**Experiment No 3: String Manipulations**

**Exp No:**3 **Name:** Sneha Sriram Kannan

**Date:** 11-09-2020 **Register Number:** 185001157

**1. AIM**

To write a MASM program to move a string of bytes given the source string, size of source, starting location of source and destination location as input

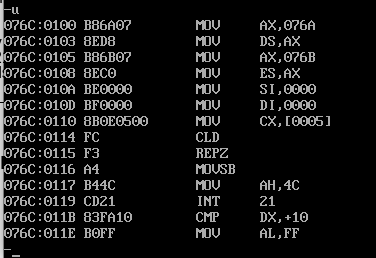
**ALGORITHM**

* Initialize the data segment with the string to move and the length of the string
* Allocate memory in the extra segment for the destination
* Load address of data segment into DS register
* Load the address of the extra segment into ES register
* Load the address of the source into SI register
* Load the address of the destination into the DI register
* Store the length of the string in the CX register
* Clear the direction flag
* Repeat while CX!=0
  + Move each byte from the source to destination and decrement CX by 1 for each byte.
* End the program

**PROGRAM**

|  |  |
| --- | --- |
| **PROGRAM** | **COMMENTS** |
| assume cs:code,ds:data,es:extra  data segment  source db 71h,73h,75h,77h,79h;  len dw 0005h;  data ends  extra segment  destination db ?  extra ends  code segment  org 0100h  start:mov ax,data  mov ds,ax  mov ax,extra  mov es,ax  mov si,offset source  mov di,offset destination  mov cx,len  cld  rep movsb  mov ah,4ch  int 21h  code ends  end start | Binds the label and the segment  Assigns values to source  Assigns value 0005h to len since 5 bytes  Space is allocated for the destination  Sets the offset  Memory location of data moved to ax  ax contents copied to ds  Memory location of extra moved to ax  ax contents copied to es  Address of source is stored in SI  Address of destination is stored in DI  Cx is initialized with the length of the string to be moved  Direction flag is cleared  While CX is not 0, data moved from SI to DI  Program termination |

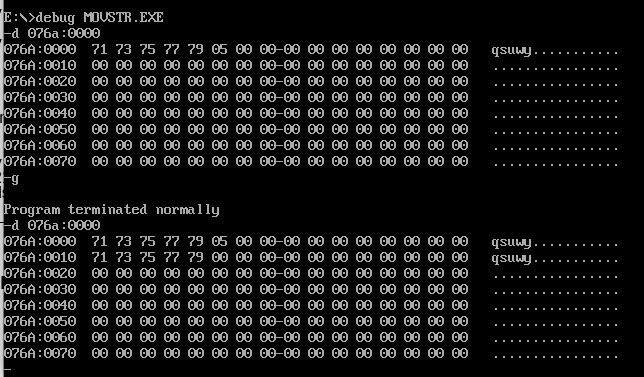
**UNASSEMBLED CODE**

****

**SNAPSHOT OF SAMPLE I/O**

**INPUT:** source - 71 73 75 77 79; len - 0005h

**OUTPUT:** destination - 71 73 75 77 79

****

**RESULT**

The string was moved from the given source to the given destination using string instructions.

**2. AIM**

To write a MASM program to compare 2 strings of bytes and store 0 of they are the same, or the index of the first mismatch

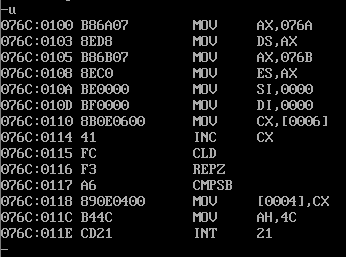
**ALGORITHM**

* Initialize the data segment with the 1 of the strings, result, and the length of the string
* Initialize the extra segment with the other string to compare
* Load address of data segment into DS register
* Load the address of the extra segment into ES register
* Load the address of the source into SI register
* Load the address of the destination into the DI register
* Store the length of the string in the CX register
* Increment the CX register
* Clear the direction flag
* Repeat while equal
  + Compare the byte in the SI and DI register. Decrement CX if equal and SI and DI registers are changed. If not equal, break.
* Store the value of CX in result
* End the program

**PROGRAM**

|  |  |
| --- | --- |
| **PROGRAM** | **COMMENTS** |
| assume cs:code,ds:data,es:extra  data segment  str1 db 1h,72h,73h,79h;  result dw 0000h;  len dw 0004h;  data ends  extra segment  str2 db 71h,72h,73h,74h;  extra ends  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,len  inc cx  cld    repe cmpsb  mov result, cx  mov ah, 4ch  int 21h  code ends  end start | Binds the label and the segment  Assigns values to str1  Assigns 0 to the result  Assigns value 0004h to len since 4 bytes  Assigns values to str2  Sets the offset  Memory location of data moved to ax  ax contents copied to ds  Memory location of extra moved to ax  ax contents copied to es  Address of source is stored in SI  Address of destination is stored in DI  Cx is initialized with the length of the strings to be compared  Direction flag is cleared  While the bytes are equal, compare contents pointed by SI and DI. If not equal, then stop. Otherwise, decrement CX  Move value in CX to result  Program termination |

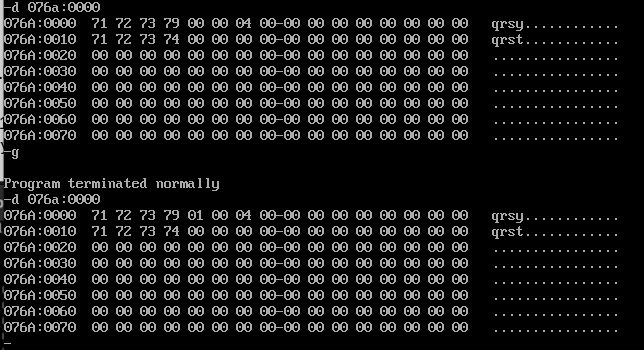
**UNASSEMBLED CODE**

****

**SNAPSHOT OF SAMPLE I/O**

**INPUT:** str1 - 71 72 73 79; len - 0004h ; str2 - 71 72 73 74; result - 00

**OUTPUT:** result - 0001 since there is a mismatch in the 1st byte from the right

****

**RESULT**

The string of bytes were compared and the result was produced depending on whether they were equal or not.

**3. AIM**

To write a MASM program to search for a byte in a string and return 0 if not found or the index of the first occurrence if found.

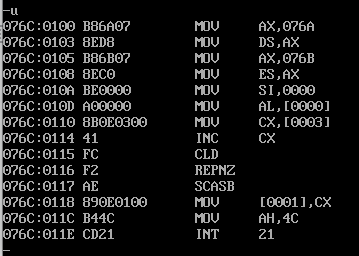
**ALGORITHM**

* Initialize the data segment with the substring, length of the string and variable to store the result
* Initialize the extra segment with the string to search in
* Load address of data segment into DS register
* Load the address of the extra segment into ES register
* Load the address of the source into SI register
* Move the substring into al register
* Store the length of the string in the CX register
* Increment the CX register
* Clear the direction flag
* Repeat while not equal
  + Compare the byte in the SI and ALregister. Decrement CX if not equal and SI and DI registers are changed. If equal, break.
* Store the value of CX in result
* End the program

**PROGRAM**

|  |  |
| --- | --- |
| **PROGRAM** | **COMMENTS** |
| assume cs:code,ds:data,es:extra  data segment  str db 72h;  result dw 0000h;  len dw 0004h;  data ends  extra segment  source db 71h, 72h, 73h, 74h  extra ends  code segment  org 0100h  start:mov ax,data  mov ds,ax  mov ax,extra  mov es,ax  mov si,offset src  mov al,str  mov cx,len  inc cx    Cld  repne scasb  mov result, cx  mov ah, 4ch  int 21h  code ends  end start | Binds the label and the segment  Assigns values to str  Assigns 0 to the result  Assigns value 0004h to len since 4 bytes  Assigns values to source  Sets the offset  Memory location of data moved to ax  ax contents copied to ds  Memory location of extra moved to ax  ax contents copied to es  Address of source is stored in SI  Contents of str stored in AL  Cx is initialized with the length of the string  CX incremented  Direction flag is cleared  While the bytes are not equal, compare contents pointed by SI and DI. If equal, then stop. Otherwise, decrement CX  Move value in CX to result  Program termination |

