STRING MANIPULATIONS

Exp No: 3 Name: Srinithyee S K

Date: 12-09-20 Register Number: 185001166

AIM:

To write assembly language programs to perform the following basic string operations.

- 1. To move a string of bytes.
- 2. To compare two strings of bytes.
- 3. To search a byte in a string of bytes.
- 4. To move a string without using string instructions.

PROGRAM 1: MOVING A STRING OF BYTES

- 1. Begin
- 2. Declare data segment.
- 3. Initialize data segment with variables for storing the source string and its length.
- 4. Close the data segment.
- 5. Declare extra segment.
- 6. Initialize extra segment with a variable to store the destination string.
- 7. Close the extra segment.
- 8. Declare code segment.
- 9. Set a preferred offset (preferably 100h)
- 10. Load the data segment content into AX register.
- 11. Transfer the contents of AX register to DS register.
- 12. Load the extra segment content into AX register.
- 13. Transfer the contents of AX register to ES register.
- 14. Transfer to CX the length of source string.
- 15. Have SI point to source string and DI to destination string.
- 16. Clear the direction flag.
- 17. Repeat until CX is zero:
 - (I) Transfer data from SI to DI using MOVSB.
- 18. Safely exit the program using an interrupt signal.
- 19. Close the code segment.
- 20. End

PROGRAM	COMMENTS
assume cs:code, ds:data, es:extra	Declare code and data segment.
data segment str1 db "MASM" data ends	Initialize data segment with values. Stores a string STR1.
extra segment str2 db ? strlen dw 0004h extra ends	Initialize extra segment with values. Declaring a string with no preset value. Declaring the length of the string STR1.
code segment	Start the code segment.
org 0100h	Initialize an offset address.
start: mov ax, data	Transfer data from "data" to AX.
mov ds, ax	Transfer data from memory location AX to DS.
mov ax, extra	Transfer the data from "extra" to AX.
mov es, ax	Transfer the data AX to ES.
mov si, offset str1	Store the starting offset address of STR1 in SI.
mov di, offset str2	Store the starting offset address of STR2 in DI.
mov cx, strlen	Store the length of STR1 in CX.
cld	Clear directional flag value.
rep movsb	Repeat MOVSB instruction till $CX \neq 0$.
	MOVSB copies bytes from DS to ES.
mov ah, 4ch	
int 21h	Interrupt the process with return code and exit.
code ends	
end start	

```
BB DOSBox 0.74-3, Cpu speed:
                      3000 cycles, Frameskip 0, Progra...
                                                           ×
        076A:0040
076A:0050
        076A:0060
076A:0070
Q:\>DEBUG MOUSTR.exe
076C:0100 B86A07
                  MOV
                         AX,076A
076C:0103 8ED8
                   MOV
                         DS,AX
076C:0105 B86B07
                         AX,076B
                   MOV
076C:0108 8EC0
                   MOV
                         ES,AX
076C:010A BE0000
                   MOV
                         SI,0000
076C:010D BF0000
                   MOV
                         DI,0000
076C:0110 26
                   ES:
076C:0111 8B0E0100
                   MOV
                         CX,[0001]
076C:0115 FC
                   CLD
076C:0116 F3
                   REPZ
076C:0117 A4
                   MOUSB
076C:0118 B44C
                   MOV
                         AH,4C
076C:0110 D110
076C:011A CD21
076C:011C FA
                   INT
                         21
                   CLI
076C:011D 10B0FF72
                   ADC
                         [BX+SI+72FF1,DH
```

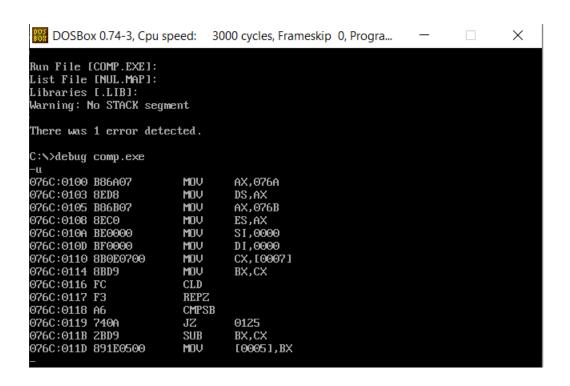
SAMPLE I/O SNAPSHOT:

```
BOSBox 0.74-3, Cpu speed:
          3000 cycles, Frameskip 0, Progra...
                           X
076C:011A CD21
076C:011C 0000
        ADD
           [BX+SI],AL
076C:011E 0000
           [BX+SI],AL
        ADD
-d 076A:0000
076A:0000 4D 41 53 4D 00 00 00 00-00 00 00 00 00 00 00 00
076A:0030
   076A:0040
   076A:0050
   076A:0060
-g
Program terminated normally
-d 076B:0000
076B:0000 4D 41 53 4D 00 00 00 00-00 00 00 00 00 00 00 00
076B:0020
   076B:0030
076B:0040
   076B:0050
   076B:0060
076B:0070
```

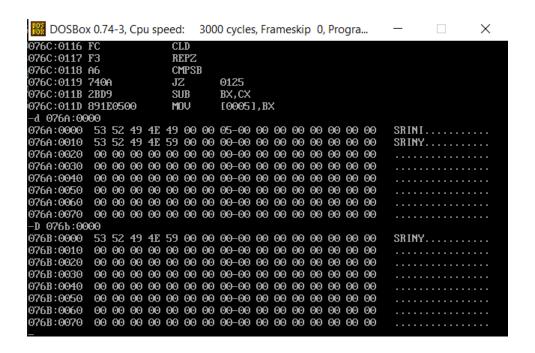
PROGRAM – 2: COMPARING 2 STRINGS OF BYTES:

- 1. Begin.
- 2. Declare data segment.
- 3. Initialize data segment with variables for storing the source string and its length.
- 4. Close the data segment.
- 5. Declare extra segment.
- 6. Initialize extra segment with variables for storing the destination and its string.
- 7. Close the extra segment.
- 8. Declare code segment.
- 9. Set a preferred offset (preferably 100h)
- 10. Load the data segment content into AX register.
- 11. Transfer the contents of AX register to DS register.
- 12. Load the extra segment content into AX register.
- 13. Transfer the contents of AX register to ES register.
- 14. Transfer to CX the length of source string.
- 15. Have SI point to source string and DI to destination string.
- 16. Clear the direction flag.
- 17. Repeat until CX is zero or a mismatch is found:
 - (I) Compare data in SI and DI using CMPSB, and increment the pointers.
- 18. If a mismatch is found, find the index of it by subtracting it with the source string's length.
- 19. Else, store zero as result. (String equality)
- 20. Safely exit the program using an interrupt signal.
- 21. Close the code segment.
- 22. End

PROGRAM	COMMENTS
assume cs:code, ds:data, es:extra	Declare code and data segment.
data segment str1 db "SRINI" streq dw 0000h strlen dw 0005h data ends	Initialize data segment with values. Stores a string STR1. Variable to store the result of the comparison. Variable to hold the length of STR1.
extra segment str2 db "SRINY" extra ends	Initialize extra segment with values. Declaring a string STR2.
code segment org 0100h start: mov ax, data mov ds, ax mov ax, extra mov es, ax mov si, offset str1 mov di, offset str2 mov cx, strlen mov bx, cx cld repe cmpsb	Start the code segment. Initialize an offset address. Transfer data from "data" to AX. Transfer data from memory location AX to DS. Transfer the data from "extra" to AX. Transfer the data AX to ES. Store the starting offset address of STR1 in SI. Store the starting offset address of STR2 in DI. Store the length of STR1 in CX. Copy the value of CX to BX. Clear directional flag value. Repeat CMPSB instruction till ZF = 1. CMPSB compares bytes of ES to corresponding bytes of
jz equstr sub bx, cx mov streq, bx mov ah, 4ch int 21h equstr: mov streq, 0000h	DS. Jump to "EQUSTR" if ZF = 0. Subtract value of CX from value of BX. Transfer the value of BX to variable STREQ. Interrupt the process with return code and exit. Transfer 0 to STREQ indicating STR1 and STR2 are equal.
mov ah, 4ch int 21h code ends end start	Interrupt the process with return code and exit.



