**SSN College of Engineering**

**Department of Computer Science and Engineering**

**UCS1512 – Microprocessors Lab**

**Model Practical**

**Name : Kshitij Sharma Date: 09/11/2020 Reg No: 185001080**

**Question 1:**

**Aim:** Write a ALP using 8086 to perform String Manupliation

**Algorithm: Moving a String of bytes**

* Move the data segment to the AX register and then move it to the DS register.
* Move the extra segment to the AX register and then move it to the ES register.
* Move the count value to CX register.
* Move the offset of str1 and str2 to SI and DI respectively.
* Clear the Direction Flag.
* Now use rep movsb instruction to move the number of bytes specified by count from str1 to str2.This rep movsb instruction moves data from [SI] to [DI](Offsets are specified by SI and DI while Base addresses are stored in DS and ES respectively) in each instruction while incrementing SI and DI by 1 and decrementing CX by 1 and stops when CX=0.

**Program:**

mov cx,count

mov si, offset str1

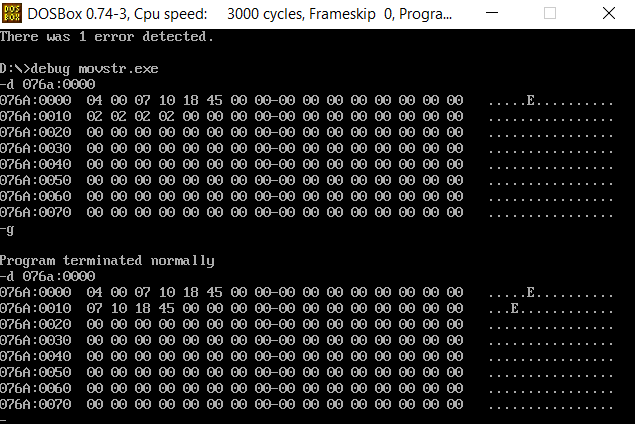
mov di, offset str2

cld

rep movsb

mov ah,4ch

int 21h



**Algorithm: comparing two strings**

* Move the data segment to the AX register and then move it to the DS register.
* Move the extra segment to the AX register and then move it to the ES register.
* Move the COUNT value to CX register.
* Move the offset of str1 and str2 to SI and DI respectively.
* Clear the Direction Flag.
* Now use repe cmpsb instruction to compare the individual bytes of the two strings. When a mismatch occurs or if CX becomes zero, repe cmpsb stops. In each iteration, CX is decremented by 1 while SI and DI are incremented by 1 as DF=0.
* Move COUNT to AX and CX to BX.
* When CX is not zero it implies mismatch has occurred. Perform 16 bit subtraction of AX and BX and then move AX to STATUS.
* If CX=0,there are two possibilities. Either mismatch occurred in last byte or both the strings are equal. Now Set the direction flag using STD and again use repe cmpsb. If CX is zero move 0000h to AX and then MOV AX to STATUS. Else, move COUNT to AX and then AX to STATUS.

**Program:**

mov cx,count

mov si, offset str1

mov di, offset str2

cld

repe cmpsb

mov ax,count

mov bx,cx

mov cx,0000h

cmp bx,0000h

jz here1

sub ax,bx

jnc here

neg ax

inc cx

here:

mov status, ax

mov ah,4ch

int 21h

here1:

std

mov cx,count

repe cmpsb

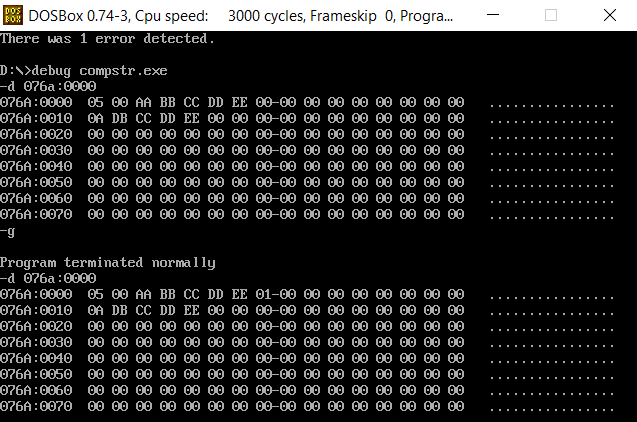
cmp cx,0000h

mov ax,0000h

jz here

mov ax,count

jmp here



**Algorithm : Searching for a byte in a String**

* Move the data segment to the AX register and then move it to the DS register.
* Move the extra segment to the AX register and then move it to the ES register.
* Move the COUNT value to CX register.
* Move the STR1 to AL register and offset of STR2 to DI.
* Clear the Direction Flag.
* Now use repne scasb instruction to compare STR1 and the bytes of STR. When a matching byte is found or if CX becomes zero, repne scasb stops. In each iteration, CX is decremented by 1 while DI is incremented by 1 as DF=0.
* Move COUNT to DX and CX to BX.
* When CX is not zero it implies that a matching byte has been found. Perform 16 bit subtraction of DX and BX and then move DX to STATUS.
* If CX=0,there are two possibilities. Either last byte is the matching byte or there are no matching bytes. Now Set the direction flag using STD and again use repne scasb. If CX is zero move 0000h to DX and then MOV DX to STATUS. Else, move COUNT to DX and then DX to STATUS

**Program:**

mov cx,count

mov al,str1

mov di, offset str

cld

repne scasb

mov dx,count

mov bx,cx

mov cx,0000h

cmp bx,0000h

jz here1

sub dx,bx

jnc here

neg dx

inc dx

here:

mov status, dx

mov ah,4ch

int 21h

here1:

std

mov cx,count

repne scasb

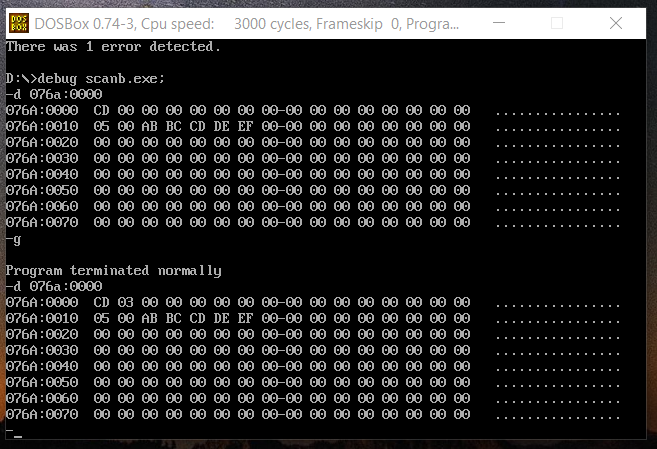
cmp cx,0000h

mov dx,0000h

jz here

mov dx,count

jmp here



**Result:**

An ALP to perform String manupliation like comparing moving and searching**.**

**Question2:**

**Aim:** **Write an ALP using 8086to Fibonacci series operation.**

**Algorithm:**

* Move the data segment into the register
* Store ax in ds register
* Store n in cx where n is the length of the series
* Clear al
* Store the offset of fib into si
* Store al in si
* Since the first 2 elements of the sequence are initilized, we need to decrement the counter by 2
* Moves the element in the (i-1)th position into AL
* Moves the (i)th element with the (i-1)th element already present in AL
* Increment SI to point to the next position
* Store the sum in the new position
* The instructions between label L1 and this LOOP instruction are executed “CX” times

**Program:**

mov AX, data

mov DS, AX

mov cx, n

mov al, 00h

mov SI, offset fib

mov [si], al

inc si

inc al

mov [si],al

sub cx,0002h

l1: mov al,[si-1]

add al,[si]

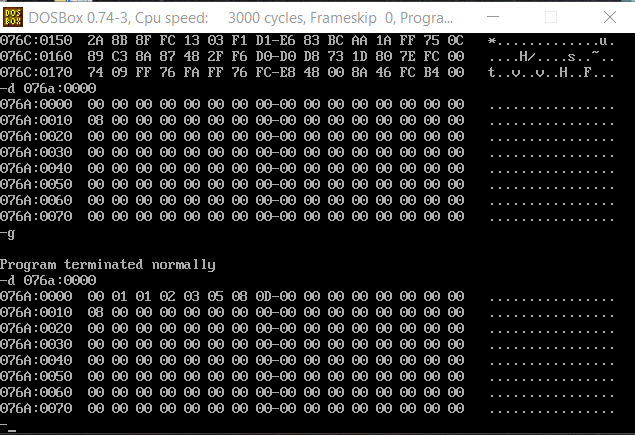
inc si

mov [si],al

loop l1

MOV AH, 4ch

INT 21h



**Result:**

An ALP for constructing a Fibonacci series was implemented.