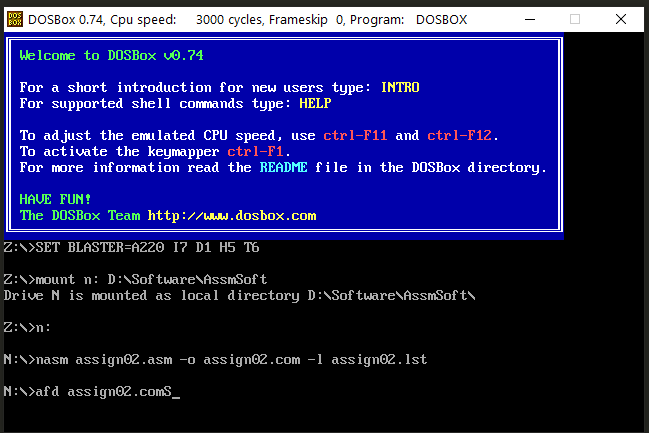
|  |  |  |
| --- | --- | --- |
| https://upload.wikimedia.org/wikipedia/commons/thumb/4/4e/VU_Logo.png/260px-VU_Logo.png | Computer Architecture and Assembly Language Programming (CS401)  Assignment No. 2 | Total marks = 20  Deadline Date:  June 3, 2021 |

|  |
| --- |
| Solution  NAME: TAMKEEN SAJJAD  ID: MC200400003  Course: MIT |
|  |

**Submission details**

# Assignment SORTING ONLy Program

## **Starting NASM and AFD**



## **Assembly language program.**

;bubble sorting

[org 0x0100]

jmp start

start:

mov bx,4; 4 will skip the first two values, 0 is default and start from beginning

mov byte[swap],0

compare:

mov ax, [vuid + bx]

cmp ax, [vuid + bx + 2]

jle noswap

;swaping

mov dx, [vuid + bx + 2]

mov [vuid + bx + 2], ax

mov [vuid + bx], dx

mov byte[swap],1

noswap:

add bx, 2

cmp bx, [len]

jne compare

cmp byte[swap],1

je start

mov ax, 0x4c00

int 0x21

;MC200400003

; 0, 2, 4, 6, 8,10,12, 14, 16, 18, 20

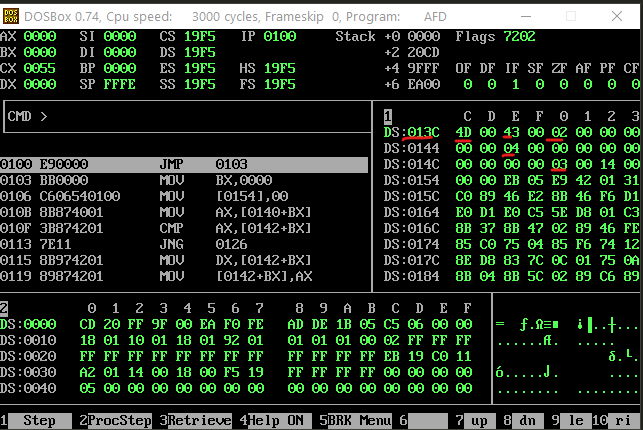
vuid: dw 'M', 'C', 2, 0, 0, 4, 0, 0, 0, 0, 3

len: dw 20; (length of VUID \*2 - 2)

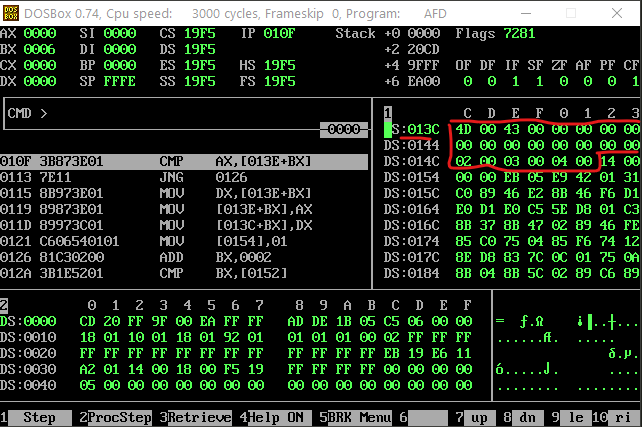
swap: db 0

## **Starting:**

**The VUID is shown at the location 0x013C, 0x4D indicates M, and 0x43 indicated C. The next array indicates the number of the ID: MC200400003.**

