

Experiment 05 : String Operations Part-I

Learning Objective: Student should be able to apply string operations (i) Accept, (ii) Display, (iii) Concatenate and (iv) Compare in ALP.

Tools: TASM/MASM

Theory:

String Instruction:

- String is the series of bytes stored sequentially in the memory, string Instructions operate such “Strings”.
- The source elements is taken from the data segment using the SI register.
- The destination element is in the extra segment pointed by the DI register.
- SI and / or DI are incremented/ decremented after each operation depending upon the direction flag “DF” in the flag register.

1. MOVS: MOVSB/ MOVSW (move string):

- It is used to transfer a word/ byte from data segment to extra segment.
- The offset of the source in data segment is in SI.
- The offset of the destination in extra segment is in DI.
- SI and DI are incremented / decremented after the transfer depending upon the
 - direction flag.
- Eg: MOVSB ;ES:[DI]← DS:[SI]... byte transfer

• ;SI← SI + OR – 1 ... depending upon DF

;DI← DI + OR – 1 ... depending upon DF

MOVSW ;{ES:[DI], ES:[DI + 1]}← {DS:[SI], DS:[SI + 1]}...word

;SI← SI + or – 2

;DI← DI + or – 2

2. LODS: LODSB/LODSW (load string)

- It is used to load AL (or AX) register with a byte (or word) from data segment.

- The offset of the source in data segment is in SI.
- SI is incremented/ decremented after the transfer depending upon the direction flag (DF).
- Eg: LODSB ;AL \leftarrow DS:[SI]... byte transfer ;SI \leftarrow SI+ or – 1 ... depending upon DF
LODSW;AL \leftarrow DS:[SI];AH \leftarrow DS:[SI + 1].. word transfer ;SI \leftarrow SI + or – 2

3. STOS: STOSB/ STOSW (store string):

- It is used to store AL (or AX) into a byte (or word) in the extra segment.
- The offset of the source in extra segment is in DI.
- DI is incremented/ decremented after the transfer depending upon the direction flag (DF).
- Eg: STOSB ;ES:[DI] \leftarrow AL... byte transfer ;DI \leftarrow DI+ or – 1 ... depending upon DF
STOSW ;ES:[DI] \leftarrow AL; ES:[DI + 1] \leftarrow AH ... word transfer ;DI \leftarrow DI + or – 2 ...
depending upon DF

4.CMPS: CMPSB/ CMPSW (compare string):

- It is used to compare a byte (or word) in the data segment with a byte (or word) in the extra segment.
 - The offset of the byte (or word) in data segment is in SI.
 - The offset of the byte (or word) in extra segment is in DI.
 - SI and DI are incremented/ decremented after the transfer depending upon the direction flag.
 - Comparison is done by subtracting the byte (or word) from extra segment from the byte (or word) from data segment. The flag bits are affected, but the result is not stored anywhere.
- Eg: CMPSB ;Compare DS:[SI] with ES:[DI] ... byte operation
;SI \leftarrow SI + or – 1 ... depending upon DF
;DI \leftarrow DI + or – 1 ... depending upon DF
CMPSW ;Compare {DS:[SI], DS:[SI + 1]} with {ES:[DI], ES:[DI + 1]}
;SI \leftarrow SI + or – 2 ... depending upon DF
;DI \leftarrow DI + or – 2 ... depending upon DF

5.SCAS: SCASB/ SCASW (STRING COMPARE ACCUMULATOR)

- It is used to compare the contents of AL (or AX) with a byte (or word) in the extra segment.
- The offset of the byte (or word) in extra segment is in DI.
- DI is incremented/ decremented after the transfer depending upon the direction flag (DF).
- Comparison is done by subtracting the byte (or word) from extra segment from AL (or AX).
- The flag bits are affected, but the result is not stored anywhere.
- Eg: SCASB;Compare AL with ES:[DI] ... byte operation.
;DI ← DI + or – 1 ... depending upon DF.
SCASW ;Compare {AX} with {ES:[DI], ES:[DI + 1]} ... word operation
;DI ← DI + or – 1 ... depending upon DF.

Explanation :

Using Macro display the Menu for entering string, calculate length, concatenate, compare and exit. Accept the choice from user using INT 21H function 01H.

