**SSN College of Engineering**

**Department of Computer Science and Engineering**

**UCS1512 – Microprocessors Lab**

**STRING MANIPULATIONS**

Ex No: 03 Name: **Kshitij Sharma**

Date: 10-09-2020 Register Number:**185001080**

**AIM:**

To write and execute 8086 ASL programs for string manipulations like moving a string of bytes, comparing two strings of bytes, searching a byte in a string using string instructions and also to move a string without using string instructions.

**PROCEDURE:**

* Firstly, write the 8086 ASL program for moving a string of bytes using editor(like notepad) and save it with .asm(movestr.asm) extension and move to the MASM folder.
* Now mount the MASM folder in DOSBOX(“mount d e:/masm”) and then enter into the mounted drive(“d:”)
* Now using “edit movestr.asm”,we can edit or create a asm file for execution and then save and exit.
* Assemble the code using “masm movstr.asm” to generate the “movstr.obj” file.
* Link the file using “link movstr.obj;” to generate the executable “movstr.exe” file.
* Now enter the debug mode using “debug movstr.exe” to execute and analyse memory contents. The various commands used in debug mode are as follows:-
* U :- To display unassembled code
* D :- Used as ‘D segment:offset’ to see the content of memory locations starting from segment:offset address.
* E:- To change the value in memory
* G:- To execute
* Q:- To quit

**3a) MOVING A STRING OF BYTES**

**ALGORITHM:**

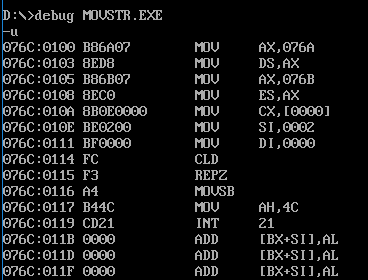
* Move the data segment to the AX register and then move it to the DS register.
* Move the extra segment to the AX register and then move it to the ES register.
* Move the count value to CX register.
* Move the offset of str1 and str2 to SI and DI respectively.
* Clear the Direction Flag.
* Now use rep 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:**

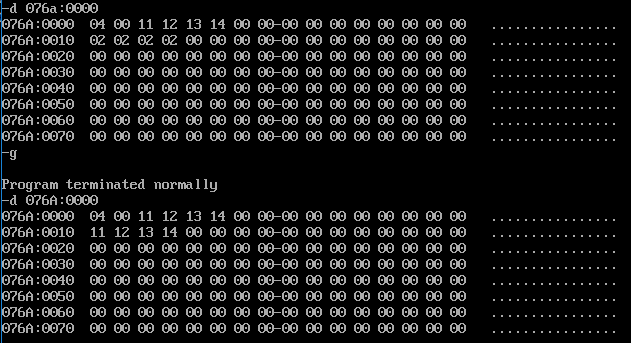
|  |  |
| --- | --- |
| **PROGRAM** | **COMMENTS** |
| mov cx,count  mov si, offset str1  mov di, offset str2  cld  rep movsb | Transfers count’s value to CX register.  Transfers offset address of str1 to SI register.  Transfers offset address of str2 to DI register.  Clear Direction Flag(DF=0)  Repeats moving bytes from [SI] to [DI](For offsets SI and DI the base addresses are present in DS and ES respectively) until CX becomes 0.  In each iteration,CX is decremented by 1 while SI and DI are incremented by 1 as DF=0. |
| mov ah,4ch  int 21h | Moves the hexadecimal value 4c to ah. When Software interrupt 21 is called with AH=4C, then current process terminates.  (i.e., These two instructions are used for the termination of the process). |

**SNAPSHOT:**

**UNASSEMBLED CODE:**

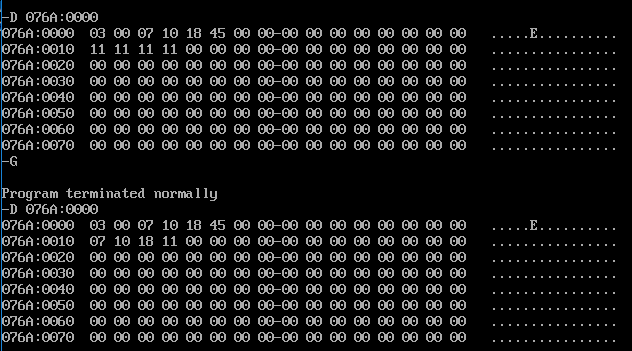


**Input :** str1: 11 12 13 14 str2: 02 02 02 02 Count:0004

**Output:** str2: 11 12 13 14 

**Input :** str1: 07 10 18 45 str2: 11 11 11 11 Count: 0003

**Output:** str2: 07 10 18 11



**3B) COMPARING 2 STRINGS OF BYTES:**

**ALGORITHM:**

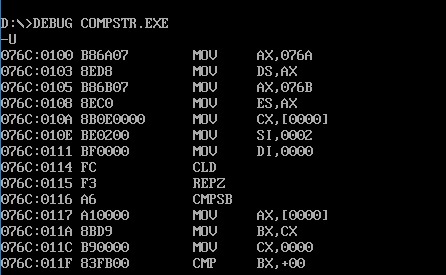
* Move the data segment to the AX register and then move it to the DS register.
* Move the extra segment to the AX register and then move it to the ES register.
* Move the COUNT value to CX register.
* Move the offset of str1 and str2 to SI and DI respectively.
* Clear the Direction Flag.
* Now use repe 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:**

|  |  |
| --- | --- |
| **PROGRAM** | **COMMENTS** |
| 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 | Transfers count’s value to CX register.  Transfers offset address of str1 to SI register.  Transfers offset address of str2 to DI register.  Clear Direction Flag(DF=0)  Repeats comparing bytes from [SI] to [DI](For SI and DI the base addresses are present in DS and ES respectively) until first mismatch occurs or if CX = 0  In each iteration,CX is decremented by 1 while SI and DI are incremented by 1 as DF=0.  Move COUNT value to AX.  Move BX value to CX.  Initialise CX as 0000h.  Compares BX value and 0000h  If ZF=1,jump to here1 label.  AX=AX-BX  If CF=0,jump to here label.  AX=2’s complement(AX)  CX=CX+1  Move AX to STATUS.  Moves the hexadecimal value 4c to ah. When Software interrupt 21 is called with AH=4C, then current process terminates.  (i.e., These two instructions are used for the termination of the process).  Set Direction Flag(DF=1)  Move COUNT value to CX.  Repeats comparing bytes from [SI] to [DI](For offsets provided by SI and DI the base addresses are present in DS and ES respectively) until first mismatch occurs or if CX = 0  In each iteration,CX,SI,DI are decremented by 1 as DF=1.  Compare CX and 0000h.  Move 0000h to AX.  If ZF=1(as a result of cmp CX,0000h),jump to here label.  Move COUNT value to AX.  Jump to Here label. |

**SNAPSHOT:**

**UNASSEMBLED CODE:**



**Input :** str1:aa bb cc dd ee str2:0a db cc dd ee Count: 0005

**Output:** 0001



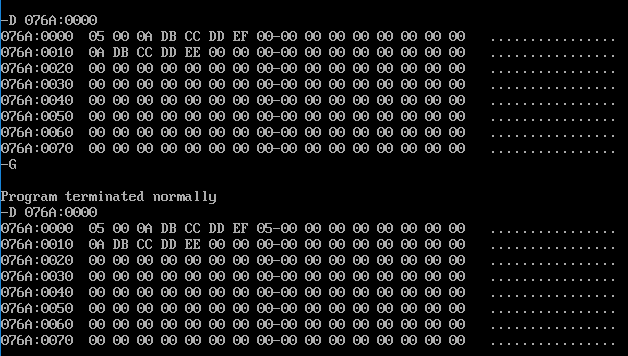
**Input :** str1:0a db cc dd ee str2:0a db cc dd ee Count: 0005

**Output:** 0000



**Input :** str1:0a db cc dd ef str2:0a db cc dd ee Count: 0005

**Output:** 0005



**3C)SEARCHING A BYTE IN A STRING:**

**ALGORITHM:**

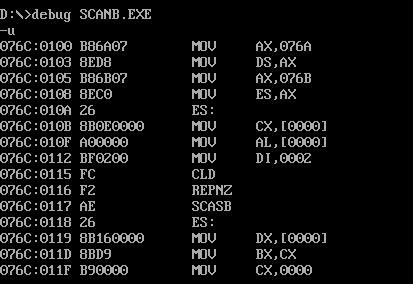
* Move the data segment to the AX register and then move it to the DS register.
* Move the extra segment to the AX register and then move it to the ES register.
* Move the COUNT value to CX register.
* Move the STR1 to AL register and offset of STR2 to DI.
* Clear the Direction Flag.
* Now use repne 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:**

|  |  |
| --- | --- |
| **PROGRAM** | **COMMENTS** |
| 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 | Transfers count’s value to CX register.  Transfers str1 to AL  Transfers offset address of STR to DI register.  Clear Direction Flag(DF=0)  Repeats comparing the byte in AL and [DI](For DI the base address is present in ES) until first matching byte is found or if CX = 0.In each iteration, CX is decremented by 1 while DI is incremented by 1 as DF=0.  Move COUNT value to DX.  Move BX value to CX.  Initialise CX as 0000h.  Compares BX value and 0000h  If ZF=1,jump to here1 label.  DX=DX-BX  If CF=0,jump to here label.  DX=2’s complement(DX)  CX=CX+1  Move DX to STATUS.  Moves the hexadecimal value 4c to ah. When Software interrupt 21 is called with AH=4C, then current process terminates.  (i.e., These two instructions are used for the termination of the process).  Set Direction Flag(DF=1)  Move COUNT value to CX.  Repeats comparing the byte in AL and [DI](For DI the base address is present in ES) until first matching byte is found or if CX = 0.In each iteration, CX,DX are decremented by 1 as DF=1.  Compare CX and 0000h.  Move 0000h to DX.  If ZF=1(as a result of cmp CX,0000h),jump to here label.  Move COUNT value to DX.  Jump to Here label. |