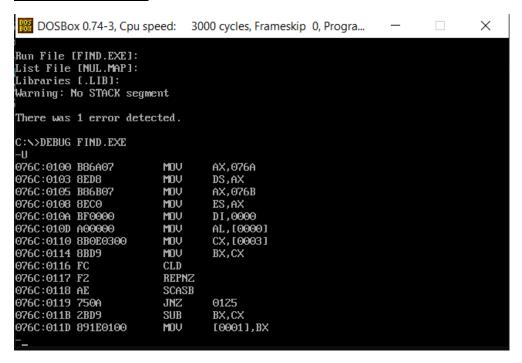
SAMPLE I/O SNAPSHOT:



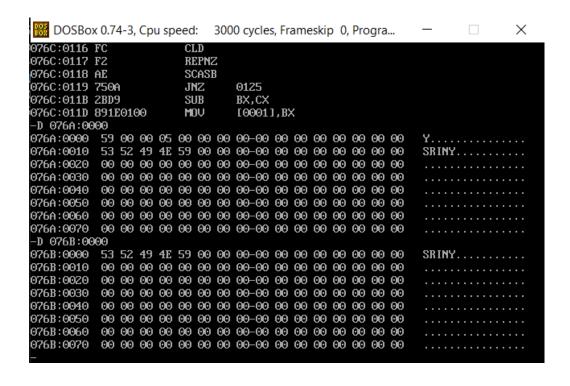
PROGRAM 3: SEARCHING A BYTE IN A STRING

- 1. Begin.
- 2. Declare data segment.
- 3. Initialize data segment with a variable for the byte to be searched and variables to store the location and source string's length.
- 4. Close the data segment.
- 5. Declare extra segment.
- 6. Initialize extra segment with a variable to store the source string.
- 7. Close the extra segment.
- 8. Declare code segment.
- 9. Set a preferred offset (preferably 100h)
- 10. Load the data segment content into AX register.
- 11. Transfer the contents of AX register to DS register.
- 12. Load the extra segment content into AX register.
- 13. Transfer the contents of AX register to ES register.
- 14. Transfer to CX the length of source string.
- 15. Store in AL the byte to be searched for and in DI the source string.
- 16. Clear the direction flag.
- 17. Repeat until CX is zero or until a match is found:
 - (I) Compare data in AL and DI using SCASB, and increment the pointers.
- 18. If a match is found, find its position by subtracting it with the source string's length.
- 19. Else, store zero as result. (Byte not found)
- 20. Close the code segment.
- 21. End

PROGRAM			RAM	COMMENTS
assume cs:code, ds:data, es:extra			ta, es:extra	Declare code and data segment.
1.4.				Totaline data assumed with audion
data se	egment	.11.	"Y"	Initialize data segment with values.
	str2	db	=	Stores a string STR1.
	strloc	dw	0000h	Variable to store the index location.
data e	strlen	dw	0005h	Variable to hold the length of STR1.
data e	nas			
extra segment				Initialize extra segment with values.
	str1	db	"VENKY"	Declaring a string STR2.
extra e	ends			
code s	egment			Start the code segment.
code s	org 0100h			Initialize an offset address.
start:	•			Transfer data from "data" to AX.
50020	mov ds, ax			Transfer data from memory location AX to DS.
	mov ax, extra			Transfer the data from "extra" to AX.
	mov es, ax			Transfer the data AX to ES.
mov di, offset str1			str1	Store the starting offset address of STR1 in DI.
	mov al, str2			Store STR2 in AL.
	mov cx, strlen			Store the length of STR1 in CX.
mov bx, cx				Copy the value of CX to BX.
cld				Clear directional flag value.
repne scasb				Repeat SCASB instruction till $ZF = 0$.
	-			SCASB scans for the occurrence of the byte in AL in ES.
	jnz notfnd			Jump to "NOTFND" if $ZF = 1$.
sub bx, cx				Subtract value of CX from value of BX.
mov strloc, bx			ζ.	Transfer the value of BX to variable STRLOC.
	mov ah, 4ch			
	int 21h			Interrupt the process with return code and exit.
notfnd	notfnd: mov strloc, 0000h			Transfer 0 to STRLOC indicating STR2 was not found in STR1.
mov ah, 4ch				
	int 21h			Interrupt the process with return code and exit.
code e	ends			
end sta	art			



SAMPLE I/O SNAPSHOT:



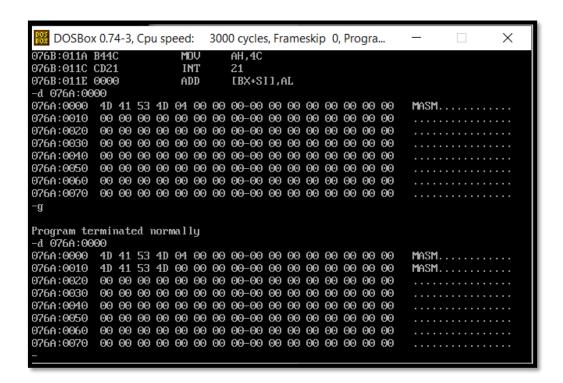
PROGRAM 4: MOVING WITHOUT USING STRING INSTRUCTIONS

- 1. Begin
- 2. Declare data segment.
- 3. Initialize data segment with variables for storing the source string and its length.
- 4. Close the data segment.
- 5. Declare extra segment.
- 6. Initialize extra segment with a variable to store the destination string.
- 7. Close the extra segment.
- 8. Declare code segment.
- 9. Set a preferred offset (preferably 100h)
- 10. Load the data segment content into AX register.
- 11. Transfer the contents of AX register to DS register.
- 12. Load the extra segment content into AX register.
- 13. Transfer the contents of AX register to ES register.
- 14. Store in CX the length of the source string.
- 15. Have SI point to the source string and DI to the destination string.
- 16. Clear the direction flag.
- 17. Repeat until CX is zero:
 - (I) Transfer data from SI to DI
 - (II) Increment SI and DI
 - (III) Decrement CX
- 18. Safely exit the program using an interrupt signal.
- 19. Close the code segment.
- 20. End

PROGRAM	COMMENTS
assume cs:code, ds:data, es:extra	Declare code and data segment.
data segment	Initialize data segment with values.
str1 db "MASM"	Stores a string STR1.
strlen dw 0004h	Variable to hold the length of STR1.
str2 db ?	Declaring a string with no preset value.
data ends	
code segment	Start the code segment.
org 0100h	Initialize an offset address.
start: mov ax, data	Transfer data from "data" to AX.
mov ds, ax	Transfer data from memory location AX to DS.
mov si, offset str1	Store the starting offset address of STR1 in SI.
mov di, offset str2 + 000Ah	Store the starting offset address of STR2 + 000Ah in DI.
mov cx, strlen	Store the length of STR1 in CX.
looper: mov bl, [si]	Copy the value at SI's address location to BL.
mov [di], bl	Copy the value in BL to DI's address location.
inc si	Increment SI.
inc di	Increment DI.
dec cx	Decrement CX.
jz break	Jump to "BREAK" if CX = 0.
jmp looper	Unconditionally jump back to "LOOPER".
break: mov ah, 4ch	
int 21h	Interrupt the process with return code and exit.
code ends	interrupt the process with return code and exit.
end start	
Ciiu Stai t	

```
DOSBox 0.74-3, Cpu speed:
                    3000 cycles, Frameskip 0, Progra...
                                                       \times
        076A:0040
       076A:0050
       076A:0060
076A:0070
       Q:>>DEBUG MOUSTR2.EXE
-u
076B:0100 B86A07
                 MOV
                       AX,076A
076B:0103 8ED8
                 MOV
                       DS,AX
076B:0105 BE0000
076B:0108 BF1000
                       SI,0000
                 MOU
                       DI,0010
                 MOV
                       CX,[0004]
076B:010B 8B0E0400
                 MOV
076B:010F 8A1C
                       BL,[SI]
                 MOV
076B:0111 881D
                       [DI],BL
                 MOV
076B:0113 46
                 INC
                       SI
076B:0114 47
                       DI
                 INC
076B:0115 49
                 DEC
                       cx
076B:0116 7402
                 JΖ
                       011A
076B:0118 EBF5
                 JMP
                       010F
076B:011A B44C
                 MOV
                       AH,4C
076B:011C CD21
                 INT
                       [BX+SI],AL
076B:011E 0000
                 ADD
```

SAMPLE I/O SNAPSHOT:



RESULT:

The assembly level programs were written to perform the above specified basic string operations and the output was verified.