**Computer Organization and Assembly Language**

|  |  |
| --- | --- |
| **Lab 13** | |
| **Topic** | 1. Video Memory 2. Interrupts 3. Interrupts hooking |

**PART 1**

**Example (Taking Input from User)**

MOV AX, 0xB800

MOV ES, AX ; Initializing ES with video memory address

MOV AH, 0 ; service number

INT 0x16 ; calling interrupt number 16h

; When you call interrupt 16h with service number 0, processor waits for keyboard input. When a key is pressed, its ASCII value is stored in AL register.

; Printing the character on screen.

MOV DI, 0 ; screen location di=0 top left.

MOV AH,07h ; attribute byte

STOSW ; displaying on screen

**Example (Terminating input from User)**

MOV AX, 0xB800

MOV ES, AX ; Initializing ES with video memory address

XOR DI, DI ; screen location di=0 top left.

again:

MOV AH, 0 ; service number

INT 0x16 ; calling interrupt number 16h

; When you call interrupt 16h with service number 0, processor waits for keyboard input. When a key is pressed, its ASCII value is stored in AL register.

; Printing the character on screen.

MOV AH,07h ; attribute byte

STOSW ; displaying on screen

cmp al, 0x1b

jne again

**Printing Character Using Interrupt Example**

start:  mov ah, 0

int 16h ; wait for any key....

cmp al, 27  ; if key is 'esc' then exit.

je stop

;al contains ascii of pressed key

mov ah, 0Eh ; print it.

int 10h

jmp start

stop:

.exit

**Printing String Using Interrupt Example**

mov al, 1;update curser after every character printing

mov bh, 0;page 0, means first page

mov bl, 00111011b;attribules

mov cx, 15 ; message size

mov dl, 10 ;row

mov dh, 7 ;col

push cs

pop es

lea bp, msg1

mov ah, 13h

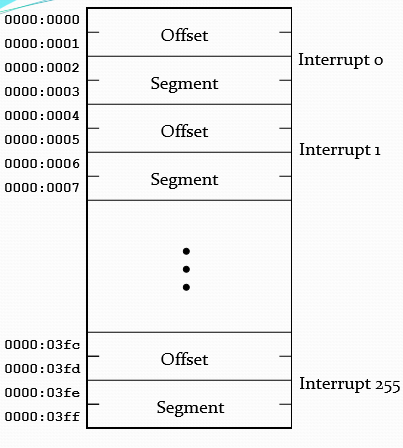
int 10h

ret

msg1 db " hello, world! "

**Interrupt vector table-address mapping**

* Offset: n\*4 ; offset address of nth interrupt
* Segment: n\*4+2 ; base address of nth interrupt



**Interrupt zero: INT 0 ISR hooking-divide by zero**

|  |  |
| --- | --- |
| MOV AX, 100  DIV BL  Ret | jmp start  message db 'Your message for divide overflow',0;  start:  xor di, di;  mov es, di  mov ax, offset isr0;  mov es:[di],ax;  mov es:[di+2], cs;  mov ax, 100  div bl  ret  isr0:  mov ax, 0xb800  mov es, ax;  lea si, message;  mov ah,7  nextchar:  lodsb;  cmp al, 0  je skip  stosw  jmp nextchar  skip:  iret |

**Another Interrupt hooking Example**

XOR DI, DI;

MOV ES, DI

LEA AX, ISR0;

MOV ES:[16h\*4],AX;

MOV ES:[16h\*4+2], CS;

mov ah,0;

int 16h;

MOV AX, 4C00H

INT 21H

ISR0:

MOV AX, 0XB800

MOV ES, AX;

MOV ES:[0], 0X0741;

IRET

**PART 2**

## Task 1:

Write an assembly language program, which prints given string on screen in following order.

1. Print every character with some delay
2. Clear screen after printing complete string
3. Repeat step 1 and 2 in infinite loop

SOL:

Jmp start

str db 'Computer Science',0

start:

mov cx,5

mov dx,0xffff

mov ax,0xb800

mov es,ax

L1:

mov di,0

lea si,str

L2:

mov ah,0x07

LODSB

CMP al,0

JE clr

STOSW

INC bp

mov ah,86h

INT 15h

JMP L2

clr:

mov di,0

mov ax,0x0720

L3:

STOSW

dec bp

JNE L3

Jmp L1

## Task 2:

Print capital character ‘L’ on screen of the emulator using stars character ‘\*’. The height of the character and total numbers of asterisk (\*) is based on the given inputs by the user. The information required to print the character is:

* Size of the character (Total Number of stars used to print the character ‘L’)
* Percentage of height  (Percentage of stars used in height of character ‘L’)

For example if Size is 10, percentage of height is 50%. The remaining will be the percentage of width i.e. 50% then the printed character is as follows:

\*

\*

\*

\*

\*

\* \* \* \* \*

If Size is 20, percentage of height is 20% then

\*

\*

\*

\*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

Note: When converting the percentage into required number of stars then use the absolute values only.

SOL:

Jmp start

size dw 0

leng dw 0

enterNum:

mov ah,0

mov bp,sp

mov di,[bp+2]

L1:

INT 0x16

CMP al,13

JE ex

mov bl,al

sub bl,0x30

mov ax,[di]

mov cl,10

mul cl

add ax,bx

mov [di],ax

Jmp L1

ex:

ret

start:

lea bx,size

PUSH bx

CALL enterNum ; Enter size first press enter

pop bx

lea bx,leng

PUSH bx

CALL enterNum ; Enter the percentage of length and press enter

pop bx

mov ax,leng

mov bx,size

mul bx

mov bx,100

div bx

mov bx,0xb800

mov es,bx

mov cx,ax

mov bp,size

sub bp,cx

mov di,0

mov ah,0x07

mov al,42

l:

mov es:[di],ax

add di,160

loop l

mov cx,bp

w:

STOSW

loop w

ret

## Task 3:

Take two 5 hex digit numbers from user through keyboard interrupt. And display sum by extended addition of those two number on next line. Print the sum on screen.

SOL:

## Jmp start

## num1 dd 0

## num2 dd 0

## sum dd 0

## temp db 8

## 

## userInput:

## mov bp,sp

## mov si,[bp+4]

## mov di,[bp+2]

## mov bx,[si]

## mov dx,[si+2]

## mov ax,0xb800

## mov es,ax

## mov cx,5

## L1:

## mov ah,0

## INT 0x16

## CMP al,13

## JE ex

## mov ah,0x07

## STOSW

## CMP al,0x39

## JA hexa

## sub al,0x30

## Jmp next

## hexa:

## sub al,87 ;to convert the small alphabets (a-f) to the value.

## next:

## shl al,4

## mov temp,4

## L2:

## shl al,1

## rcl bx,1

## rcl dx,1

## dec temp

## JNZ L2

## loop L1

## ex:

## mov [si],bx

## mov [si+2],dx

## ret

## printNum:

## mov bp,sp

## mov si,[bp+4]

## mov di,[bp+2]

## mov cx,0

## mov dx,[si+2]

## mov ax,[si]

## separate:

## mov temp,4

## mov bx,0

## rotate:

## shr dx,1

## rcr ax,1

## rcr bl,1

## dec temp

## JNZ rotate

## shr bl,4

## push bx

## inc cx

## CMP ax,0

## JNZ separate

## CMP dx,0

## JNZ separate

## mov ax,0xb800

## mov es,ax

## disp:

## pop ax

## CMP ax,9

## JA hex

## xor al,0x30

## Jmp ne

## hex:

## add al,0x37

## ne:

## mov ah,0x07

## STOSW

## loop disp

## ret

## 

## start:

## lea si,num1

## PUSH si

## mov di,0

## push di

## CALL userInput

## pop si

## pop si

## lea si,num2

## PUSH si

## mov di,160

## push di

## CALL userInput

## pop si

## pop si

## mov ax,[num1]

## mov dx,[num1+2]

## add ax,[num2]

## adc dx,[num2+2]

## 

## mov [sum],ax

## mov [sum+2],dx

## lea bx,sum

## push bx

## mov di,320

## push di

## CALL printNum

## pop ax

## pop ax

## Task 4:

Hook an interrupt for arithmetic overflow (INT 4). Write an ISR which display a message “OVERFLOW OCCURRED” on screen. To test your subroutine add these two numbers. 0x7FFF + 0x0001

SOL:

Jmp start

msg db 'Overflow occured',0;

ISR4:

mov ax,0xb800

mov es,ax

lea si,msg

mov ah,0x07

mov di,0

dis:

LODSB

CMP al,0

JE exit

STOSW

Jmp dis

exit:

iret

start:

xor di,di

mov es,di

mov bx,offset ISR4

mov es:[4\*4],bx

mov es:[4\*4+2],cs

mov ax,0x7FFF

mov dx,0x01

add ax,dx

into ;interrupt of overflow is not called until we write the instruction of 'into'