|  |
| --- |
| FAST National University |
| **Lab 4** |
|  |

**Computer Organization and Assembly Language**

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

Fast School of Computing

FAST-NU, Lahore, Pakistan

# Activity 1

## **Assembly Language Code**

## [org 0x100]

## jmp start

## RollNumber: dw 7646h

## OddIndices: dw 0xAAAA

## EvenIndices: dw 5555h

## start:

## mov ax, [RollNumber]

## AND [OddIndices], ax

## shr word [OddIndices], 1

## AND [EvenIndices], ax

## shl word [EvenIndices], 1

## mov ax, 0

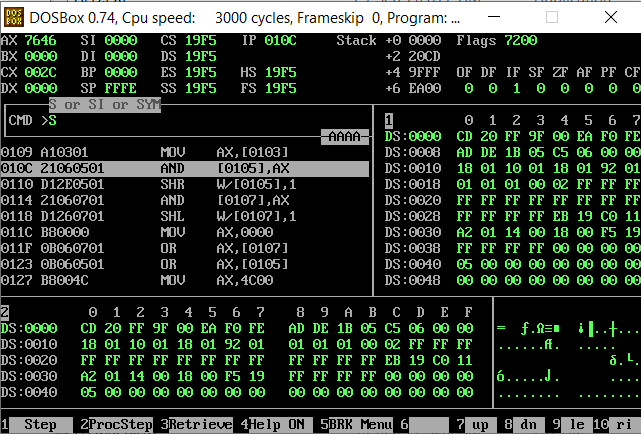
## OR ax, [EvenIndices]

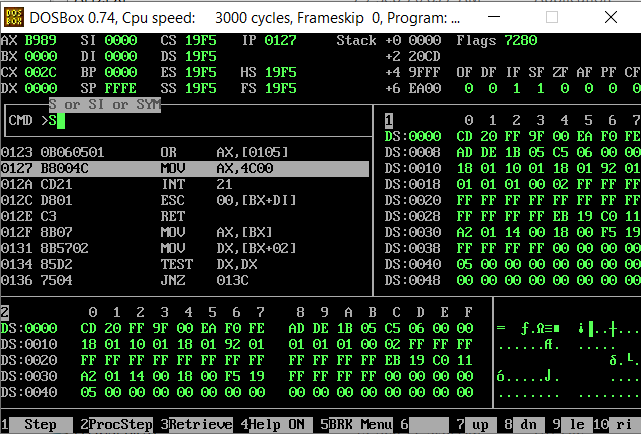
## OR ax, [OddIndices]

## mov ax, 0x4c00

## int 0x21

## **Debugging Screenshots**





# Activity 2

## **Assembly Language Code**

[org 0x100]

jmp start

RollNumber: dw 7646h

OddIndices: dw 0xCCCC

EvenIndices: dw 3333h

start:

mov ax, [RollNumber]

AND [OddIndices], ax

shr word [OddIndices], 2

AND [EvenIndices], ax

shl word [EvenIndices], 2

mov ax, 0

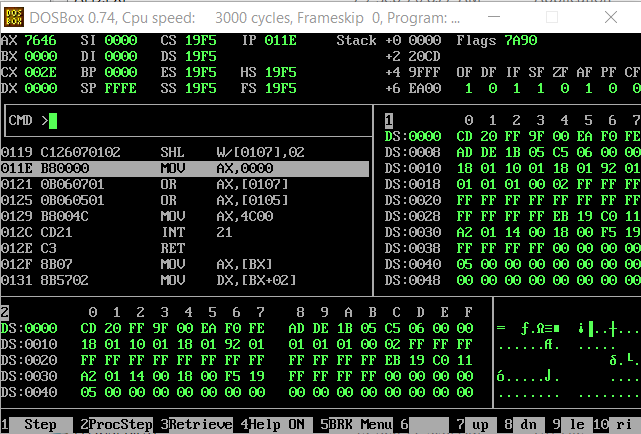
OR ax, [EvenIndices]

