----------**Sample of Report**----------

CO215, CO Lab 2021, Assignment 9

Roll number: CSB19057

Name: PRASANJIT DUTTA

Objective:

1. To learn about procedures in assembly language
2. To learn passing of parameters (by value, by reference) through stack;
3. To learn writing own assembly language procedure.

Exercises:

----------**Bubble_Sort: Program to bubble sort an array of integers**----------

```
CR          EQU     0DH
LF          EQU     0AH
MAX_SIZE    EQU     10
.MODEL      SMALL
.STACK              100H
.DATA

array               DW      10 DUP (?)
prompt_msg          DB      'Enter single digit non-zero integers to be sorted.', CR, LF    ; CR=13 and
LF=10 are defined in the first two statements
                    DB      'Enter zero to terminate the input (at most 10).', CR, LF, '$'
output_msg          DB      'Array of numbers in sorted order:', CR, LF, '$'
nwln        DB      CR, LF, '$'

.CODE

main        PROC
    .STARTUP

    mov     ah, 09h
    lea     dx, prompt_msg
    int     21h

    mov     bx, OFFSET array        ; load address of array to bx
    mov     cx, MAX_SIZE
    sub     dx, dx                  ; clear dx

rd_loop:
    mov     ah, 01h
```

```
    int     21h
    sub     al, 30h                 ; converting char to num
    mov     ah, 0                   ; clearing ah, so that value of ax is the same as value of al.

    cmp     ax, 0                   ; if value is not 0, continue reading else exit
    je      stop_read
    mov     [bx], ax
    add     bx, 2
    inc     dx                       ; increment no. of integers
    loop    rd_loop

stop_read:
    push    dx                      ; push no. of integers (pass by value)
    mov     cx, dx
    push    OFFSET array            ; push array pointer (pass by reference)
    call    bubble_sort             ; call bubble sort procedure

    mov     ah, 09h
    lea     dx, output_msg
    int     21h


    mov     bx, OFFSET array


prt_loop:                   ; print sorted array
    mov     ah, 02h
    mov     dx, [bx]        ; since it is a single-digit number value of dx is same as value of dl
    add     dl, 30h
    int     21h

    mov     ah, 09h
    lea     dx, nwln
    int     21h

    add     bx, 2                   ; increment bx by 2 because every array location is of 2 bytes
    loop    prt_loop

    mov     ah ,4ch
    mov     al ,0
    int     21h
main        ENDP

SORTED              EQU     0
```

```
UNSORTED EQU     1
bubble_sort      PROC
    pusha                       ; push all register values
    mov     bp, sp
    mov     cx, [bp+20]         ; transfer no. of integers to cx
    mov     bx, [bp+18]         ; and integer array address to bx
; Bottom 16 locations in stack are occupied by the saved
; contents of 8 registers (ax, bx, cx, dx, sp, bp, si, di)

lp_outer:
    dec     cx
    jz      done                ; if cx=0, done
    mov     di, cx              ; else another iteration
    mov     dx, SORTED          ; assume status of array as sorted
    mov     si, bx              ; transfer array address to si

lp_inner:
    mov     ax, [si]
    cmp     ax, [si+2]
    jg      swap                ; swap if integer at [si]> at [si+2]

resume:
    add     si, 2
    dec     di
    jnz     lp_inner
    cmp     dx, SORTED
    je      done
    jmp     lp_outer

swap:
    xchg    ax, [si+2]          ; swap integers at [si] and [si+2]
    mov     [si], ax
    mov     dx, UNSORTED        ; mark status of array as unsorted
    jmp     resume

done:
    popa                        ; pop to respective register locations
    ret     4
bubble_sort      ENDP

END main
```

```
----------selection_Sort: Program to selection sort an array of integers----------

CR          EQU     0DH
LF          EQU     0AH
MAX_SIZE    EQU     10
.MODEL      SMALL
.STACK              100H
.DATA

array               DW      10 DUP (?)
prompt_msg          DB      'Enter single digit non-zero integers to be sorted.', CR, LF    ; CR=13 and
LF=10 are defined in the first two statements
                    DB      'Enter zero to terminate the input (at most 10).', CR, LF, '$'
output_msg          DB      'Array of numbers in sorted order:', CR, LF, '$'
nwln        DB      CR, LF, '$'

.CODE

main        PROC
    .STARTUP

    mov     ah, 09h
    lea     dx, prompt_msg
    int     21h

    mov     bx, OFFSET array        ; load address of array to bx
    mov     cx, MAX_SIZE
```

```
    sub     dx, dx                  ; clear dx


rd_loop:
    mov     ah, 01h
    int     21h
    sub     al, 30h                 ; converting char to num
    mov     ah, 0                   ; clearing ah, so that value of ax is the same as value of al.

    cmp     ax, 0                   ; if value is not 0, continue reading else exit
    je      stop_read
    mov     [bx], ax
    add     bx, 2
    inc     dx                       ; increment no. of integers
    loop    rd_loop

stop_read:
    push    dx                      ; push no. of integers (pass by value)
    mov     cx, dx
    push    OFFSET array            ; push array pointer (pass by reference)
    call    selection_sort          ; call bubble sort procedure

    mov     ah, 09h
    lea     dx, output_msg
    int     21h


    mov     bx, OFFSET array
```

```
prt_loop:                    ; print sorted array
    mov     ah, 02h
    mov     dx, [bx]         ; since it is a single-digit number value of dx is same as value of dl
    add     dl, 30h
    int     21h

    mov     ah, 09h
    lea     dx, nwln
    int     21h

    add     bx, 2                    ; increment bx by 2 because every array location is of 2 bytes
    loop    prt_loop

    mov     ah ,4ch
    mov     al ,0
    int     21h
main        ENDP

SORTED              EQU     0
UNSORTED EQU        1
selection_sort      PROC
    pusha ; push all register values
  mov bp, sp
  mov cx, [bp+20] ; transfer no. of integers to cx
  mov bx, [bp+18] ; and integer array address to bx
```

```
   ; Bottom 16 locations in stack are occupied by the saved
   ; contents of 8 registers (ax, bx, cx, dx, sp, bp, si, di)
   mov si, bx ; transfer array address to si
lp_outer:
   dec cx
   jz done ; if cx=0, done
   mov bp, 2 ; else another iteration
   mov ax, [si]
   mov dx,0
   mov di,0
lp_inner:
   cmp ax, [si+bp]
   jg update
   jmp rm
update:
   mov di, bp
   mov ax, [si+bp]
rm:
   add bp,2
   inc dx
   cmp dx,cx
   jz swap
   jmp lp_inner
resume:
   add si, 2
   jmp lp_outer
swap:
```

```
    xchg ax, [si] ; swap integers at [si] and min index
    mov bp,di
    mov [si+bp], ax
    jmp resume
done:
    popa ; pop to respective register locations
    ret 4

selection_sort      ENDP

END main
```

Observations:

BUBBLE SORT

INSTRUCTION NO.	Starting memory location	Machine Code in Hexadecimal
1	0732D	60096
2	0732E	8B139 EC236
3	07330	8B139 4E078 14020
4	07333	8B139 5E094 12018
5	07336	49073
6	07337	74116 25037
7	07339	8B139 F9249
8	0733B	BA186 00000 00000
9	0733E	8B139 F3243

10	07340	8B139 04004
11	07342	3B059 44068 02002
12	07345	7F127 0D013
13	07347	83131 C6131 02002
14	0734A	4F079
15	0734A	75117 F3243
16	0734D	83131 FA250 00000
17	07350	74116 0C012
18	07352	EB235 E2226
19	07354	87135 44068 02002
20	07357	89137 04004
21	07359	BA186 01001 00000
22	0735C	EB235 E9233
23	0735E	61097
24	0735F	C2194 04004 00000

SELECTION **SORT**

INSTRUCTION NO.	**Starting memory location**	**Machine Code in Hexadecimal**
1	0732D	60096
2	0732E	8B139 EC236
3	07330	8B139 4E078 14020
4	07333	8B139

		5E094 12018
5	07336	8B139 F3243
6	07338	49073
7	07339	74116 2C044
8	0733B	BD189 02002 00000
9	0733E	8B139 04004
10	07340	BA186 00000 00000
11	07343	BF191 00000 00000
12	07346	3B059 02002
13	07348	7F127 02002
14	0734A	EB235
04004		
15	0734C	8B139 FD253
16	0734E	8B139 02002
17	07350	83131 C5197 02002
18	07353	42066
19	07354	3B059 D1209
20	07356	74116 07007
21	07358	EB235 EC236
22	0735A	83131 C6198 02002
23	0735D	EB235 D9217
24	0735F	87135 04004
25	07361	8B139 EF239
26	07363	89137 02002

27	07365	EB235 F3243
28	07367	61097
29	07368	C2194 04004 00000

Comparison:

Program	No. of Instructions	Program Length (Bytes)	Number of Memory Operations	Remarks
Bubble Sort	24	53	6	
Selection Sort	29	62	7	

Learning Outcome:

1. **Parameter Passing:**

   In 8086 parameters can be passed using stacks, register ,pointers or memory by storing the parameters and accessing the values inside the procedures

2. **Use of Alternative Addressing Modes:**

   Different addressing modes specifies different ways of calculating effective memory address of an operand by using information held in registers or constants contained within a machine instruction or elsewhere

3. **Change of CS and IP values during procedure call and return:**

   If the CALL is made with in the same code segment as that of the segment the call instruction is present in, both (IP) and (CS) are pushed onto the stack. The return must correspondingly pop two words from the stack. In the case of the call a call to a procedure which is in a different segment from that which contains the CALL instruction is made, only the contents of IP will be saved and retrieved when call and return instructions are used.

   In case of the return instruction call ,the stack pointer is incremented by an additional four addresses after the IP or the IP and CS are popped off the stack.