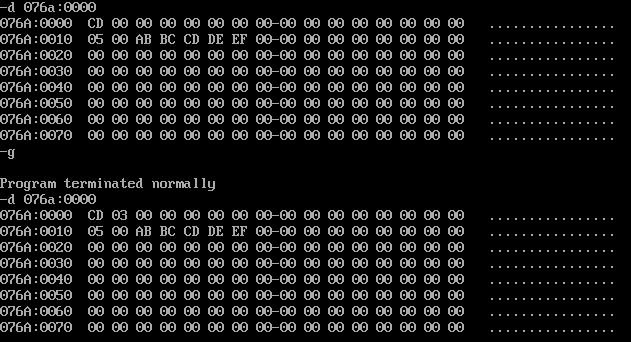
**SNAPSHOT:**

**UNASSEMBLED CODE:**



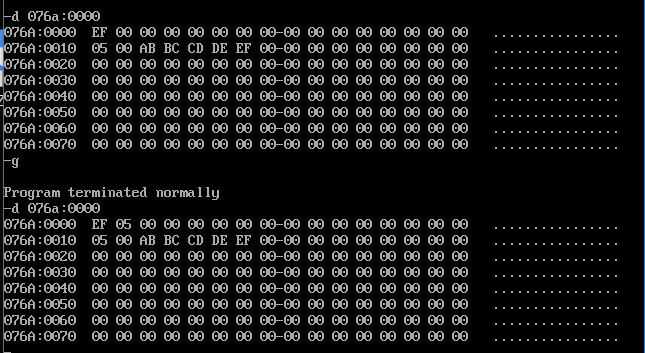
**Input :** str: ab bc cd de ef str1: cd Count: 0005

**Output:** 0003



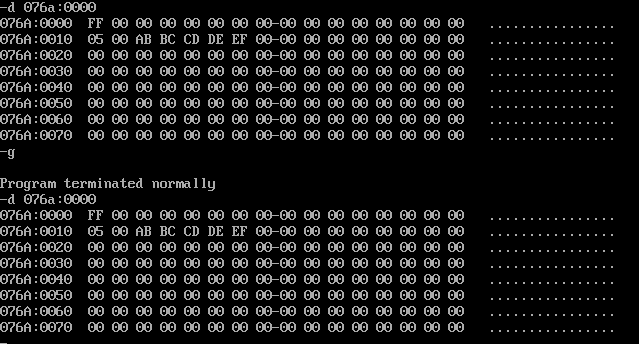
**Input :** str: ab bc cd de ef str1: ef Count: 0005

**Output:** 0005



**Input :** str: ab bc cd de ef str1: ff Count: 0005

**Output:** 0000



**3D)MOVING A STRING WITHOUT USING STRING INSTRUCTIONS:**

**ALGORITHM:**

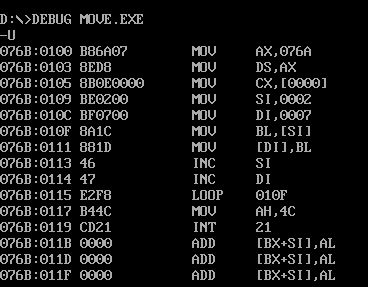
* Move the data segment to the AX register and then move it to the DS register.
* Move the extra segment to the AX register and then move it to the ES register.
* Move the COUNT value to CX register.
* Move the offset of str1 and str2 to SI and DI respectively.
* Move [SI](offset is specified by SI while base address is specified by DS) to BL.
* Move BL register value to [DI] (offset is specified by DI while base address is specified by DS).
* Increment SI and DI and use loop instruction which will continue this until CX becomes 0.

**PROGRAM:**

|  |  |
| --- | --- |
| **PROGRAM** | **COMMENTS** |
| mov cx,count  mov si,offset str1  mov di, offset str2  here:  mov bl,[si]  mov [di],bl    inc si  inc di  loop here | Transfers count’s value to CX register.  Transfers offset address of str1 to SI register.  Transfers offset address of str2 to DI register.  Move value specified in [SI]((offset is specified by SI while base address is specified by DS) to BL register.  Move BL register value to [DI]( offset is specified by DI while base address is specified by DS).  SI=SI+1  DI=DI+1  Loop here label until CX becomes 0.For every iteration CX is decremented by 1. |
| mov ah,4ch  int 21h | Moves the hexadecimal value 4c to ah. When Software interrupt 21 is called with AH=4C, then current process terminates.  (i.e., These two instructions are used for the termination of the process). |

**SNAPSHOT:**

**UNASSEMBLED CODE:**



**Input :** str1:aa bb cc dd ee str2: ca Count: 0005

**Output:** str2: aa bb cc dd ee



**RESULT:**

Thus,8086 ASL programs for string manipulations like moving a string of bytes, comparing two strings of bytes, searching a byte in a string using string instructions and also to move a string without using string instructions have been executed successfully using MS - DOSBox.