# LAB 11: STRING MANIPULATION

**OBJECTIVES:**

To perform string operations using string instructions:

Comparison of strings, reversing of string, Searching of string

**Pre-Lab:** Task

**CMPS / CMPSB / CMPSW (COMPARE STRING BYTES OR STRING WORDS)**

This instruction can be used to compare a byte / word in one string with a byte / word in another string. SI is used to hold the offset of the byte or word in the source string, and DI is used to hold the offset of the byte or word in the destination string.

The **AF, CF, OF, PF, SF, and ZF** flags are affected by the comparison, but the two operands are not affected. After the comparison, SI and DI will automatically be incremented or decremented to point to the next or previous element in the two strings. If DF is set, then SI and DI will automatically be decremented by 1 for a byte string and by 2 for a word string. If DF is reset, then SI and DI will automatically be incremented by 1 for byte strings and by 2 for word strings. The string pointed to by SI must be in the data segment. The string pointed to by DI must be in the extra segment.

The CMPS instruction can be used with a REPE or REPNE prefix to compare all the elements of a string.

MOV SI, OFFSET FIRST Point SI to source string

MOV DI, OFFSET SECOND Point DI to destination string

CLD DF cleared,

SI and DI will auto-increment after compare

MOV CX, 100 put number of string elements in CX

REPE CMPSB Repeat the comparison of string bytes until end of string or until compared bytes is not equal. CX functions as a counter, which the REPE prefix will cause CX to be decremented after each compare.

**In-Lab Task:**

**Q. Define two string arrays of equal size ‘Allah-O-Akbar’**

**Move first array at offset address starting at 1120h in DS (DF=0)**

**Move second array at offset address starting at 2310h in ES (DF=0).**

**Compare these 2 strings using repeat instruction; Comment about CX register.**

**Code:**

.model small

.stack 100h

.data

; Define the string arrays

string1 db 'Allah-O-Akbar$'

string2 db 'Allah-O-Akbar$'

.code

main:

; Set up data segment (DS) and extra segment (ES) registers

mov ax, @data

mov ds, ax

mov es, ax

; Move the first string to DS:1120h

lea si, string1 ; Load address of string1 into SI

mov di, 1120h ; Destination offset address in DS

call MoveString ; Call subroutine to move the string

; Move the second string to ES:2310h

lea si, string2 ; Load address of string2 into SI

mov di, 2310h ; Destination offset address in ES

call MoveString ; Call subroutine to move the string

; Compare the two strings

mov cx, 11 ; Set CX to the length of the strings (excluding the null terminator)

lea si, [1120h] ; Load address of the first string into SI

lea di, [2310h] ; Load address of the second string into DI

cld ; Clear Direction Flag (to move forward through the strings)

repe cmpsb ; Compare strings byte by byte until CX becomes zero or a mismatch is found

jz stringsEqual ; If strings are equal, jump to stringsEqual

; Strings are not equal

mov ah, 4Ch ; Exit program with return code 0

int 21h

stringsEqual:

; Strings are equal

mov ah, 4Ch ; Exit program with return code 0

int 21h

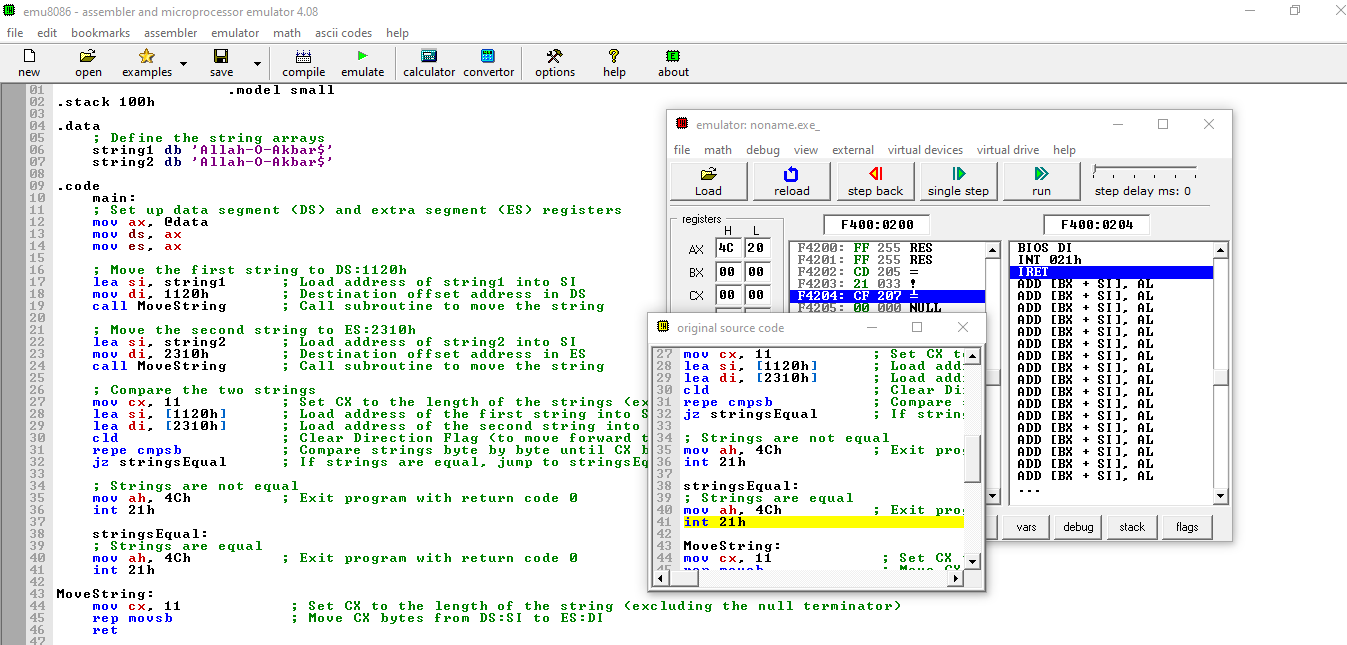
MoveString:

mov cx, 11 ; Set CX to the length of the string (excluding the null terminator)

rep movsb ; Move CX bytes from DS:SI to ES:DI

ret

end main



**Program stops on Strings are equal Label.**

**Q. Change one and middle letter of second string to another letter then use compare CMPS using REPE. Comment about CX.**

