**Computer Organization and Assembly Language**

|  |  |
| --- | --- |
| **Lab 12** | |
| **Topic** | 1. String operations 2. Video memory |

**PART 1**

**Display Function**

jmp start

str1 db 'HELLO WORLD'

start:

mov ax, 0xb800;

Mov es, ax;

mov di, 500;

mov cx, 11; ; string length, 11 characters.

lea si, str1;

mov ah, 0x1A; ; Attribute byte, use any number

l1:

Mov al, [si];

Inc si ; pointing to next character in string

Mov es:[di],ax; ; printing message on the screen;

Add di,2;

loop l1

.exit

**Clear screen Function**

jmp start

str1 db 'HELLO WORLD'

start:

mov ax, 0xb800;

Mov es, ax;

mov di, 500;

mov cx, 11; ; string length, 11 characters.

lea si, str1;

mov ah, 0x1A; ; Attribute byte, use any number

l1:

Mov al, [si];

Inc si; pointing to next character in string

Mov es:[di],ax; ; printing message on the screen;

Add di,2;

loop l1

mov cx, 2000; ; total screen locations.

mov ax, 0x0720; Attribute byte (07) and (20h) ASCII for space character.

mov di, 0; ; start from top left

l2:

Mov es:[di],ax; ; writing blank spaces on whole screen

Add di,2;

loop l2

.exit

**STRING FUNCTION WITH VIDEO MEMORY**

jmp start

str1 db 'HELLO WORLD'

str2 db 100 DUP(?)

start:

lea si, str1;

lea di, str2;

mov cx, 11; ; string length, 11 characters.

mov bp,0

mov ah, 0x1A; ; Attribute byte, use any number

; By default ES is pointing to Data Segment

display:

mov ax,0xb800

mov es,ax ; Moving Video Memory Address

lodsb

mov es:[bp],ax

add bp,2

mov dx,ds ;Moving Data Segment address to store value

mov es,dx

stosb

loop display

; movsb;

.exit

**Display Number Function**

jmp start

printnum:

mov bp, sp

mov ax, [bp+2] ; load number in cx

Divide:

mov bx, 10 ; use base 10 for division

div bx ; remainder goes to DX and quotient goes to AX

push dx

inc cx

mov dx,0

cmp ax,0

jnz Divide

Display:

mov ax, 0xb800;

Mov es, ax;

mov di, 500;

l1:

pop ax

add al,0x30

mov ah, 0x1A; ; Attribute byte, use any number

Mov es:[di],ax; ; printing message on the screen;

Add di,2;

loop l1

ret

start:

mov ax, 4529

push ax ; place number on stack

call printnum ; call the printnum subroutine

.exit

***Console Display:***

Note : Each cell represents a word.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Row 1,Col 1** | **Row 1,Col 2** | **….** |  |  | **Row 1,Col 80** |
| **Row 2,Col 1** | **Row 2,Col 2** | **…** |  |  | **Row 2,Col 80** |
| **…** | **…** | **…** |  |  | **…** |
| **…** | **…** | **…** |  |  | **…** |
| **…** | **…** | **…** |  |  | **…** |
| **…** | **…** | **…** |  |  | **…** |
| **…** | **….** | **…** |  |  | **…** |
| **…** | **…** | **…** |  |  | **…** |
| **Row 25,Col 1** | **Row 25,Col 2** | **…** | **…** | **…** | **Row 25,Col 80** |

; if you change the second byte, you can change the color of

; the character even after it is printed.

; character attribute is 8 bit value,

; high 4 bits set background color and low 4 bits set foreground color.

; hex bin color

; 0 0000 black

; 1 0001 blue

; 2 0010 green

; 3 0011 cyan

; 4 0100 red

; 5 0101 magenta

; 6 0110 brown

; 7 0111 light gray

; 8 1000 dark gray

; 9 1001 light blue

; a 1010 light green

; b 1011 light cyan

; c 1100 light red

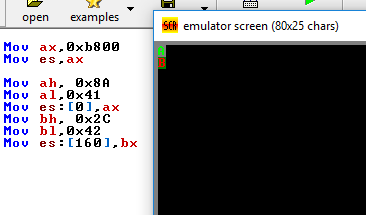
; d 1101 light magenta

; e 1110 yellow

; f 1111 white

**For example:**

* Different attribute values of each word
* Different locations can be accessed for the display.



**NOTES:**

|  |  |
| --- | --- |
| **movsb** | 1. Mov ES:[DI],DS:[SI] 2. Inc si 3. inc di |
| **movsw** | 1. Mov ES:[DI],DS:[SI] 2. add si,2 3. add di,2 |
| **scasb** | 1. cmp al,ES:[DI]   ;ZF=1 if same   1. inc di |
| **scasw** | 1. cmp ax,ES:[DI]   ;ZF=1 if same   1. add di,2 |
| **cmpsb** | 1. cmp DS:[SI],ES:[DI] 2. inc si 3. inc di |
| **cmpsw** | 1. cmp DS:[SI],ES:[DI] 2. add si,2 3. add di,2 |
| **lodsb** | 1. mov al,DS:[SI] 2. inc si |
| **lodsw** | 1. mov ax,DS:[SI] 2. add si,2 |
| **stosb** | 1. mov ES:[DI],al 2. inc di |
| **stows** | 1. mov ES:[DI],ax 2. add di,2 |

**PART 2**

**NOTE: Use string operations in every task.**

## Task 1:

Write an assembly language program to display any string of known size with the font color green and background of color red.