****

## **After execution:**

****

# Assignment SORTING and SUM Program

## **Assembly language program.**

;bubble sorting

[org 0x0100]

jmp start

start:

mov bx,4; 4 will skip the first two values, 0 is default and start from beginning

mov byte[swap],0

compare:

mov ax, [vuid + bx]

cmp ax, [vuid + bx + 2]

jle noswap

;swaping

mov dx, [vuid + bx + 2]

mov [vuid + bx + 2], ax

mov [vuid + bx], dx

mov byte[swap],1

noswap:

add bx, 2

cmp bx, [len]

jne compare

cmp byte[swap],1

je start

;summation

mov bx,vuid; point bx

add bx, 4; advance bx to 3rd number MC200400003

mov cx,9; load count of vuid in cx

mov ax,0; initialize sum to zero

l1:

add ax, [bx]; add number to ax

add bx, 2; advance bx to next number

sub cx, 1; numbers to be added reduced

jnz l1; if numbers remain add next

mov [total], ax; write back sum in memory

mov ax, 0x4c00

int 0x21

;MC200400003

; 0, 2, 4, 6, 8,10,12, 14, 16, 18, 20

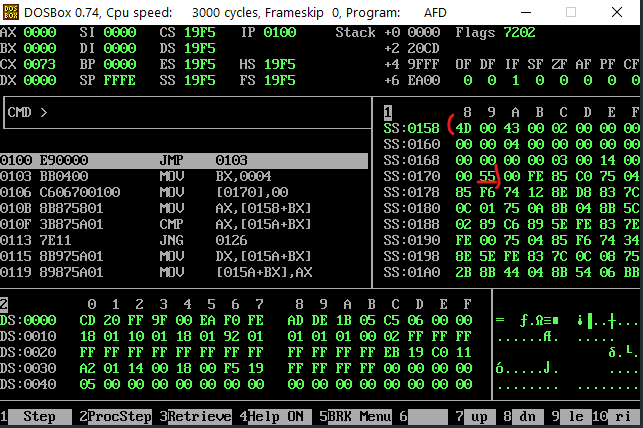
vuid: dw 'M', 'C', 2, 0, 0, 4, 0, 0, 0, 0, 3

len: dw 20; (length of VUID \*2 - 2)

swap: db 0

total: dw 0x55

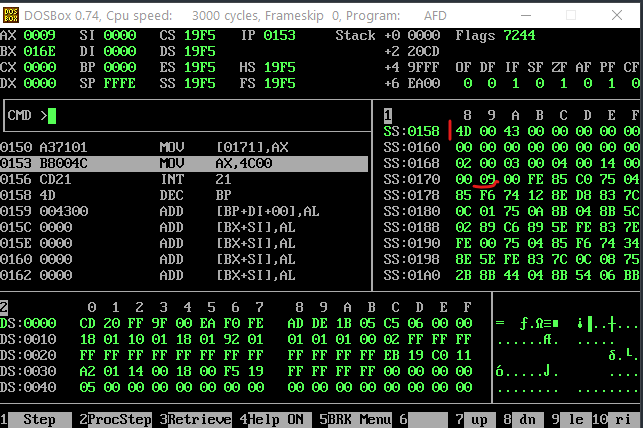
## **Starting:**



## **After execution:**

Sorted array at 0x158: 00, 00, 00, 00, 00, 00, 02, 00, 03, 00, 04, 00,

The total sum at 0x171: 09

****

# Example for Sorting

## **Program:**

;bubble sorting

[org 0x0100]

jmp start

start:

mov bx,0

mov byte[swap],0

compare:

mov ax, [vuid + bx]

cmp ax, [vuid + bx + 2]

jle noswap

;swaping

mov dx, [vuid + bx + 2]

mov [vuid + bx + 2], ax

mov [vuid + bx], dx

mov byte[swap],1

noswap:

add bx, 2

cmp bx, [len]

jne compare

cmp byte[swap],1

je start

mov ax, 0x4c00

int 0x21

;MC200400003

; 0, 2, 4, 6, 8, 10, 12, 14, 16, 18

vuid: dw 3, 8, 1, 9, 12, 5, 13, 2, 6, 7

len: dw 18; (length of VUID \*2 - 1)

swap: db 0

## **Listing:**

1 ;bubble sorting

2

3 [org 0x0100]

