mov eax, [esi]
       mov [edi], eax
       add esi, TYPE arr1
       add edi, TYPE arr2

Loop L2

;display arr2 on the screen
mov edx, offset strPrompetResult
call writestring
call crlf

mov esi, offset arr2
mov ecx, arrSize            ;loop counter

L3:
```

```
        mov eax, [esi]
        call writeint
        call crlf
        add esi, TYPE arr1
Loop L3

        exit
main ENDP
```

2- Write an assembly program that calculates and prints the total of an entered integer array.

```
arrSize EQU 10

.data

arr1 SDWORD arrSize DUP(?)
sum SDWORD 0

strPrompet BYTE "Please enter the values of array 1",0
strPrompetResult BYTE "The sum = ",0

.code
main PROC

mov edx, offset strPrompet
call writestring
call crlf

;read the values of array 1 from the user and add it to sum
mov esi, offset arr1
mov ecx, arrSize            ;loop counter

L1:
        call readInt
        mov [esi], eax              ; copy the read value to arr 1
        add sum, eax          ; accumulate the result in sum
                                    ;the value is still in eax
        add esi, TYPE arr1
Loop L1

; output the sum
mov edx, offset strPrompetResult
call writestring
call crlf

mov eax, sum
call writeint
call crlf

        exit
main ENDP
```

3- Write an assembly program that finds the min and max of an entered integer array.

```
arr_size equ 5
.data
arr DWORD arr_size DUP(0)
min DWORD 9fffffffh
max DWORD 0

strPrompt BYTE 'Enter array items:', 0
strMinMsg BYTE 'The min value = ', 0;a message to be displayed to user
strMaxMsg BYTE 'The Max value = ', 0;a message to be displayed to user

.code
main PROC
	mov edx, offset strPrompt ;put the address of strPrompt in edx
	call WriteString;as a parameter to WriteString
	MOV ESI, OFFSET arr
	MOV ECX, arr_size
L1:
	CALL ReadDec
	MOV [ESI], EAX    ;what saved frm ReadDec
	ADD ESI, TYPE arr ;update esi

	CMP EAX, min ;compare with min
	JB updateMin ;update min if there is a new min

	CMP EAX, max ;compare with max
	JA updateMax ;update max if there is a new max
LOOP L1      ;this line will be hit ... when??
JMP next

updateMin:
	MOV min, EAX  ;update the min
	Loop L1             ;continue loop if ECX > 0
	JMP next            ;GOTO the end if loop is done

updateMax:
	MOV max, EAX  ;update the max
	Loop L1             ;continue loop if ECX > 0
	JMP next            ;GOTO the end if loop is done

next:
	MOV EDX, OFFSET  strMinMsg ;print the min msg
	CALL WriteString

	MOV EAX, min              ;print the min
	Call WriteDec
	CALL CRLF

	MOV EDX, OFFSET strMaxMsg ;print the max msg
	CALL WriteString

	MOV EAX, max
	CALL WriteDec            ;display the max
	Call CRLF
```

```
exit
main ENDP
```

4- Write an assembly program that inputs a student's score and prints her/his grade.
The student will be "Excellent" if her/his score is between 90-100, "Very Good" if the score is between 80-89, "Good" if the score is between 70-79, "Fair" if the score is between 60-69, and "Fail" if the score is lower than 60. (Assume scores are integers.)

```
.data

       prompt BYTE "Enter a student score: ",0

      ;The output msg
       promptA BYTE "Excellent",0
       promptB BYTE "Very Good",0
       promptC BYTE "Good",0
       promptD BYTE "Fair",0
       promptF BYTE "Fail",0
       promptE BYTE "Error!!",0

.code
main PROC
       mov edx, offset prompt
       call writestring
       call readdec
.IF eax > 100 || eax < 0
       mov edx, offset promptE
.ELSEIF eax >= 90
       mov edx, offset promptA
.ELSEIF eax >= 80
       mov edx, offset promptB
.ELSEIF eax >= 70
       mov edx, offset promptC
.ELSEIF eax >= 60
       mov edx, offset promptD
.ELSE
       mov edx, offset promptF
.ENDIF

call writestring ;print the output msg based on the prev selection

call crlf
exit
main ENDP
```

5- Write an assembly program that accespts multiple scores and prints the grade. (the program should ask the user if he wants to continue)
The student will be “Excellent” if her/his score is between 90-100, “Very Good” if the score is between 80-89, “Good” if the score is between 70-79, “Fair” if the score is between 60-69, and “Fail” if the score is lower than 60. (Assume scores are integers.)

```
scores_cnt equ 5
.data
	prompt byte "Enter 5 scores: ",0

	promptA byte "Studens Have Excellent ",0
	CntA DWORD 0

	promptB byte "Students Have Very Good ",0
	CntB DWORD 0

	promptC byte "Students Have Good ",0
	CntC DWORD 0

	promptD byte "Students Have Fair ",0
	CntD DWORD 0

	promptF byte "Students Have Fail ",0
	CntF DWORD 0

	promptE byte "Error!!",0

	stragain byte "Do you want to enter another score (Y/N)? ",0
.code
main PROC
.REPEAT
	mov edx, offset prompt
	call writestring
	call readdec
	.IF eax > 100 || eax < 0
		mov edx, offset promptE
	.ELSEIF eax >= 90
		mov edx, offset promptA
	.ELSEIF eax >= 80
		mov edx, offset promptB
	.ELSEIF eax >= 70
		mov edx, offset promptC
	.ELSEIF eax >= 60
		mov edx, offset promptD
	.ELSE
		mov edx, offset promptF
	.ENDIF

	mov edx, offset stragain
	call writestring
	call readchar ;input char stored in AL register
.UNTIL al == 'N' || al == 'n'
exit
main ENDP
```

6- Write an assembly program that accepts multiple students' scores, and then prints number of students in each grade. Grades and their score ranges are defined in the previous exercise. The program should also print an error message if the entered score is less than 0 or greater than 100.

```
scores_cnt equ 5
.data
	prompt byte "Enter 5 scores: ",0

	promptA byte "Studens Have Excellent ",0
	CntA DWORD 0

	promptB byte "Students Have Very Good ",0
	CntB DWORD 0

	promptC byte "Students Have Good ",0
	CntC DWORD 0

	promptD byte "Students Have Fair ",0
	CntD DWORD 0

	promptF byte "Students Have Fail ",0
	CntF DWORD 0

	promptE byte "Error!!",0

main PROC
	mov edx, offset prompt
	call writestring
	MOV ecx, scores_cnt

L1:
	call readdec
	.IF eax > 100 || eax < 0
		JMP errorState
	.ELSEIF eax >= 90
		INC CntA
	.ELSEIF eax >= 80
		INC CntB
	.ELSEIF eax >= 70
		INC CntC
	.ELSEIF eax >= 60
		INC CntD
	.ELSE
		INC CntF
	.ENDIF
LOOP L1
JMP next

errorState:
	MOV edx, OFFSET promptE
	call writestring
	call crlf
	JMP endState
```

```
next:
	MOV edx, OFFSET promptA
	call writestring
	MOV eax, CntA
	call writeDec
	call crlf

	MOV edx, OFFSET promptB
	call writestring
	MOV eax, CntB
	call writeDec
	call crlf

	MOV edx, OFFSET promptC
	call writestring
	MOV eax, CntC
	call writeDec
	call crlf

	MOV edx, OFFSET promptD
	call writestring
	MOV eax, CntD
	call writeDec
	call crlf

	MOV edx, OFFSET promptF
	call writestring
	MOV eax, CntF
	call writeDec
	call crlf

endState:
exit
main ENDP
```