**Computer Organization and Assembly Language**

|  |  |
| --- | --- |
| **Lab 11** | |
| **Topic** | 1. Video memory |

**PART 1**

**VIDEO MEMORY**

***Console Display:***

Note : Each cell represents a word (2 byte).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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.

; character attribute is 8 bit value,

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

LET AX have 16 bits with character ‘A’ as a value byte and Brown background with white foreground color.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Blinking of the foreground color | Attribute byte | | | | | | | Value byte | | | | | | | |
| Background | | | Foreground | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

; hex bin color

; 0 0000 black

; 1 0001 blue

; 2 0010 green

; 3 0011 cyan possible background colors

; 4 0100 red

; 5 0101 magenta

; 6 0110 brown

; 7 0111 light gray possible foreground color

; 8 1000 dark gray

; 9 1001 light blue

; a 1010 light green

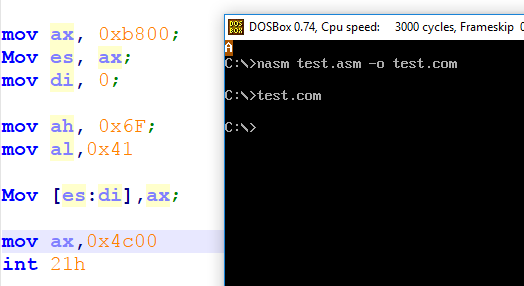
; b 1011 light cyan

; c 1100 light red

; d 1101 light magenta

; e 1110 yellow

; f 1111 white



**Display string on screen**

[org 0x100]

jmp start

str1 db 'HELLO WORLD'

start:

mov ax, 0xb800; ;segment address from where video memory starts.

Mov es, ax;

mov di, 0; ;location on screen where we want to display our character.

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

mov si, str1;

mov ah, 0x9A; ; Attribute byte for the character to be displayed.

l1:

Mov al, [si]; ;reading the characters in al.

Inc si ; pointing to next character in string

Mov [es:di],ax; ; printing message on the screen, whole register of size word is written at

Add di,2; ;physical memory location -> 0xB8001

loop l1

mov ax,0x4c00

int 21h

**Display code which writes and clears the string from screen.**

*USE CTRL+F11 to reduce cycles / sec or CTRL+F12 to increase the speed of dosbox.*

[org 0x100]

jmp start

str1 db 'HELLO WORLD'

start:

mov ax, 0xb800;

Mov es, ax;

mov di, 500;

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

mov 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;

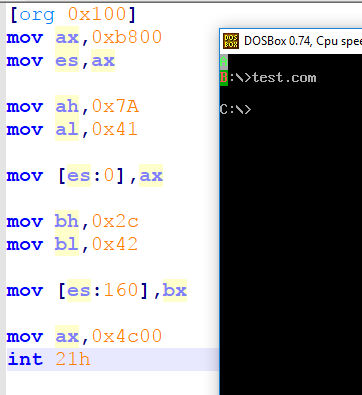
loop l2

mov ax,0x4c00

int 21h

**For example:**

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



**Display Number Function**

[org 0x100]

jmp start

printnum:

push bp

mov bp, sp

mov ax,[bp+4]

mov cx,0

mov bx, 10

Divide:

; use base 10 for division

mov dx,0

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

push dx

inc cx

cmp ax,0

jnz Divide

Display:

mov ax, 0xb800;

Mov es, ax;

mov di, 0;

l1:

pop ax

add al,0x30

mov ah, 0x07; ; Attribute byte, use any number

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

Add di,2;

loop l1

pop bp

ret 2

start:

mov ax, 1234

push ax ; place number on stack

call printnum ; call the printnum subroutine

; mov ax,0x1A34

; Mov [es:di],ax;

mov ax,0x4c00

int 21h

**PART 2**

## Task 1:

Write an assembly language program to display any string of known size with the **blinking** font color white and background of color magenta.

## Task 2:

Write an assembly language program to display your first name and last name on the screen as follows. Use loops to calculate the next screen location to display the next character.

First name on first column and second name on third column.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **H** |  | **A** |  |  |  |  |  |  |  |
| **A** |  | **L** |  |  |  |  |  |  |  |
| **S** |  | **I** |  |  |  |  |  |  |  |
| **S** |  |  |  |  |  |  |  |  |  |
| **A** |  |  |  |  |  |  |  |  |  |
| **N** |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

## Task 3:

Write an assembly language program to read the string and then display the Lower Case letters only on screen without showing space characters. Use black font color with white background color.

Let string:

*‘tHe* Quick BrOwn fox JumpS over The LaZy DOg*.’,0*

## Task 4:

Write an assembly language program to:

Let:

* string1 db ‘Computer Organization and’,0
* string2 db ‘Assembly language’,0
* string3 db 0

1. write one string at row 1 and other string at row3.
2. Concatenate the strings and display at row 5.

## 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 1,2,3,4,5,6,7,8,90,8,7,6,5,4,3,2,1

Observe the color scheme.

*Pattern of display should be*

|  |
| --- |
| Sample String:  1+2+3+4+5+6+7+8+90+8+7+6+5+4+3+2+1 = 162 |