4 00000000 E90000 jmp start

5 start:

6 00000003 BB0000 mov bx,0

7 00000006 C606[5200]00 mov byte[swap],0

8

9 compare:

10 0000000B 8B87[3C00] mov ax, [vuid + bx]

11 0000000F 3B87[3E00] cmp ax, [vuid + bx + 2]

12 00000013 7E11 jle noswap

13

14 ;swaping

15 00000015 8B97[3E00] mov dx, [vuid + bx + 2]

16 00000019 8987[3E00] mov [vuid + bx + 2], ax

17 0000001D 8997[3C00] mov [vuid + bx], dx

18 00000021 C606[5200]01 mov byte[swap],1

19

20 noswap:

21 00000026 81C30200 add bx, 2

22 0000002A 3B1E[5000] cmp bx, [len]

23 0000002E 75DB jne compare

24

25 00000030 803E[5200]01 cmp byte[swap],1

26 00000035 74CC je start

27

28 00000037 B8004C mov ax, 0x4c00

29 0000003A CD21 int 0x21

30

31

32 ;MC200400003

33 ; 0, 2, 4, 6, 8, 10, 12, 14, 16, 18

34 0000003C 03000800010009000C- vuid: dw 3, 8, 1, 9, 12, 5, 13, 2, 6, 7

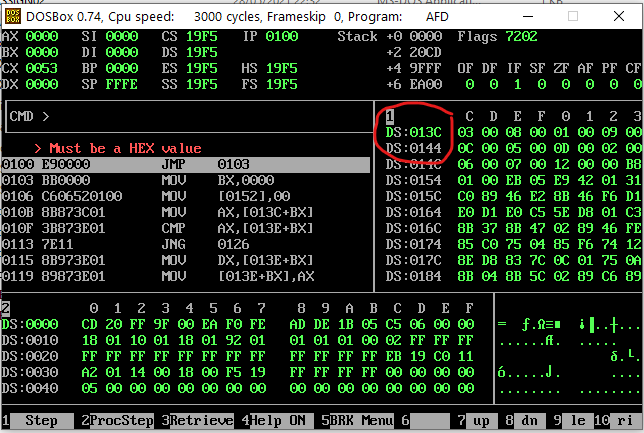
35 00000045 0005000D0002000600-

36 0000004E 0700

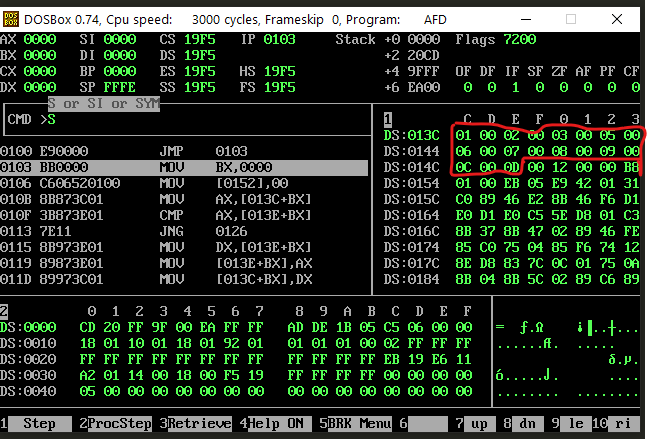
37 00000050 1200 len: dw 18; (length of VUID \*2 - 1)

38 00000052 00 swap: db 0

## **Starting; See at the location 013C**

****

## **After execution look at the 0x013C:**

****