**Pontificia Universidad Católica Madre y Maestra**

****

**ARQUITECTURA COMPUTACIONAL**

ST-ISC-359-T-001

**Presentado por:**

Cesar Darinel Ortiz 2008-1175

**Presentado al profesor:**

ÁLVARO A. REYES P.

**Tema:**

Programa screen saver assembly

**Fecha de entrega:**

Jueves, 08 de Noviembre del 2018

; Autor: Cesar Darinel Ortiz

; Tarea: screensaver laboratorio

; Fecha Entrega: 08/11/2018

.model small

.stack 556

.data

.386 ; agregar para llamadas mas largas

;========================Variables declaradas aquí===========================

caracter db 1 dup ('\*') ;símbolo a mostrar

pos\_x db 1 dup(1) ;posición virtual en X

pos\_y db 1 dup(1) ;posición virtual en y

mov\_x db 1 dup(1) ;monto a sumar o restar en x

mov\_y db 1 dup(1) ;monto a sumar o restar en y

cambiocolor db 1 dup(1) ;indicador(booleano) para saber si debo cambiar el color

ultimoleido db 1 dup(' ') ; guardo el ultimo carácter para regresarlo

;============================================================================

.code

main:

    mov ax,@data

    mov ds,ax

    mov es, ax ;set segment register

and sp, not 3 ;align stack to avoid AC fault

    mov pos\_x,0 ; inicializo x

    mov pos\_y,0             ; inicializo y

    mov mov\_x,1             ; inicializo monto x

    mov mov\_y,1             ; inicializo monto x

    mov cambiocolor,0       ; muestro que no cambiare color

    mov ax,0b800h           ; para utilizar la memoria

    mov es,ax               ;segmento dir. de mem de video.

    xor ax,ax               ;ax en cero

    xor bx,bx               ;bx en cero

    mov al,es:[bx]          ;busco la posición 0

    mov ultimoleido,al      ;guardo lo para regresar ese valor

;==========ciclo\_principal mantener código ==================================

ciclo\_principal:

    call pinta              ; llamo a método que pinta

    call correr             ; método con validaciones

    call pausa              ; método pausa

    call finalizo           ; método que finaliza

    jmp ciclo\_principal    ; mantenemos el ciclo infinito

;========= ==Filanizo ==================================

finalizo:

    push ax bx cx dx        ; guardo en cache la data para no molestar otros programas

    mov ah,0Bh ;verifico si insertaron algo por teclado

    int 21h ;llamada al SO

    cmp al,0ffh ;comparo al con alguna tecla pulsada

    je fin ;salimos del programa si digitan algo

    pop dx cx bx ax         ;retornamos las variables del cache

    ret

;=== =============imprimo en pantalla==================================

pinta:

    call limpia             ;retorno la variable que guarde

    call optener\_posi       ;obtengo la nueva posición a pintar

    call guardaValor        ;guardo el valor que voy a cambiar

    call cambiar\_color      ; cambio color

    mov al,caracter         ; agrego la variable a pintar

    mov es:[bx],ax          ; pinto variable con el color tomado

    ret

;====================================limpio ==================================

limpia:

    push ax cx dx           ; guardo en cache la data para no molestar otros programas

    mov ah,7                ; guardo el color base

    mov al,ultimoleido      ; guardo el carácter a imprimir

    mov es:[bx],ax          ; retorno el carácter

    pop dx cx ax            ; retornamos las variables del cache

    ret

;==========guarda Valor de la pantalla==================================

guardaValor:

    push ax cx dx           ; guardo en cache la data para no molestar otros programas

    mov al,es:[bx]          ; guardo en al el valor que quiero retener

    mov ultimoleido,al      ; guardo en la variable para no perder dato

    pop dx cx ax            ; retornamos las variables del cache

    ret

;============cambio color a presentar en pantalla =================

; 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

cambiar\_color:

    push bx cx dx           ; guardo en cache la data para no molestar otros programas