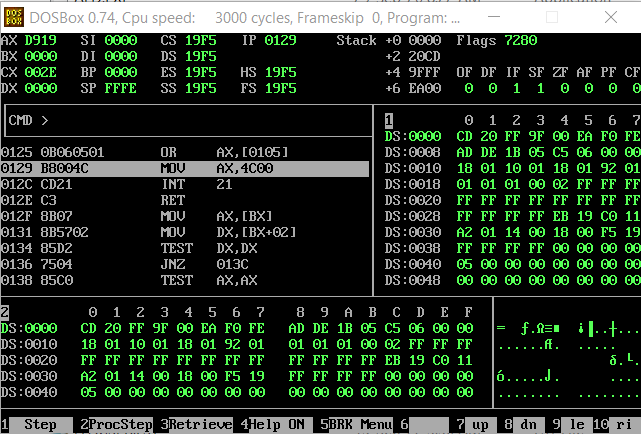
OR ax, [OddIndices]

mov ax, 0x4c00

int 0x21

## **Debugging Screenshots**





# Activity 3

## **Assembly Language Code**

[org 0x100]

jmp start

RollNumber: dw 7646h

OddIndices: dw 0xF0F0

EvenIndices: dw 0x0F0F

start:

mov ax, [RollNumber]

AND [OddIndices], ax

shr word [OddIndices], 4

AND [EvenIndices], ax

shl word [EvenIndices], 4

mov ax, 0

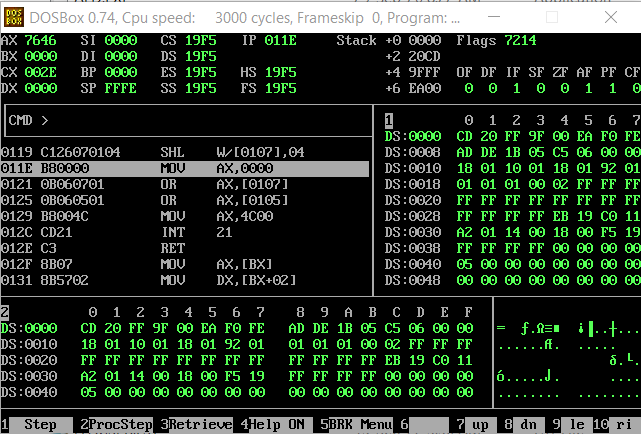
OR ax, [EvenIndices]

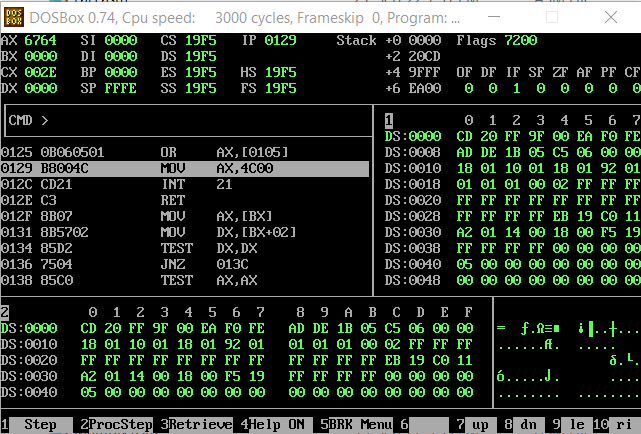
OR ax, [OddIndices]

mov ax, 0x4c00

int 0x21

## **Debugging Screenshots**





# Activity 4

## **Assembly Language Code**

## [org 0x100]

## jmp start

## f: dw 0

## a: dw 1111111111111111b

## start:

## mov ax, 7646h

## mov bx, ax

## XOR bx, [a] ;Calculating Compliment of AX

## 

## mov cx, ax

## OR cx, bx ;Calculating A||B

## mov dx, 0x1BCD

## XOR dx, ax ;Calculating (A ⊙ 0x1BCD)

## AND cx, dx

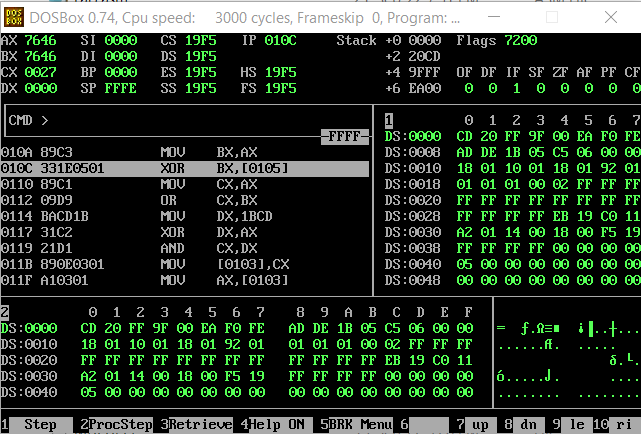
## mov [f], cx

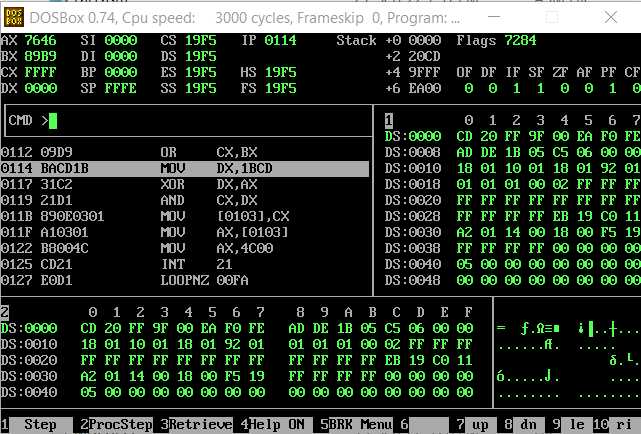
## mov ax, [f] ;Storing data in ax at the end for read access

## mov ax, 0x4c00

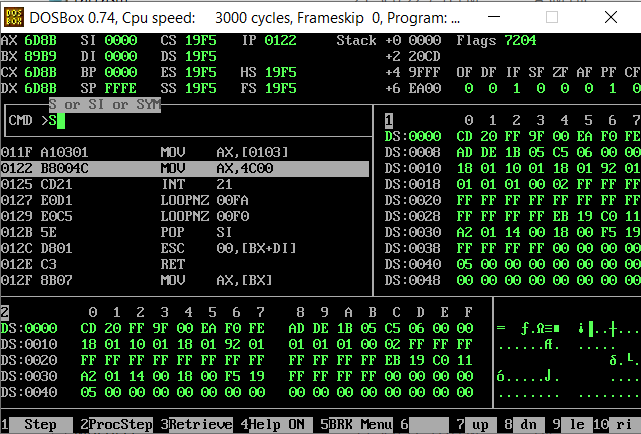
## int 0x21

## **Debugging Screenshots**





Result stored in ax and [f]:



# Activity 5

## **Assembly Language Code**

[org 0x100]

jmp start

f: dd 0

concat: dd 0

mulRes: dd 0

start:

mov ax, 7646h

mov bx, ax

not bx

mov [concat], bx

mov [concat+2], ax

mul bx

mov [mulRes], ax

mov [mulRes+2], dx

mov cx, [mulRes]

add cx, [concat]

mov [f], cx

mov cx, [mulRes+2]

adc cx, [concat+2]

mov [f+2], cx

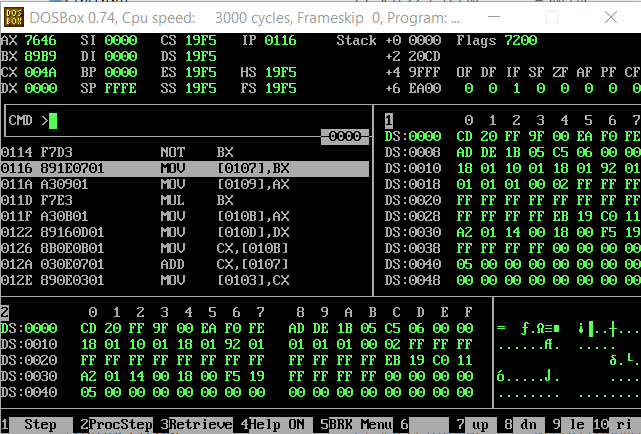
mov ax, [f] ;Storing least significant 16 bits of the result in ax for easily read access

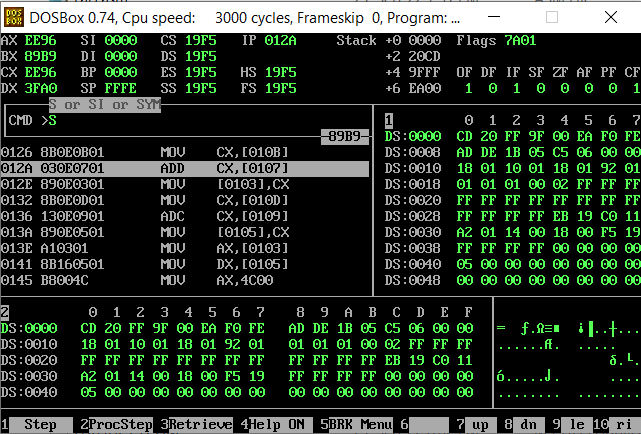
mov dx, [f+2] ;Storing most significant 16 bits of the result in dx for easily read access

mov ax, 0x4c00

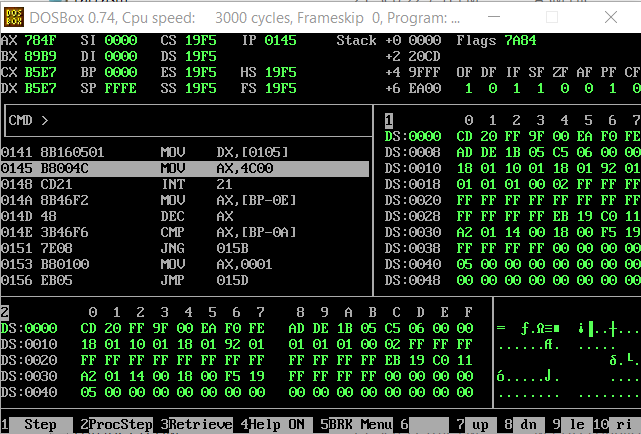
int 0x21

## **Debugging Screenshots**





Result stored in C.F. (16th bit), dx (most significant 16-bits) and ax (least significant 16-bits) i.e. B5E7 784F:



# Activity 6

## **Assembly Language Code**

Short Jump with 8-bit operand code:

[org 0x100] ;Short jump can be conditional and unconditional

mov ax, 0 ;Placeholder code

mov bx, 1

cmp ax, bx

jne end ;8-bit (2 hex digits) operand (7508 )

mov ax, 0 ;Placeholder code

mov bx, 1

jne end ;8-bit (2 hex digits) operand (7500)

end:

mov ax, 0x4c00

int 0x21

Near Jump with 16-bit operand code:

[org 0x100] ;Near jump can be unconditional only

mov ax, 0 ;Placeholder code

mov bx, 1

jmp end ;16-bit (4 hex-digits) operand (E90900)

mov ax, 0 ;Placeholder code

mov bx, 1

jmp end ;16-bit (4 hex-digits) operand (E90900)

end:

mov ax, 0x4c00

int 0x21

Far Jump with 16-bit segment and 16-bit offset address: (unmeaningful code)

[org 0x100] ;Far jumps can only be unconditional.

mov ax, 0 ;Placeholder code

mov bx, 14

cmp ax, bx

jmp [cs:bx] ;16-bit segment address + 16-bit offset address operand

mov ax, 0 ;Placeholder code

mov bx, 14

jmp [cs:bx] ;16-bit segment address + 16-bit offset address operand

end:

mov ax, 0x4c00

int 0x21