****

**Green University of Bangladesh**

**Department of Computer Science and Engineering (CSE)**

**Faculty of Sciences and Engineering**

**Semester: (Fall: Year 2023), B.Sc. in CSE (Day)**

**Lab Report No:** 06

**Course Title:** Microprocessor & Microcontroller Lab

**Course Code:** CSE 304 **Section:** 213D2

**Lab Experiment Name:** Implement Procedure in Assembly Language Programming

**Student Details**

|  |  |  |
| --- | --- | --- |
| **Name** | | **ID** |
| **1.** | Md Masud Rana | 213902101 |

**Lab Date :** 26-10-2023

**Submission Date :** 03-12-2023

## **Course Teacher’s Name :** Sudip Ghoshal

|  |
| --- |
| **Lab Report Status**  **Marks: ………………………………… Signature:.....................**  **Comments:.............................................. Date:..............................** |

**1. TITLE OF THE LAB REPORT EXPERIMENT**

Implement Procedure in Assembly Language Programming

**2. OBJECTIVES/AIM**

* To understand the use Procedure in Assembly Language
* To understand sub routines with the transfer the execution of a program
* To understand the use of call and return instruction in procedure
* To understand program control transfer and Instruction Register(IR)

**3. PROCEDURE**

**Problem 1: find largest, smallest and average in an array of 5 numbers**

**Step 1:** start

**Step 2:** initialize data segment arr, i, avg and newline

**Step 3**: Include data segment into main procedure

**Step 4:** set pointer to the array, and initialize cx to 5 and take array element from user

**Step 5:** **call ‘findLargest’**.procedure

**Step 6:** Initialize pointer si, to ‘arr’ and i to 0

**Step 7:** Set max as first value into ‘bl’ register

**checklargest:**

Compare [si], bl

JGE swapL as mov bl, [si]

Increment si, i,

Check i and 5

Jump JL ‘**checkLargest’** level

JGE ‘printLargest’ level

**Step 8:** **call ‘findSmallest’** procedure

Similar to the **findLargest.** Just condition will be:

Compare [si], bl

JLE swapS as mov bl, [si]

**Step 9: call findAverage** tocalculate average

Set pointer si, to ‘arr’

Initialize Cx, 5 and al to 0

**calculateAVG loop**

store each element into ‘bl’ register and subtract 48 to calculate as decimal value

increment ‘si’

continue **calculateAVG loop**

**Step 10:** end

**Problem 2: Sorting array of 7 size in ascending and descending order**

**Step 1:** start

**Step 2:** initialize data segment arr, and newline

**Step 3**: Include data segment into main procedure

**Step 4:** set pointer si to ‘arr’ and cx to 7 and take array element from user and store into ‘arr’

**Step 5:** **call ‘sortAscending’**.procedure

**Step 6:** initialize si pointer to ‘arr’, inc ‘si’ and dx to si then dec ‘si’ and ch to 04

**OuterLoop**:

Load counter from ch into cl

Load index from dx into di

**InnerLoop**:

Load byte at [si] into al

Load byte at [di] into bl

Compare al and bl

If al >= bl, jump to **nexts**

Swap the bytes at [si] and [di]

**nexts**:

Increment di

Decrement cl

If cl is not zero, jump to **InnerLoop**

Increment si

Increment dx

Decrement ch

If ch is not zero, jump to **OuterLoop**

**Step 7:** call **printArray**

**Step 8:** call **‘sortDescending’**

**Step 9:** call initializeVariable

OuterLoop:

Load counter from ch into cl

Load index from dx into di

InnerLoop:

Load byte at [si] into al

Load byte at [di] into bl

Compare al and bl

If al <= bl, jump to nexts

Swap the bytes at [si] and [di]

nexts:

Increment di

Decrement cl

If cl is not zero, jump to InnerLoop

Increment si

Increment dx

Decrement ch

If ch is not zero, jump to OuterLoop

**Step 10:** call ‘**printArrray’**

**Step 11: e**nd

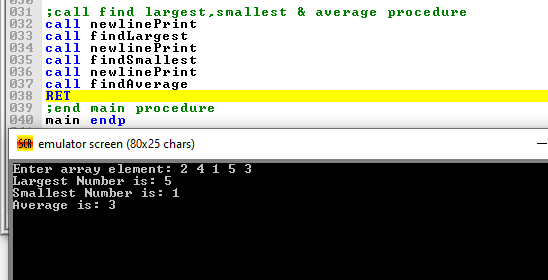
**4. IMPLEMENTATION**

**Problem 1: find largest, smallest and average in an array of 5 numbers**

2

1

|  |  |
| --- | --- |
| ;call= transfer the execution of the current program  ;ret = return to the execution of the program where we left  include 'emu8086.inc'  org 100h  .model small  .stack 100h  .data  arr db 5 dup(?)  i db ? ;iterator to continue loop  avg db ?  newline db 10, 13, "$"  .code  main proc  ;include data segment  mov ax, @data  mov ds, ax  mov si, offset arr ;set pointer to the array  mov cx, 5 ;since array size is 5    print 'Enter array element: '  L1:  call scan ;scan function call  mov [si], al  inc si ;increment array index  mov dl, 32 ;dl 32 for space  call printf  loop L1    ;call find largest,smallest & average procedure  call newlinePrint  call findLargest  call newlinePrint  call findSmallest  call newlinePrint  call findAverage  RET  ;end main procedure  main endp        ;find largest number in an array  findLargest proc  mov si, offset arr ;set pointer to the array  mov bl, [si] ;first value store in 'bl' as largest  mov i, 0 ;iterator for loop instead cx    checkLargest:  cmp [si], bl  JGE swapL  backLargest:  inc si  inc i ;increment both si & i  cmp i, 5  JL checkLargest  JGE printLargest    swapL: ;swap two value  mov bl, [si]  jmp backLargest ;back to the loop    ;print largest number  printLargest:  print 'Largest Number is: '  mov dl, bl  call printf    RET  findLargest endp | ;smallest number in an array  findSmallest proc  mov si, offset arr  mov bl, [si]  mov i, 0    ;loop start  checkSmallest:  cmp [si], bl  JLE swapS  backSmallest:  inc si  inc i  cmp i, 5  JL checkSmallest  JGE printSmallest  ;loop end    swapS: ;swap two value  mov bl, [si]  jmp backSmallest ;back to the loop    ;print largest number  printSmallest:  print 'Smallest Number is: '  mov dl, bl  call printf    RET  findSmallest endp    ;find average  findAverage proc  mov si, offset arr  mov cx,5  mov al,0  calculateAVG:  mov bl, [si]  sub bl, 48  add al, bl  inc si  loop calculateAVG  ;ax=al+ah.. al contains vlaue,so heigher bytes set 0  mov ah,0  mov dl, 5  div dl    print 'Average is: '  add al, 48  mov dl, al  call printf  RET  findAverage endp  ;newline print procedure  newlinePrint proc  mov ah, 9  lea dx, newline  int 21h  RET  newlinePrint endp    ;scan procedure to take input  scan proc  mov ah, 1  int 21h  RET  scan endp    ;print number  printf proc  mov ah, 2  int 21h  RET  printf endp  end main |

****Output:**

4

3

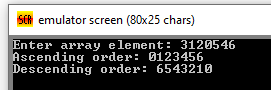
*Figure-1: Find largest, smallest and average in an array*