If choice = 1, call procedure for accepting string. Using interrupt INT 21H, function 0AH accept the string and end procedure. Return back to display Menu.

If choice = 2, call procedure for finding length of the string. Display length and return back to display Menu. (Repeat the process to enter second string)

If choice = 3, call procedure to concatenate two strings. Display the concatenation and return back to display Menu.

If choice = 4, call procedure to compare the two strings. If two strings are equal display, the strings are equal, otherwise display Strings are not equal.

If choice = 5, terminate the program. If any other key is pressed display invalid choice.

5. Procedure/Algorithm:

Algorithm:

Step I: Initialize the data and stack memory.

Step II: Using Macro display Menu.

1. Accept 2. Length 3. Concatenate 4. Compare 5. Exit.

Step III: Accept choice from user using INT 21H, function 01H.

Step IV: IS choice = 1 jump to step XI else goto step V.

Step V: IS choice = 2 jump to step XIV else goto step VI.

Step VI: IS choice = 3 jump to step XVII else goto step VII.

Step VII: IS choice = 4 jump to step XX else goto step VIII.

Step VIII: IS choice = 5 jump to step XXIII else goto step IX.

Step IX: Display Wrong choice.

Step X: Jump to step II.

Step XI: Call procedure accept.

Step XII: Accept string using INT 21H, function 0AH.

Step XIII: Return to main program and goto step II.

Step XIV: Call procedure length.

Step XV: Calculate the length of string and display it using INT 21H, function 02H.

Step XVI: Return back to main program and jump to step II.

Step XVII: Call procedure concatenate.

Step XVIII: Concatenate two strings and display.

Step XIX: Return back to main program and jump to step II.

Step XX: Call procedure compare.

Step XXI: Check if two strings are equal. If yes display strings are equal else strings are not equal.

Step XXII: Return back to main program and jump to step II.

Step XXIII: Terminate the program and stop.

Concatenation of two strings

Program Statement:

Write a program in the assembly language of 8086, to concatenate two strings.

Explanation:

Firstly, we will accept the two strings to be concatenated. Then, we will call procedure CONCAT which will concatenate the two strings. Display the concatenated strings.

Algorithm :

Step I : Start.

Step II : Accept string 1 from user.

Step III : Accept string 2 from user.

Step IV : Call procedure CONCAT.

Step V : Load length of string 1 in CX.

Step VI : Load the address of source string 1 in SI and DI.

Step VII : Copy the contents of string 1 to destination string.

Step VIII : Load SI with address of string 2.

Step IX : Copy the contents of string 2 to destination string.

Step X : Display the concatenated string.

Step XI : Procedure and return to calling program

Step XII : Stop

Compare two strings

Program statement :

Write a program in the ALP of 8086 to check the data in two strings are equal, if equal display the message “equal strings”, and if not display the message “unequal strings”.

Explanation :

We will accept two strings from the user. After accepting the strings, the first step in string comparison is to check whether their string lengths are equal. If the string lengths are not equal, we print the message unequal strings. If the string lengths are equal, we check if the contents of two strings are equal. The lengths of the two strings are initialized in the CX register.

The source and destination address are initialized in DS : SI and ES : DI registers.

Using the string instruction REPE CMPSB, two data are compared character by character.

If all the characters are matching display the message “equal strings” otherwise display “unequal strings”.

Algorithm :

Step I : Initialize the data memory.

Step II : Allocate data memory to save the strings.

Step III : Initialize DS and ES register.

Step IV : Accept the first string.

Step V : Accept the second string.

Step VI : Load the number of characters of first string in CL.

Step VII : Load the number of characters of second string in CH register.

Step VIII : Compare the lengths of the two strings. If not go to step XIII.

Step IX : Load number of characters to be compared in CX. Microprocessor & Interfacing Techniques (PU) L-24 Lab Manual

Step X : Compare the strings, character by character. If not same go to step XIII.

Step XI : Print “equal strings” using Macro.