**Code:**

.model small

.stack 100h

.data

; Define the string arrays

string1 db 'Hello$'

string2 db 'Hellz$'

.code

main:

; Set up data segment (DS) and extra segment (ES) registers

mov ax, @data

mov ds, ax

mov es, ax

; Modify the middle letter of the second string

mov si, offset string2 ; Load address of string2 into SI

add si, 2 ; Move SI to the middle of the string

mov byte ptr [si], 'o' ; Change the middle letter to 'o'

; Compare the two strings

mov cx, 5 ; Set CX to the length of the strings (excluding the null terminator)

lea si, string1 ; Load address of the first string into SI

lea di, string2 ; Load address of the second string into DI

cld ; Clear Direction Flag (to move forward through the strings)

repe cmpsb ; Compare strings byte by byte until CX becomes zero or a mismatch is found

; Check if strings are equal

jz stringsEqual ; If strings are equal, jump to stringsEqual

; Strings are not equal

mov ah, 4Ch ; Exit program with return code 0

int 21h

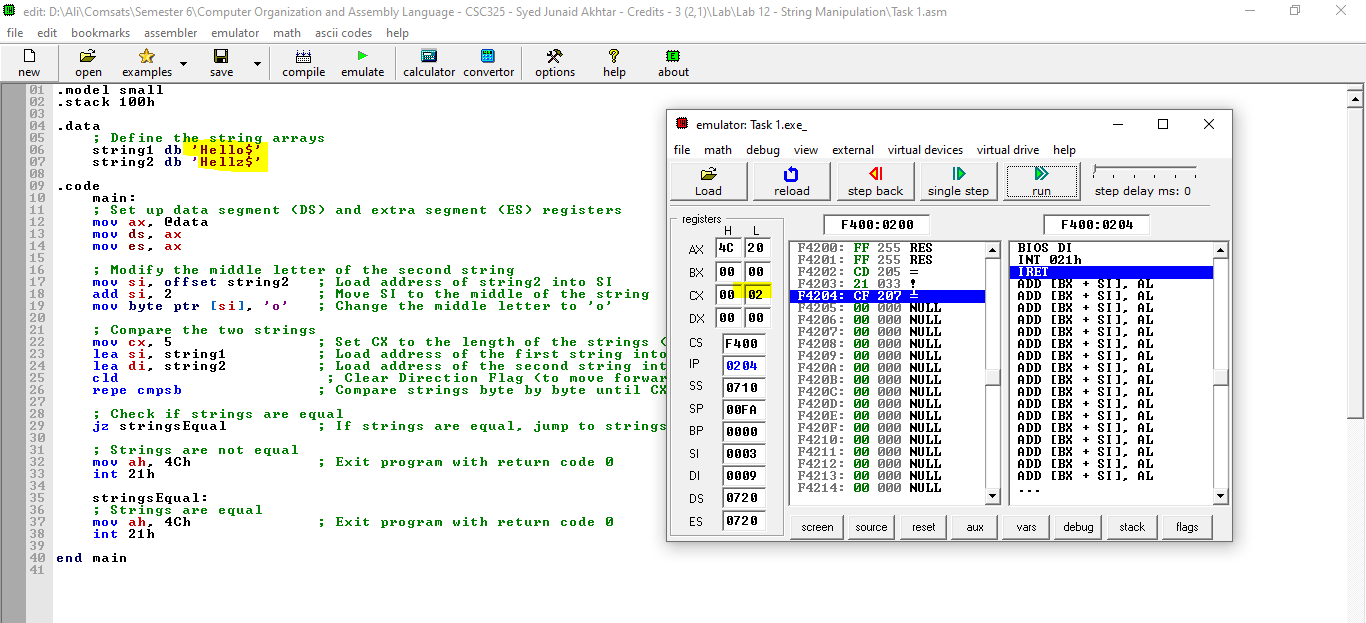
stringsEqual:

; Strings are equal

mov ah, 4Ch ; Exit program with return code 0

int 21h

end main



**After the comparison, CX will contain the number of bytes remaining to be compared if the comparison ended due to a mismatch.**

**If the comparison ended because the end of either string was reached (without finding a mismatch), CX will be 0.**

Scan ‘A’ in string array1 and array2. Calculate the count

Scan ‘a’ in the array1 and array2 and calculate the count.

Program to find the size of the string.

Reverse the given string

Program to count the vowels in the string

Change the case of string

Commands **MOVSB, STOSB, SCASB, LODSB**

**Post-Lab** Assignment

If marks of 4 students in five subjects are as follows

**A = 45, 66, 34, 55, 46**

**B = 65, 68, 74, 85, 60**

**C = 71, 74, 91, 83, 78**

**D = 66, 65, 72, 75, 61**

Calculate average marks subject wise.

**Critical Analysis /Conclusion**

|  |
| --- |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Performance (15Marks)** | | **Viva (5 Marks)** | **Total/20** |
| **Pre-Lab Exercise** | **/3** |  |  |
| **Performance** | **/4** |  |  |
| **Results** | **/3** |
| **Critical Analysis** | **/2** |
| **Post Lab Exercise** | **/3** |
| **Comments** | | | |