**Problem 2: Sorting array of 7 size in ascending and descending order**

2

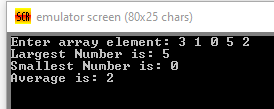
1

|  |  |
| --- | --- |
| ;sorting array in ascending and descending order  include 'emu8086.inc'  org 100h  .model small  .stack 100h  .data  arr db 7 dup(?)  newline db 10, 13, "$"  .code  main proc  mov ax, @data  mov ds, ax  ;take array element  mov si, offset arr  mov cx, 7  print 'Enter array element: '  take\_input:  mov ah, 1  int 21h  mov [si], al  inc si  loop take\_input    call printNewline  print 'Ascending order: '  call sortAscending  call printArray  call printNewline    print 'Descending order: '  call sortDescending  call printArray  call printNewline    main endp      ;sort ascending order  sortAscending proc  call initializeVariable  OuterLoops:  mov cl, ch  mov di, dx  InnerLoops:  mov al, [si]  mov bl, [di]  cmp al, bl  jc next  mov [si], bl  mov [di], al  next:  inc di  dec cl  jnz InnerLoops    inc si  inc dx  dec ch  jnz OuterLoops  RET  sortAscending endp | ;sort descending order  sortDescending proc  call initializeVariable  OuterLoop:  mov cl, ch  mov di, dx  InnerLoop:  mov al, [si]  mov bl, [di]  cmp al, bl  jnc nexts  mov [si], bl  mov [di], al  nexts:  inc di  dec cl  jnz InnerLoop  inc si  inc dx  dec ch  jnz OuterLoop  RET  sortDescending endp  ;setup some variable for sort  initializeVariable proc  mov si, offset arr  inc si  mov dx,si  dec si  mov ch, 06h ;cx to 6 means 7  RET  initializeVariable endp  ;print array  printArray proc  mov si, offset arr  mov cx, 7  arrLoop:  mov ah, 2  mov dl, [si]  int 21h  inc si  loop arrLoop  RET  printArray endp  ;print newline  printNewline proc  mov ah, 9  lea dx, newline  int 21h  RET  printNewline endp  end main |

**Output:**

*Figure-2: Sort an array in ascending and descending order*

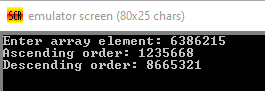
**5. TEST RESULT / OUTPUT**

**Problem 1:**

*Figure-3: Find largest, smallest and average from an array*

- The largest, smallest and average finder program run fine and calculated result is accurate showed in figure-1. Where the array was 2 4 1 5 3. Largest is 5 and smallest 1 and average is 3. Which is correct

- I have taken another test cases with different element. I have got the right answer. Showed in figure-3. In both test cases I have got the right answer. This ensure that my program run perfectly

**Problem 2:**

*Figure-4: Sort an array as ascending and descending order using procedure*

- The program properly sort in ascending and descending order of and array of size 7. In figure-2 i have showed a test result which is properly sorted

- Again I have taken an another array showed in figure-4. Which is also sorting the array

- since we have 7 size array so I have taken both cases 7 sizes different array element

**6. ANALYSIS AND DISCUSSION**

**Finding the Largest, Smallest, and Average:**

In the first problem of our lab, I have implemented a program on the 8086 microprocessor to find the largest, smallest, and average values in an array. To achieve this, I have employed a systematic approach, iterating through the array elements and comparing them to dynamically update the largest and smallest values. Additionally, I have accumulated the sum of all elements to later calculate the average. The implementation involved careful consideration of the data movement and comparison instructions in the x86 assembly language.

**Sorting in Ascending and Descending Order:**

* The second problem involved sorting the array in both ascending and descending orders. I have utilized the Bubble Sort algorithm, a straightforward yet effective sorting technique suitable for the constraints of the 8086 microprocessor. The algorithm iteratively compared adjacent elements and swapped them if they were in the wrong order. This process continued until the entire array was sorted.
* My implementation successfully arranged the elements in ascending and descending orders. However, it is essential to note that the Bubble Sort algorithm has a time complexity of O(n^2), where 'n' is the number of elements in the array.

**7. SUMMARY**

In summary, our lab work involved implementing essential algorithms for array manipulation on the 8086 microprocessor using procedure as well as we have got crystal clear about loop concept. I have successfully completed the challenges of finding the largest, smallest, and average values in an array, as well as sorting the array in both ascending and descending orders. While my solutions is effective for the given tasks. Overall, this lab provided valuable hands-on experience in programming for the 8086 microprocessor procedure, loops, array concepts