SOL:

Jmp start

str1 db 'My name is Rabia'

size dw 16

start:

mov cx,size

lea si,str1

mov ax,0xb800

mov es,ax

mov ah,0x42

mov di,0

L1:

LODSB

STOSW

loop L1

ret

## Task 2:

Write an assembly language program to display your full name diagonally on the screen. Use loops to calculate the next screen location to display the next character.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **M** |  |  |  |  |  |  |  |  |  |
|  | **O** |  |  |  |  |  |  |  |  |
|  |  | **H** |  |  |  |  |  |  |  |
|  |  |  | **H** |  |  |  |  |  |  |
|  |  |  |  | **A** |  |  |  |  |  |
|  |  |  |  |  | **M** |  |  |  |  |
|  |  |  |  |  |  | **M** |  |  |  |
|  |  |  |  |  |  |  | **A** |  |  |
|  |  |  |  |  |  |  |  | **D** |  |

SOL:

Jmp start

str db 'Show me the code',0

start:

mov ax,0xb800

mov es,ax

lea si,str

mov ah,0x0A

mov di,0

L1:

LODSB

STOSW

add di,160

CMP al,0

JNE L1

ret

## Task 3:

Write an assembly language program to read the string and then display the Upper Case letters only on screen.

Let string:

*“tHe* Quick BrOwn fox JumpS over The LaZy DOg*.”*

SOL:

Jmp start

str db 'tHe Quick BrOwn fox JumpS over The LaZy DOg.',0

start:

lea si,str

mov ax,0xb800

mov es,ax

mov ah,0x07

L1:

LODSB

CMP al,65

JNAE L2

CMP al,90

JNLE L2

STOSW

L2:

CMP al,0

JNE L1

ret

## Task 4:

Write an assembly language program to:

Let:

* string1 db ‘Computer Organization and’
* string2 db ‘Assembly language’
* string3 db 100 dup(?)

1. write one string at row 1 and other string at row2.
2. Concatenate the strings and display at row 3.

SOL:

Jmp start

str1 db 'Computer Organization and',0

str2 db 'Assembly language',0

str3 db 100 dup(0)

display:

mov bp,sp

mov di,[bp+2]

mov si,[bp+4]

mov ax,0xb800

mov es,ax

mov ah,0x07

L1:

LODSB

STOSW

CMP al,0

JNE L1

ret

start:

lea si,str1

mov di,0

push si

push di

CALL display

pop si

pop si

lea si,str2

mov di,160

push si

push di

CALL display

pop si

pop si

lea si,str1

lea di,str3

mov bp,320

L2:

mov ax,0xb800

mov es,ax

mov ah,0x07

LODSB

mov es:[bp],ax

add bp,2

mov dx,ds

mov es,dx

STOSB

CMP al,0

JNE L2

lea si,str2

L3:

mov ax,0xb800

mov es,ax

mov ah,0x07

LODSB

mov es:[bp],ax

add bp,2

mov dx,ds

mov es,dx

STOSB

CMP al,0

JNE L3

ret

## Task 5:

Write an assembly language program to calculate the sum of an Array. Display the original Array values and their sum on screen. Let the array be

Array1 db 1,2,3,4,5,6,7,8,9,8,7,6,5,4,3,2,1

*Pattern of display should be*

|  |
| --- |
| Sample String:  1+2+3+4+5+6+7+8+9+8+7+6+5+4+3+2+1 = 81 |

SOL:

jmp start

array1 db 1,2,3,4,5,6,7,8,9,8,7,6,5,4,3,2,1

sum db 0

printNum:

mov bp,sp

mov ax,[bp+4]

mov di,[bp+2]

mov bx,10

divide:

div bx

push dx

inc cx

mov dx,0

CMP ax,0

JNE divide

disp:

mov ax,0xb800

mov es,ax

mov ah,0x07

L4:

pop ax

or al,0x30

mov ah,0x07

STOSW

loop disp

ret

start:

mov cx,17

lea si,array1

mov ax,0xb800

mov es,ax

mov ah,0x07

L1:

LODSB

add sum,al

OR al,0x30

STOSW

CMP cx,1

JE next

mov es:[di],0x072B

add di,2

next:

loop L1

mov es:[di],0x073D

add di,2

mov al,sum

mov ah,0

push ax

push di

CALL printNum

pop ax

pop ax

ret

## Task 6:

Write an assembly language program to count and display the number of comma ‘,’ in a given sentence.

Note:

* Display the string on screen first and then read the characters from video memory not from string directly.
* You can use string instructions SCASB or CMPSB for the comparison of ‘,’.

*Pattern of display screen should be.*

|  |
| --- |
| Sample String:  Commas are used to separate a sentence's elements, to connect independent clauses, to avoid confusion, and much more. Here, we offer a brief, basic guide to using commascorrectly.  OUTPUT:  Number of comma: 5 |

SOL:

Jmp start

str db 'Commas are used to separate a sentences elements, to connect independent clauses, to avoid confusion, and much more. Here, we offer a brief, basic guide to using commas correctly.',0

msg db 'Number of commas = ',0

write:

mov bp,sp

mov di,[bp+2]

mov si,[bp+4]

mov ax,0xb800

mov es,ax

mov ah,0x07

L1:

LODSB

STOSW

CMP al,0

JNE L1

ret

read:

mov bp,sp

mov di,[bp+2]

mov ax,0xb800

mov es,ax

mov ah,0x07

mov al,','

mov dx,[bp+4]

L2:

SCASW

JNE next

INC cx

next:

CMP di,dx

JNE L2

ret

start:

lea si,str

mov di,0

push si

push di

CALL write

pop ax

pop ax

push di

mov di,0

push di

CALL read

pop di

pop di

add di,320

lea si,msg

PUSH si

Push di

CALL write

pop ax

pop ax

mov ax,cx

mov ah,0x07

or al,0x30

mov es:[di],ax

ret