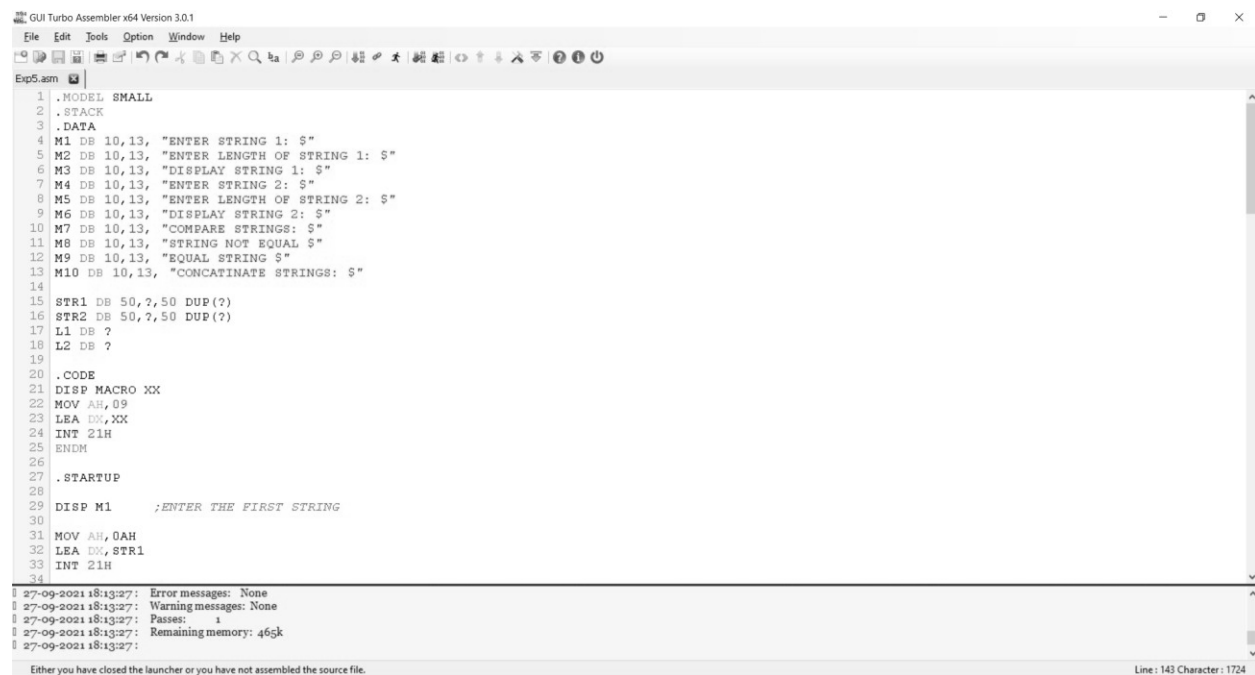
Step XII : Jump to step XIV.

Step XIII : Print “unequal strings” using Macro.

Step XIV : Stop.

Application: Use of string operations in the Assembly Language programming to write modular program.

Design:



```
1 .MODEL SMALL
2 .STACK
3 .DATA
4 M1 DB 10,13, "ENTER STRING 1: $"
5 M2 DB 10,13, "ENTER LENGTH OF STRING 1: $"
6 M3 DB 10,13, "DISPLAY STRING 1: $"
7 M4 DB 10,13, "ENTER STRING 2: $"
8 M5 DB 10,13, "ENTER LENGTH OF STRING 2: $"
9 M6 DB 10,13, "DISPLAY STRING 2: $"
10 M7 DB 10,13, "COMPARE STRINGS: $"
11 M8 DB 10,13, "STRING NOT EQUAL $"
12 M9 DB 10,13, "EQUAL STRING $"
13 M10 DB 10,13, "CONCATINATE STRINGS: $"
14
15 STR1 DB 50,2,50 DUP(?)
16 STR2 DB 50,2,50 DUP(?)
17 L1 DB ?
18 L2 DB ?
19
20 .CODE
21 DISP MACRO XX
22 MOV AH,09
23 LEA DX,XX
24 INT 21H
25 ENDM
26
27 .STARTUP
28
29 DISP M1 ;ENTER THE FIRST STRING
30
31 MOV AH,0AH
32 LEA DX,STR1
33 INT 21H
34
```

27-09-2021 18:13:27: Error messages: None
27-09-2021 18:13:27: Warning messages: None
27-09-2021 18:13:27: Passes: 1
27-09-2021 18:13:27: Remaining memory: 465k
27-09-2021 18:13:27:

Line : 143 Character : 1724

```
GUI Turbo Assembler v64 Version 3.0.1
File Edit Tools Option Window Help
Exp5.asm
35 DISP M2 ;LENGTH OF THE FIRST STRING
36 LEA SI, STR1+1
37
38 MOV CL,[SI]
39 MOV L1,CL
40 ADD CL,30H
41 MOV DL,CL
42 MOV AH,02
43 INT 21H
44
45 DISP M3 ;DISPLAT STR3
46 LEA SI,STR1+2
47 MOV CL,L1
48
49 BACK1:
50 MOV DL,[SI]
51 MOV AH,02
52 INT 21H
53 INC SI
54 DEC CL
55 JNZ BACK1
56
57 DISP M4 ;ENTER THE FIRST STRING
58
59 MOV AH,0AH
60 LEA DX,STR2
61 INT 21H
62
63 DISP M5 ;LENGTH OF THE FIRST STRING
64 LEA DI,STR2+1
65
66 MOV CL,[DI]
67 MOV L2,CL
68 ADD CL,30H
27-09-2021 18:13:27: Error messages: None
27-09-2021 18:13:27: Warning messages: None
27-09-2021 18:13:27: Passes: 1
27-09-2021 18:13:27: Remaining memory: 465k
27-09-2021 18:13:27:
Either you have closed the launcher or you have not assembled the source file.
Line: 143 Character: 1724
```

```
GUI Turbo Assembler v64 Version 3.0.1
File Edit Tools Option Window Help
Exp5.asm
69 MOV DL,CL
70 MOV AH,02
71 INT 21H
72
73 DISP M6 ;DISPLAT STR3
74 LEA DI,STR2+2
75 MOV CL,L2
76
77 BACK2:
78 MOV DL,[DI]
79 MOV AH,02
80 INT 21H
81 INC DI
82 DEC CL
83 JNZ BACK2
84
85 DISP M7
86
87 MOV CL,L1
88 MOV CH,L2
89 CMP CL,CH
90 JNZ N_EQUAL
91
92 LEA SI,STR1+2
93 LEA DI,STR2+2
94
95 BACK3:
96 MOV DL,[SI]
97 CMP DL,[DI]
98 JNZ N_EQUAL
99 INC SI
100 INC DI
101 DEC CL
102 JNZ BACK3
103 DISP M9
104 JMP CONCAT
105
27-09-2021 18:13:27: Error messages: None
27-09-2021 18:13:27: Warning messages: None
Either you have closed the launcher or you have not assembled the source file.
Line: 143 Character: 1724
```



```
GUI Turbo Assembler v64 Version 3.0.1
File Edit Tools Option Window Help
Exp5.asm
107 DISP M8
108
109 DISP M10
110
111 CONCAT:
112 MOV CL, L1
113 MOV BL, CL
114 MOV CH, L2
115 ADD BL, CH
116
117 LEA SI, STR1+2
118 LEA DI, STR2+2
119
120 BACK4:
121 INC SI
122 DEC CL
123 JNZ BACK4
124
125 BACK5:
126 MOV DL, [DI]
127 MOV [SI], DL
128 INC SI
129 INC DI
130 DEC CH
131 JNZ BACK5
132
133 LEA SI, STR1+2
134 BACK6:
135 MOV DL, [SI]
136 MOV AH, 02
137 INT 21H
138 INC SI
139 DEC BL
140 JNZ BACK6
141
142 .EXIT
143 END
27-09-2021 18:13:27: Error messages: None
Line: 143 Character: 1724
```

Result and Discussion:

Learning Outcomes: The student should have the ability to

LO1: List the string instructions

LO2: Describe the string addressing mode.

LO3: Use of string instructions in the program to perform different string operations.

Course Outcomes: Upon completion of the course students will be able to make use of instructions of 8086 to build assembly and Mixed language programs.

Conclusion: Thus, the Assembly Language Program (ALP) on various string operations like (i) Accept the string, (ii) Display the string, (iii) Concatenate two strings and (iv) compare two strings was completed.

Viva Questions:

1. List the different string instructions.
2. Explain different string addressing modes.

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				