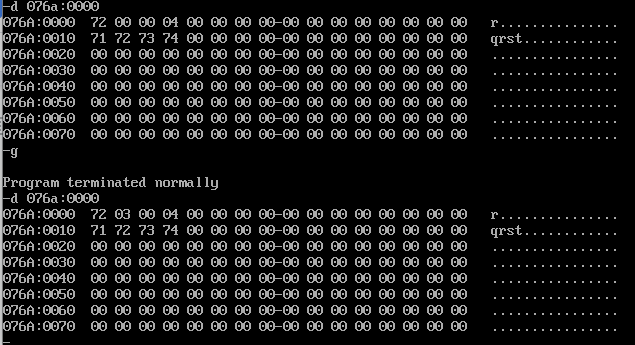
**UNASSEMBLED CODE**

****

**SNAPSHOT OF SAMPLE I/O**

**INPUT:** str - 72, source - 71 72 73 74; len - 0004 ; result - 0000

**OUTPUT:** result - 0003 since 72 is found in the 3rd byte from the right

****

**RESULT**

The searching of a substring in a string was carried out and the result was produced.

**4. AIM**

To write a MASM program to move a string of bytes given the source string, size of source, starting location of source and destination location as input without using string instructions

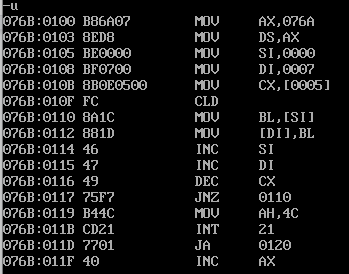
**ALGORITHM**

* Initialize the data segment with the string to move and the length of the string, and allocate memory for the destination
* Load address of data segment into DS register
* Load the address of the source into SI register
* Load the address of the destination into the DI register
* Store the length of the string in the CX register
* Clear the direction flag
* While Zero Flag is not 0
  + Move the contents at the address pointed to by SI to bl
  + Move contents of bl to address pointed by di
  + Increment SI and DI
* End the program

**PROGRAM**

|  |  |
| --- | --- |
| **PROGRAM** | **COMMENTS** |
| assume cs:code,ds:data  data segment  source db 71h,73h,75h,77h,79h;  len dw 0005h;    destination db ?  data ends  code segment  org 0100h  start:mov ax,data  mov ds,ax  mov si,offset source  mov di,offset destination  mov cx,len  Cld  loop1:mov bl,[si]  mov [di],bl  inc si  inc di  dec cx  jnz loop1  mov ah,4ch  int 21h  code ends  end start | Binds the label and the segment  Assigns values to source  Assigns value 0005h to len since 5 bytes  Space is allocated for the destination  Sets the offset  Memory location of data moved to ax  ax contents copied to ds  Address of source is stored in SI  Address of destination is stored in DI  Cx is initialized with the length of the string to be moved  Direction flag is cleared  Move contents pointed to by SI to BL  Move contents of BL to address pointed to by DI  Increment SI  Increment DI  Decrement CX  Go to label loop1 if zero flag not set  Program termination |

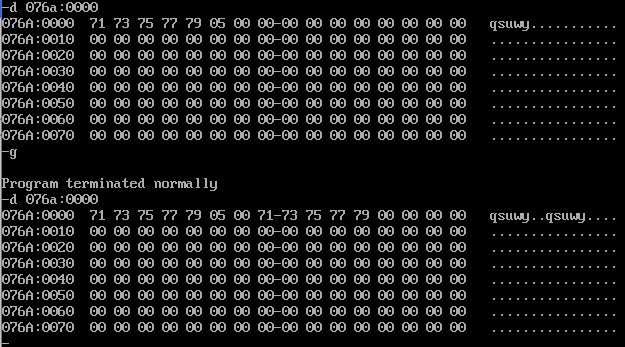
**UNASSEMBLED CODE**



**SNAPSHOT OF SAMPLE I/O**

**INPUT:** source - 71 73 75 77 79; len - 0005h

**OUTPUT:** destination - 71 73 75 77 79

****

**RESULT**

The string was moved from the given source to the given destination without using string instructions.