|  |
| --- |
| FAST National University |
| **Lab 5** |
|  |

**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 0x0100] **;With example 1 data**

jmp start

S1: dw -3, -1, 2, 5, 6, 8, 9 ;Starting address should be stored in si, size in cx

S2: dw -2, 2, 6, 7, 9 ;Starting address should be stored in di, size in dx

S3: dw 0, 0, 0, 0, 0, 0, 0

index: dw 0

difference:

shl cx, 1

shl dx, 1

mov ax, -2

outerl:

cmp ax, cx

je diffEnd

add ax, 2

mov bx, ax

mov bp, [bx + si]

mov bx, 0

innerl:

cmp bx, dx

je addNewVal

cmp bp, [di + bx]

je outerl

add bx, 2

jmp innerl

addNewVal:

push bx

mov bx, [index]

mov [S3+bx], bp

add bx, 2

mov [index], bx

pop bx

jmp outerl

diffEnd:

ret

start:

mov si, S1

mov cx, 7

mov di, S2

mov dx, 5

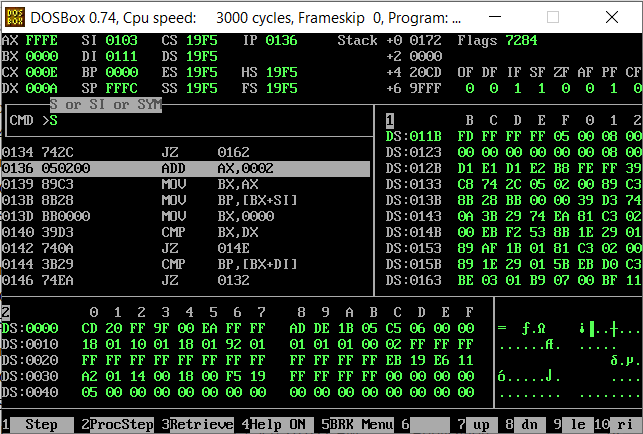
call difference

mov ax, 0x4c00

int 0x21

## **Debugging Screenshots**

Result: Set3 / ResultSet starts from memory address 0x11B in data segment.



# Activity 2

## **Assembly Language Code**

[org 0x0100]

jmp start

ones:

push bx ;Used to run the loop

push cx ;Used to count the number of 1s

push dx

mov bx, 0

mov cx, 0

mov dx, 1

l1:

cmp bx, 16

je onesEnd

inc bx

test dx, ax

jnz inccx

shl dx, 1

jmp l1

inccx:

inc cx

shl dx, 1

jmp l1

onesEnd:

mov ax, cx

pop dx

pop cx

pop bx

ret

start:

mov ax, 0x7646

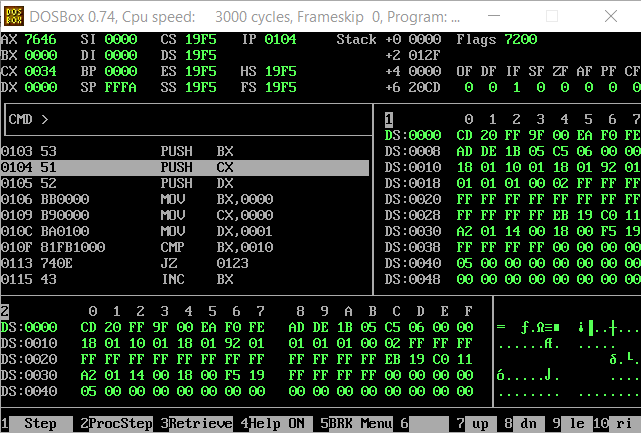
call ones

mov ax, 0x4c00

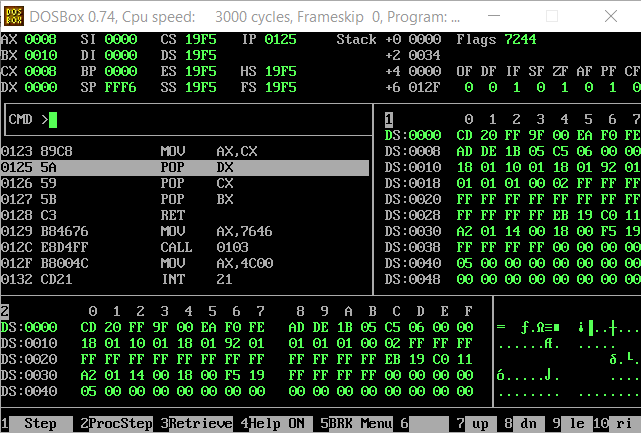
int 0x21

## **Debugging Screenshots**

Storing my roll number:



Result in ax:



# Activity 3

## **Assembly Language Code**

[org 0x0100]

jmp start

result: dw 0

ones:

push bx ;Used to run the loop

push cx ;Used to count the number of 1s

push dx

mov bx, 0

mov cx, 0

mov dx, 1

l1:

cmp bx, 16

je onesEnd

inc bx

test dx, ax

jnz inccx

shl dx, 1

jmp l1

inccx:

inc cx

shl dx, 1

jmp l1

onesEnd:

mov ax, cx

pop dx

pop cx

pop bx

ret

start:

mov ax, 0x7646

call ones

;Al-Khwarizmi Pyramid

add ax, 5 ;Value of s is now stored in ax

mov dx, 0 ;To store the sum

mov cx, 0 ;Used to run the outer loop based on size

mov si, ax

inc si

outerl:

inc cx

cmp cx, si

je end

mov bx, 1 ;Used to run the inner loop

innerloop:

cmp bx, cx

je addOnce

jne addTwice

addOnce:

add dx, bx

jmp outerl

addTwice:

add dx, bx

add dx, bx

inc bx

jmp innerloop

end:

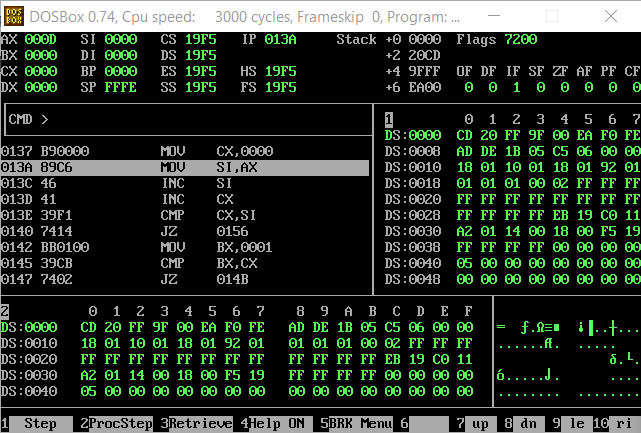
mov [result], dx ;Move the final sum to [result]

mov ax, 0x4c00

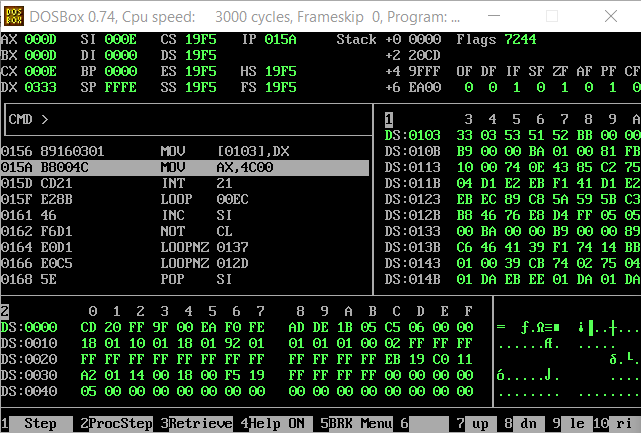
int 0x21

## **Debugging Screenshots**

Value of s stored in ax:



Final sum stored in dx and memory location (word sized) 0x0103 in data segment:



# Activity 4

## **Assembly Language Code**

[org 0x0100]

jmp start

;Memory locations that will be used for multiply subroutine

multiplicand: dd 0

multiplier: dw 0

result: dd 0

;Memory locations that will be used for addition subroutine

num1: dd 0

num2: dd 0

addResult: dd 0

;Memory location to store the final result

f: dd 0

;Multiplication subroutine

multiply:

push ax

l1:

cmp word [multiplier], 0

je mulEnd

shr word [multiplier], 1

jnc nextIter

mov ax, [multiplicand]

add [result], ax

mov ax, [multiplicand + 2]

adc [result + 2], ax

nextIter:

shl word [multiplicand], 1

rcl word [multiplicand+2], 1

jmp l1

mulEnd:

pop ax

ret

;Addition subroutine

addition:

push ax

;Copying [num1] into [result]

mov ax, [num1]

mov [addResult], ax

mov ax, [num1+2]

mov [addResult+2], ax

mov ax, [num2]

add [addResult], ax

mov ax, [num2+2]

adc [addResult+2], ax

pop ax

ret

start:

mov ax, 0x7646

mov bx, ax

not bx

mov [multiplicand], ax

mov [multiplier], bx

call multiply ;Should store A x B in result

mov [num1], bx ;Initiliazing num1 for addition

mov [num1+2], ax

mov bx, [result] ;Initializing num2 for addition

mov ax, [result+2]

mov [num2], bx

mov [num2+2], ax

call addition

mov ax, [addResult] ;Storing the final result in f

mov [f], ax

mov ax, [addResult+2]

