**Exp No: 3 String Manipulations**

Date: 13-09-2020 Name: Anirudh H

Register Number: 185001019

1. Aim:

To perform String manipulation operations such as moving a string, comparing 2 strings and searching a byte in a string using MASM.

1. Procedure for executing MASM:
   1. Open DOS BOX simulation environment
   2. Mount the directory where the assembler code is present
   3. Move to mounted directory
   4. Assemble code using “masm<filename>.asm” command
   5. Link code the corresponding Obj file using “link <filename>.obj” command
   6. Use semicolon to avoid renaming the .exe file
   7. Debug created .exe file using “debug <filename>.exe” command
   8. Once inside the debug menu type
      * u: see code
      * d: see the data segment
      * g: run code
      * g: (2nd time) exit
   9. Exit
2. Algorithm
   1. Moving the string

* Start
* Initialize the data segment with the length of string1 and string1 in bytes
* Initialize the extra segment with the final string2
* Move the data segment address and extra segment address to ds and es respectively using ax
* Set the size of string as cx
* Set the string1 offset address as si and the string2 offset address as di
* Move one byte from si to di and reduce cx by 1 (A)
* If cx not equal to 0 go to (A)
* End
  1. Comparing 2 strings
     + Start
     + Initialize the data segment with the length of string1, the string1 and the status after comparing
     + Initialize the extra segment with the string2 having its own bytes but same length as string1
     + Set the assembling offset address of the code segment
     + Move the data and extra segment to ds and es respectively using ax
     + Move the count to cx and bx, and increament cx and bx by 1
     + Move the offset address of string1 and string2 to si and di respectively
     + Compare the bytes in si and di and incremeant to the next byte, reduce cx by 1
     + Perform step ix until cx =0 or any byte is not equal
     + If cx equal to 0 move to step xiii
     + Subtract bx from cx and move the value of bx to cx to get the exact position in the string
     + Move cx to status
     + End
  2. Searching a byte in a string
* Start
* In data segment initialize the searching byte and the position of the found byte
* In extra segment initialize the size of the string and the string in which the search is going to be performed.
* Move data and extra segment ds and es respectively using ax
* Move the string size to cx
* Move the offset address of string to di and the search byte to al
* Clear the direction flag
* Search for byte in string starting at position di increment di and decrement cx
* Repeat step ix until cx = 0 or the byte is found
* If cx = 0 move to step
* Move the string size to ax and subtract cx from ax to get position
* Move ax to cx
* Move the value of cx as found position
* End
  1. Moving a string
     + Start
     + In data segment define the count or length of the string and the string1
     + In Extra segment define the empty string2 to which bytes are going to be moved, string2 length same as string1 length
     + Set the starting offset address for assembling the code
     + Move the data and extra segment to ds and es respectively using ax
     + Move the offset address of string1 and string2 to si and di respectively
     + Move the length of string to cx
     + Move the length of string to bx
     + Subtract cx from bx
     + Move byte at position bx in string1 to string2 position di using al
     + Increment di
     + Go to step viii until cx = 0
     + End

1. Program:
   1. Moving a String

|  |  |
| --- | --- |
| Program | Comments |
| data segment  count dw 0004h  str1 db 11h,12h,13h,14h data ends | Initialize the length of the string as count =4  and the string1 as str1 with 4 bytes 11,12,13,14 (in hex) |
| extra segment  str2 db 00h,00h,00h,00h extra ends | Initialize the destination string with same  length as the string1 as str2 |
| start: mov ax,data  mov ds,ax mov ax,extra mov es,ax | Move the data segment to ds using ax register  Move the extra segment to es using ax register |
| mov cx,count | Move the count or length of string to cx |
| mov si,offset str1 | Move the str1 offset address to si |
| mov di,offset str2 | Move the str2 offset address to di |
| cld | Clear the Direction Flag |
| rep movsb | Repeat moving a byte from si to di and  decrement cx till cx =0 |
| mov ah,4ch  int 21h | Terminate the execution by moving 4ch to  ah register and calling DOS interrupt sub routine 21h |
| code ends | End of code segment |
| end start | End of start |

* 1. Comparing string

|  |  |
| --- | --- |
| Program | Comments |
| data segment  count dw 0004h  str1 db 01h,02h,03h,04h status dw 0000h  data ends | Initialize the length of the string as count =4 and the string1 as str1 with 4 bytes 01,02,03,04 (in hex)  Set the status of compare as 0 intially |
| extra segment  str2 db 01h,02h,03h,05h extra ends | Initialize the destination string with same  length as the string1 as str2 with values 01,02,03,05 ( in hex) |
| code segment  org 0100h | Set the starting assembling offset address as  0100h |
| start: mov ax,data  mov ds,ax mov ax,extra mov es,ax | Move the data segment to ds using ax  register  Move the extra segment to es using ax register |
| mov cx,count | Move the count or length of string to cx |
| inc cx | Increment cx by 1 |
| mov bx,count | Move the count or length of string to bx |
| inc bx | Increment bx by 1 |
| mov si,offset str1 | Move the str1 offset address to si |
| mov di,offset str2 | Move the str2 offset address to di |
| cld | Clear the Direction Flag |
| repe cmpsb | Repeat comparing a byte from si to di and  decrement cx till cx =0 or there is a difference in the bytes |
| cmp cx,00 | To check if the cx value was 0 |
| je here | If the cx value is 0 then move to here line |
| sub bx,cx | Subtract cx from bx to get the position  where the byte is different |
| mov cx,bx | To copy the value of bx to cx |
| here: mov status,cx | To copy the value of cx to status |
| mov ah,4ch  int 21h | Terminate the execution by moving 4ch to  ah register and calling DOS interrupt sub routine 21h |
| code ends | End of code segment |
| end start | End of start |

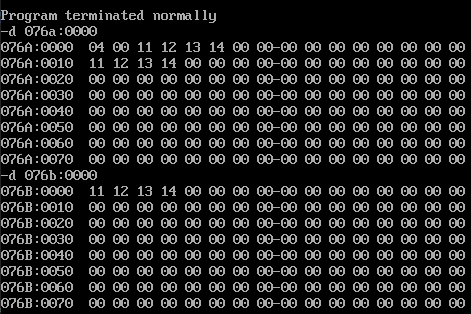
* 1. Searching a byte

|  |  |
| --- | --- |
| Program | Comments |
| data segment  search\_bt db 02h found\_pos dw 0000h  data ends | Naming the cs, ds and es segment of 8086 as code ,data and extra segment respectively |
| extra segment  str\_size dw 0004h  str1 db 01h,02h,04h,05h extra ends | Initialize the search byte as search\_bt and the position if found as found\_pos as 0 |
| code segment  org 0100h | Initialize the string size as str\_size Initialzise the strting as str1 |
| start: mov ax,data  mov ds,ax  mov ax,extra mov es,ax | Move the data segment to ds using ax register  Move the extra segment to es using ax register |
| mov cx,str\_size | Move the str\_size or length of string to cx |
| mov di,offset str1 | Move the str1 offset address to di |
| cld | Move the search\_bt to al |
| repe cmpsb | Clear the Direction Flag |
| cmp cx,00 | Repeat searching al byte from di;  Increment di and decrement cx till cx =0 or the byte is foung |
| je here | To check if the cx value was 0 |
| Mov ax,str\_size | If the cx value is 0 then move to here line |
| sub ax,cx | Move the str\_size to ax to get the position |
| mov cx,ax | Subtract cx from ax to get the position  where the byte is found |
| here: mov found\_pos,cx | To copy the value of ax to cx |
| mov ah,4ch  int 21h | To copy the value of cx to found\_pos |
| code ends | Terminate the execution by moving 4ch to  ah register and calling DOS interrupt sub routine 21h |
| end start | End of start |

* 1. Moving a string

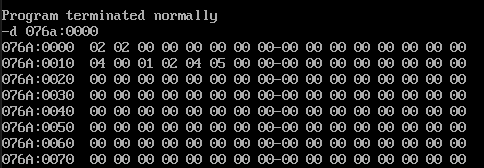
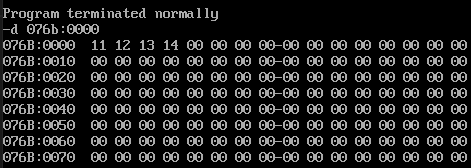
|  |  |
| --- | --- |
| Program | Comments |
| data segment  count dw 0004h  str1 db 11h,12h,13h,14h data ends | Naming the cs, ds and es segment of 8086 as code , data and extra segment respectively |
| extra segment  str2 db 00h,00h,00h,00h extra ends | Initialize the length of the string as count =4 and the string1 as str1 with 4 bytes 11,12,13,14 (in hex) |
| code segment  org 0100h | Initialize the destination string with same  length as the string1 as str2 |
| start: mov ax,data  mov ds,ax mov ax,extra mov es,ax | Set the starting assembling offset address as  0100h |
| mov cx,count | Move the data segment to ds using ax  register  Move the extra segment to es using ax register |
| mov si,offset str1 | Move the count or length of string to cx |
| mov di,offset str2 | Move the str1 offset address to si |
| next: mov bx,count | Move the str2 offset address to di |
| sub bx,cx | Set the value of bx as count |
| mov al,str1[bx]  mov es:[di],al | Subtract cx from bx to get the next position |
| inc di | Copy the byte at str1[bx] to the str2 in extra  segment offset address di using al |
| loop next | Increment di |
| mov ah,4ch  int 21h | Move back to next address |
| code ends | Terminate the execution by moving 4ch to  ah register and calling DOS interrupt sub routine 21h |
| end start | End of code segment |

1. Snapshot of sample input and output:
   1. Moving string of bytes



* 1. Comparing strings



* 1. Searching a byte
  2. Moving a string

1. Result

We have performed String manipulation operations such as moving a string, comparing 2 strings and searching a byte in a string using MASM.