    cmp cambiocolor,0       ; comparo con el indicador

    je cambiar\_color\_fin    ; si es cero no hago nada y finalizo

    inc ah                  ; incremento ah

    mov cambiocolor,0       ; apago el indicador

    cmp ah,250              ; comparo el valor con 250

    jne cambiar\_color\_fin   ; si no es 250 puedo finalizar

    mov ah,0                ; reinicio en cero

    cambiar\_color\_fin:      ; finalizo

    pop dx cx bx            ; retornamos las variables del cache

    ret

;===============imprimo en pantalla==================================

correr:

    push ax cx bx dx        ; guardo en cache la data para no molestar otros programas

    mov al, pos\_x           ; guardo los valores de X en al para trabajar

    mov ah, pos\_y           ; guardo los valores de Y en al para trabajar

    ;eje X si es mayor a 160 marco con -1

    cmp al,160              ; consulto con 160

    je finll                ; si es igual reinicio

    jmp next1               ; si no es igual consulto siguiente pregunta

    finll:

    mov cambiocolor,1       ; enciendo el indicador

    mov mov\_x,-1            ; cambio el sentido

    ;eje X si es menor a 0 marco con 1

    next1:

    cmp al,0                ; comparo con cero

    je finll1               ; si es cero reinicio

    jmp next2               ; salto a la siguiente pregunta

    finll1:

    mov cambiocolor,1       ; enciendo el indicador

    mov mov\_x,1             ; cambio el sentido

    next2:

    ;eje Y si es mayor a 160 marco con -1 (segundo cuadrante)

    cmp ah,25               ;comparo con el 25

    je finll2               ; si es 25 reinicio

    jmp next3               ; si no es igual consulto siguiente pregunta

    finll2:

    mov cambiocolor,1       ; enciendo el indicador

    mov mov\_y,-1            ; cambio el sentido

    ;eje Y si es menor a 0 marco con 1

    next3:

    cmp ah,0                ; comparo con cero

    je finll3               ; si es cero reinicio

    jmp fincorrer           ; si no es igual consulto siguiente pregunta

    finll3:

    mov cambiocolor,1       ; enciendo el indicador

    mov mov\_y,1             ; cambio el sentido

    fincorrer:

    call inc\_x              ; incremento X

    call inc\_y              ; incremento Y

    mov pos\_x, al           ; guardo el nuevo X

    mov pos\_y, ah           ; guardo el nuevo Y

    pop dx bx cx ax         ;retornamos las variables del cache

    ret

;===========cambio en X==================================

inc\_x:

add al,mov\_x                ; incremento o disminuir al

ret

;===================cambio en Y==================================

inc\_y:

add ah,mov\_y                ; incremento o disminuir ah

ret

;====================Retorno la posición en el arreglo================

optener\_posi:

    push ax cx dx           ; guardo en cache la data para no molestar otros programas

    xor ax,ax               ; reinicio ax

    xor cx,cx               ; reinicio cx

    xor bx,bx               ; reinicio bx

    mov al, pos\_x           ; guardo x a utilizar

    mov ah, pos\_y           ; guardo y a utilizar

    cmp ah,0                ; comparo Y con cero

    jne continua            ; sino es cero continuo

    mov bl,al               ; si es cero le agrego a bl la posición de X

jmp fin\_optener\_posi    ; fin del método

    continua:

    mov cl,ah               ; fin del ciclo es ah

    label1:

    add bx,160              ; añado la posición

    loop label1             ; repito ah veces

    xor ax,ax               ; reinicio ax

    mov al, pos\_x           ; guardo el valor X a al

    add bx,ax               ; sumo ese valor

    mov al, pos\_y           ; guardo el valor Y

    add bx,ax               ; sumo ese valor

    fin\_optener\_posi:

    pop dx cx ax                ;retornamos las variables del cache

    ret

;================imprimo en pantalla==================================

pausa:

    push ax bx cx dx        ; guardo en cache la data para no molestar otros programas

    mov al, 0               ;limpio al

    mov ah, 86h             ;llamo para pausa

    mov cx, 1               ;valores de tiempo de pausa

    mov dx, 2               ;valores de tiempo de pausa

    int 15h                 ;llamada al SO

    pop dx cx bx ax         ;retornamos las variables del cache

    ret

;====================================FIN==================================

fin:

;============================================================================

.exit

end main