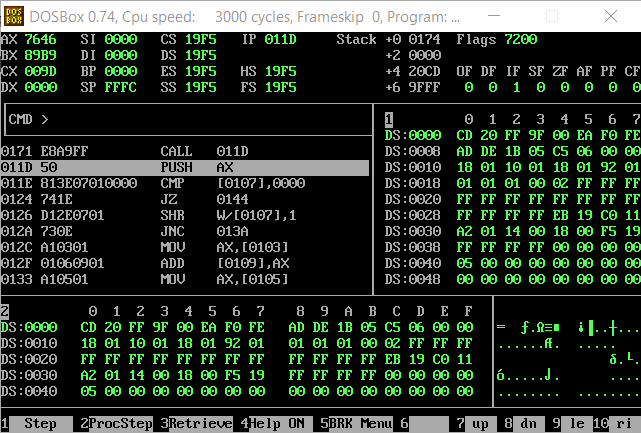
mov [f+2], ax

mov ax, 0x4C00

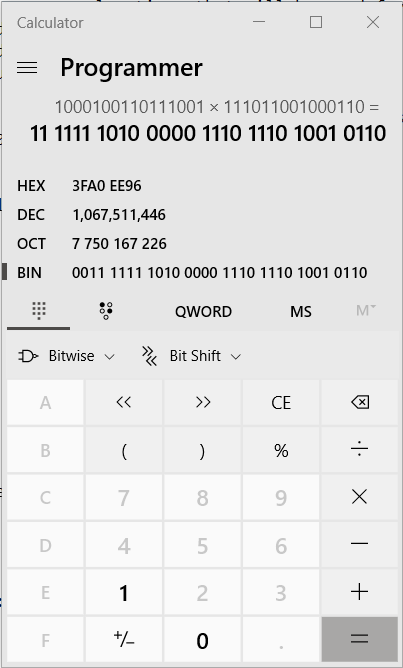
int 0x21

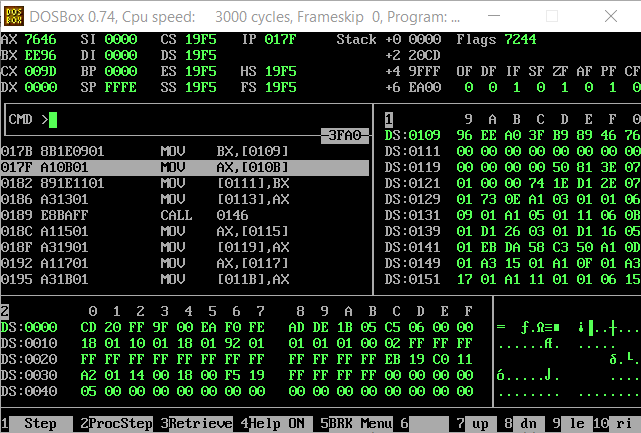
## **Debugging Screenshots**

Roll number stored in ax and NOT of ax stored in bx:

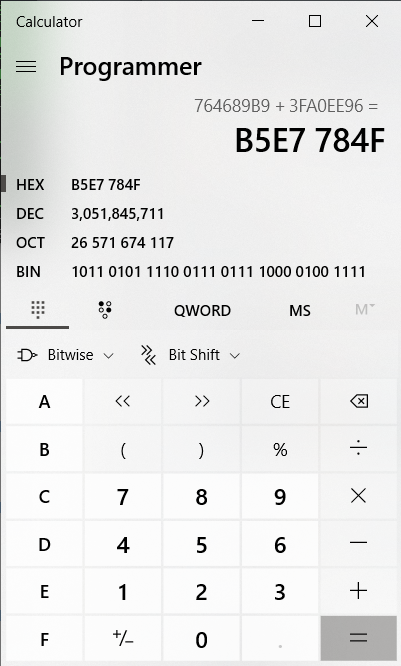


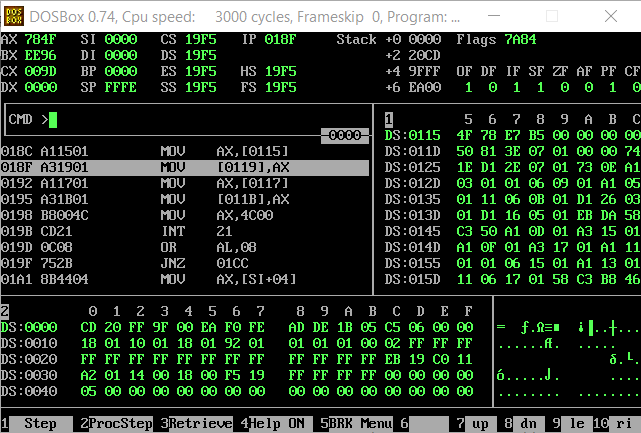
Result of multiplication stored in memory location 0x0109 (double word sized):





Result of addition stored in memory location 0x0115 in data segment:





Final result stored/copied in f (memory location 0x119) in data segment:

