|  |
| --- |
| FAST National University |
| **Lab 3** |
|  |

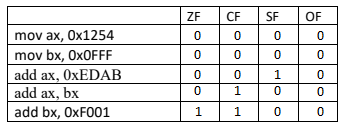
**Computer Organization and Assembly Language**

|  |  |
| --- | --- |
| **Student Name** | Muhammad Ali Butt |
| **Registration #** | 21L-7646 |
| **Instructor** | Sir Hazoor Ahmad |
| **Class** | Lab-CS2A |
| **Section** | BCS-3A2 |
| **Semester** | Fall 2022 |

Fast School of Computing

FAST-NU, Lahore, Pakistan

# Activity 1



# Activity 2

## **Assembly Language Code**

[org 0x0100]

jmp start

num: db 20

start:

mov al, [num]

mov cl, al

mov dl, 0

multiply:

add dx, ax

sub cx, 1

jnz multiply

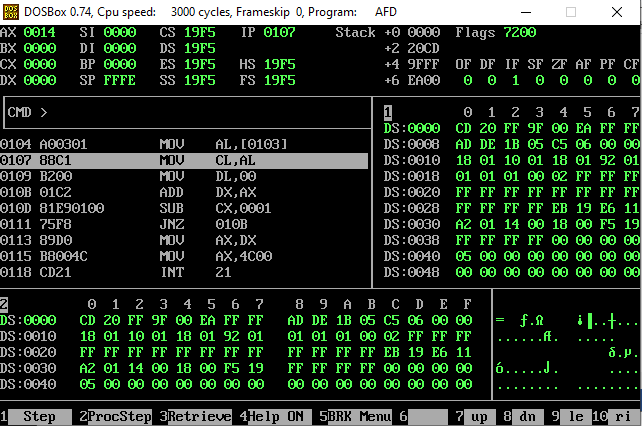
mov ax, dx

mov ax, 0x4c00

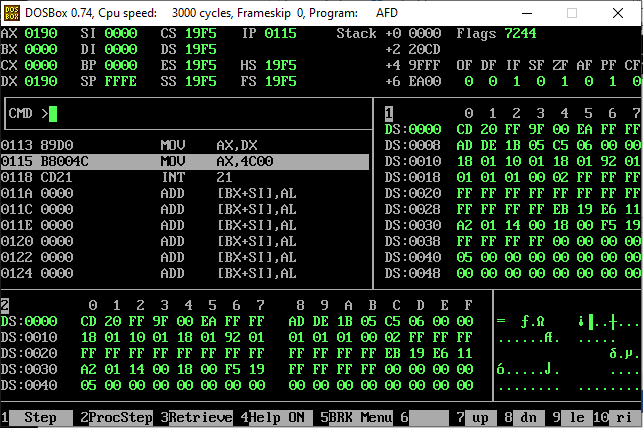
int 0x21

## **Debugging Screenshots**

After adding the number (20) in ax:



Final Result:



# Activity 3

## **Assembly Language Code**

[org 0x0100]

jmp start

array: dw 1, 9, 9, 9, 8, 8, 8, 8, 8, 8, 1, 1, 9, 9, 8, 8, 8, 8, 1, 9, 8, 8

key: dw 9

start:

mov ax, [key]

mov bx, 0

mov dx, 0 ;Initializing dx to store the frequency

compare:

cmp [array+bx], ax

je increment

add bx, 2

cmp bx, 44

je end

jne compare

increment:

add bx, 2

add dx, 1

jmp compare

end:

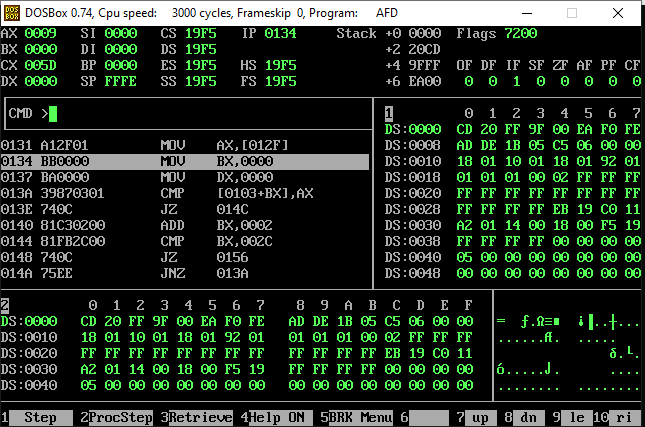
mov ax, dx ;Store result in ax

mov ax, 0x4c00

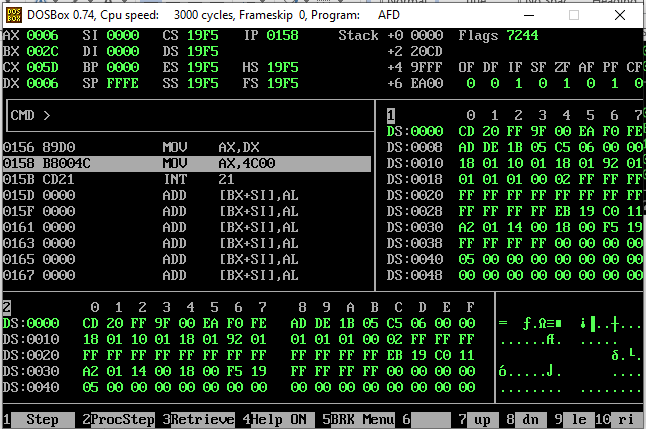
int 0x21

## **Debugging Screenshots**

Key (9) stored in ax:



Result stored in ax:



# Activity 4

## **Assembly Language Code**

[org 0x0100]

jmp start

;Program to find num!

num: dw 5

result: dw 0

start:

mov al, [num] ;Will store the result of repeated additions

mov bx, ax ;Will be used to store the number that is to be added into ax.

mov cx, ax ;Will be used to store the number of times that bx is to be added into ax.

mov dx, ax ;Will be used to store the number of times we still need to multiply ax.

sub cx, 2

Multiply:

add ax, bx

sub cx, 1

cmp cx, 0

jne Multiply

DecrementD:

sub dx, 1

mov cx, dx

sub cx, 2

mov bx, ax

cmp dx, 2

jne Multiply

je end

end:

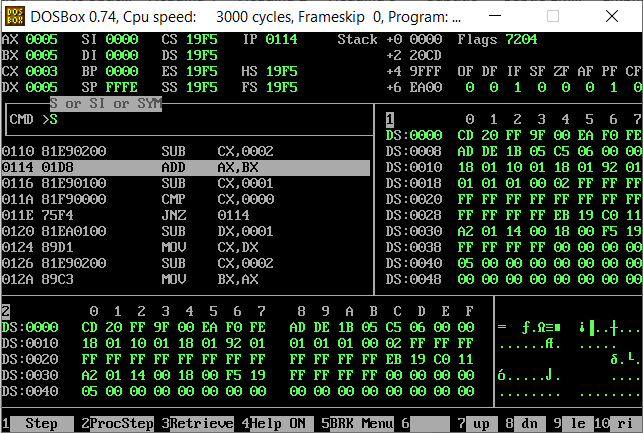
mov [result], ax ;Storing result into memory

mov ax, 0x4c00

int 0x21

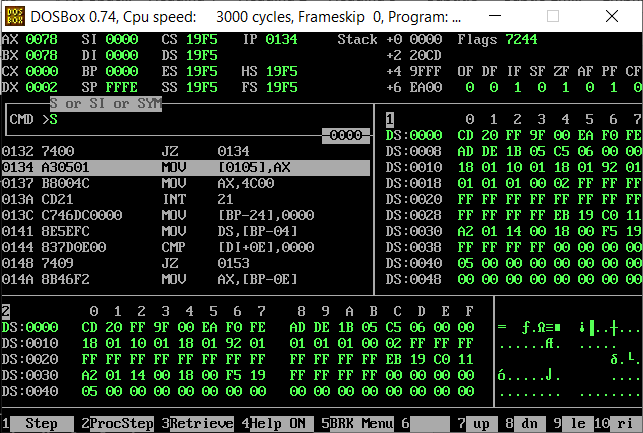
## **Debugging Screenshots**

After executing the start block:



Final Result:





# Activity 5

## **Assembly Language Code**

[org 0x0100]

jmp start

array: dw 111, 999, 888, 888, 11, 99, 88, 88, 1, 9, 8, 8

start:

mov ax, [array]

mov bx, 0

compare:

add bx, 2

cmp bx, 24

je end

cmp ax, [array+bx]

jb updateAx

jnb compare

updateAx:

mov ax, [array+bx]

jmp compare

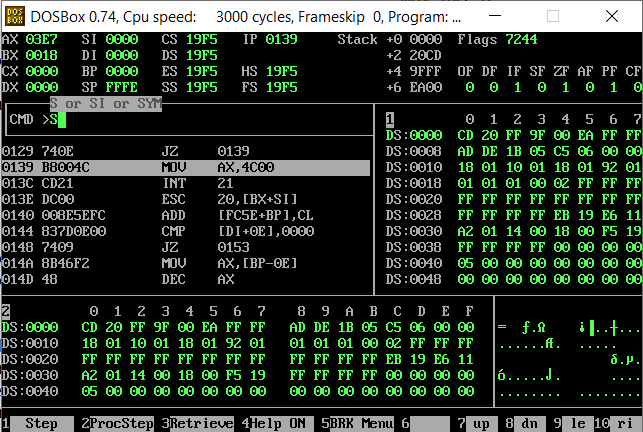
end:

mov ax, 0x4c00

int 0x21

## **Debugging Screenshots**

Final Result:



# Activity 6

## **Assembly Language Code**

Description: The program will store the largest number of the array in ax and the second largest in dx, If the whole array contains the same numbers, only then will that same number be stored in both ax and dx. The program contains various blocks of code. These blocks were used to prevent same values of ax and dx if the starting n values (where n<size) of the array were equal.

[org 0x0100]

jmp start

array: dw 111, 999, 888, 888, 11, 99, 88, 88, 1, 9, 8, 8

start:

mov ax, [array] ;Largest number will be stored in ax

mov dx, [array] ;Second largest number will be storied in dx

mov bx, 0

compare1:

add bx, 2

cmp bx, 24

je initializeDx

cmp ax, [array+bx]

jb updateAx

jnb compare1

updateAx:

mov ax, [array+bx]

jmp compare1

initializeDx: ;Get ready to find dx by resetting bx to 0

mov bx, 0

ifAxAndDxAreSame:

mov dx, [array+bx]

cmp bx, 24

je end

add bx, 2

cmp dx, ax

je storeNextValueInDx

jne compare2

storeNextValueInDx:

add bx, 2

jmp ifAxAndDxAreSame

compare2:

add bx, 2

cmp bx, 24

je end

cmp dx, [array+bx]

jb updateDx

jnb compare2

updateDx:

mov dx, [array+bx]

jmp compare2

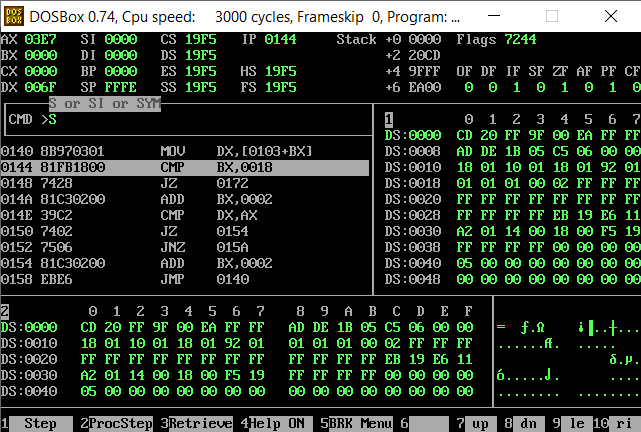
end:

mov ax, 0x4c00

int 0x21

## **Debugging Screenshots**

After finding the largest number and storing it in ax:



After finding the second largest number as well and storing it in dx:

