MANUAL TÉCNICO

PROYECTO FINAL

201602719
CHRISTOFER WILLIAM BORRAYO LOPEZ
ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1

Contenido

Main.asm

Se incluyeron las importaciones de los demás archivos para mantener un orden y coherencia dentro del código.

```
org 100h
%include "data.asm"
%include "bss.asm"
%include "login.asm"
%include "user.asm"
%include "Juego.asm"
%include "admin.asm"
%include "grafica.asm"
%include "ordenamiento.asm"
%include "getData.asm"
%include "errors.asm"
%include "extra.asm"
```

Todos los menús que aparecerán en el programa poseen la misma estructura que el siguiente

```
%macro ingresar 0
    mov ah, 09h
    mov dx, l_1
    int 21h

    mov dx, l_3
    int 21h
    resetUsr
    getDatoUsr

    mov ah, 09h
    mov dx, l_4
    int 21h
    resetPsw
    getDatoPsw

    cmp cl, 1
    je %%errorPsw
    leerUsers
    call %%fin
    %%errorPsw:
        salto
        mov ah, 09h
        mov dx, er_Psw
        int 21h
        salto
    %%fin:
%endmacro
```

Data.asm

Se incluyeron todos los textos que aparecerán en el programa, así también como vectores, matrices, datos simples, arreglos para formar sprites del juego, rutas, etc.

```
section .data
;-------------------------------------------------------------------------
;-------------------------------EXTRA-------------------------------------
;-------------------------------------------------------------------------
    pts db '**********************************', 10, 13, '$'
    ads db 'SALIENDO DEL SISTEMA....ADIOS...', 10, 13, '$'
    slt db 0dh, 0ah, '$'

;-------------------------------------------------------------------------
;-------------------------------MAINs-------------------------------------
;-------------------------------------------------------------------------
    m_1 db 'BIENVENIDO - SELECCIONE UNA OPCION', 10, 13, '$'
    m_2 db '    1 - Ingresar', 10, 13, '$'
    m_3 db '    2 - Registrar', 10, 13, '$'
    m_4 db '    3 - Salir', 10, 13, '$'

;-------------------------------------------------------------------------
;-------------------------LOGIN/REGISTRAR---------------------------------
;-------------------------------------------------------------------------
    l_1 db '-------- INGRESO --------', 10, 13, '$'
    l_2 db '-------- REGISTRO --------', 10, 13, '$'
    l_3 db 'Nombre de usuario:', 10, 13, '$'
    l_4 db 'Contrasenha:', 10, 13, '$'
    l_5 db 'USUARIO REGISTRADO CON EXITO', 10, 13, '$'

;-------------------------------------------------------------------------
;-------------------------ENCABEZADOS-------------------------------------
;-------------------------------------------------------------------------
    e_1 db 'UNIVERSIDAD DE SAN CARLOS DE GUATEMALA ', 10, 13, '$'
    e_2 db 'FACULTAD DE INGENIERIA ', 10, 13, '$'
    e_3 db 'CIENCIAS Y SISTEMAS ', 10, 13, '$'
    e_4 db 'ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1 ', 10, 13, '$'
    e_5 db 'NOMBRE: CHRISTOFER WILLIAM BORRAYO LOPEZ ', 10, 13, '$'
    e_6 db 'CARNET: 201602719 ', 10, 13, '$'
    e_7 db 'SECCION: A ', 10, 13, '$'

;-------------------------------------------------------------------------
```

```
;-------------------------------------------------------------------------
;-----------------------------SPRITES-------------------------------------
;-------------------------------------------------------------------------
;-----------------------------CARROS--------------------------------------
    CoordX_car          dw 155
    CoordY_car          dw 165
    car_f1  DB  27,27,27,12,12,12,27,27,27
    car_f2  DB  27,27,12,15,15,15,12,27,27
    car_f3  DB  17,12,12,15,15,15,12,12,17
    car_f4  DB  19, 4,12,15,15,15,12, 4,19
    car_f5  DB  17,12, 8,17, 8, 8, 8,12,17
    car_f6  DB  27, 4,19,19,18, 8, 8, 4,27
    car_f7  DB  27, 4, 8,17,17,18, 8, 4,27
    car_f8  DB  27, 4,12,15,15,15,12, 4,27
    car_f9  DB  27, 4,12,30,31,31,12, 4,27
    car_f10 DB  27, 4,12,30,30,15,12, 4,27
    car_f11 DB  17, 4,12,15,15,15,12, 4,17
    car_f12 DB  19,12,31,31,15,15,15,12,19
    car_f13 DB  17,12,31,30,30,30,30,12,17
    car_f14 DB  27, 4,14,12,12,12,14, 4,27

;-------------------------ESTRELLA BUENA----------------------------
    pre_pts     dw 0                    ;PUNTOS QUE DA
    pre_tmp     dw 0                    ;TIEMPO PARA QUE APAREZCA
    pre_tmpAnt  dw 0                    ;TIEMPO QUE HA PASADO

    pre_CoordY1 dw 0
    pre_CoordY2 dw 0
    pre_CoordY3 dw 0
    pre_CoordY4 dw 0
    pre_CoordY5 dw 0

    pre_CoordX1 dw 50
    pre_CoordX2 dw 250
    pre_CoordX3 dw 100
    pre_CoordX4 dw 200
```

```
t_txt db 42 dup('$'), '$'
;t_txt db '1       1234567         0           0000 ', 10, 13, '$'
;          12345678901234567890123456789012345678901234567890
;          0        1         2         3         4
tcant dw 0, '$'

;usuario n 0000 tmps
;1234567 8 9012 3456
ar_desor db 120 dup('$'), '$'
ar_liaux db 12 dup('$'), '$'
ar_linea db 12 dup('$'), '$'
ar_auxil db 4 dup('$'), '$'
;0123 -- NUMEROS
```

Bss.asm

Se incluyó (aunque solo fuese un tipo de dato) para no confundir su función. Es el handler al momento de escribir archivos.

```
section .bss
    filehndl resw 01h
```

User.asm

Se puede observar la misma estructura que en el menú de **“Main.asm”**

```
%macro leerUsers 0
    salto
    resetTexto

    mov ah,3dh          ; abriendo un archivo
    mov al,0            ; indicando que lo estoy abriendo en modo lectura
    mov dx,f_user       ;especifico la ruta del archivo
    int 21h
    jc %%fin
    mov bx,ax           ; handle del archivo lo copio a bx

    mov ah,3fh          ;funcion para leer archivo
    mov dx,texto        ; indico la variable en donde guardare lo leido
    mov cx,0ffh         ; numero de bytes a leer
    int 21h
    jc %%fin
    cmp ax,0            ; si ax = 0 significa que EOF
    jz %%fin

    compararUsr
    cmp cl, 0           ;USUARIO INCORRECTO
    je %%usr_inc
    cmp cl, 1           ;CONTRASEÑA INCORRECTA
    je %%psw_inc
    cmp cl, 2           ;2 USUARIO, 3 ADMIN
    je %%us_user
    %%us_admin:
        admin_Main
        jmp %%fin
    %%us_user:
        mov ax, 0
        mov [nvM], ax               ;NIVEL MAXIMO A 0
        resetPth
        user_Main
```

Para registrar usuarios, se hace uso de una matriz llamada **“texto[ ]”** en la cual se almacena los datos del archivo **“usuarios.usr”** y luego se agregan los datos del nuevo usuario.

Se procede a guardar por medio de la interrupción 3ch, también se agrega un carácter de control **‘$’** para su futura lectura o escritura.

```
%macro regiUser 0
    mov si, 0
    mov bl, '$'
    %%ciclo:
        cmp texto[si], bl
        je %%llenado_enter
        inc si
        jmp %%ciclo

    %%llenado_enter:
        mov psw[4], bl
        mov bl, 10
        mov usr[7], bl
        mov texto[si], bl
        inc si
        mov bl, 13
        mov texto[si], bl

    %%escritura:
        mov  ah, 3ch
        mov  cx, 0
        mov  dx,  f_user
        int  21h

        mov [filehndl],ax

        mov  ah, 40h
        mov  bx, [filehndl]
        mov  cx, si
        mov  dx, texto
        int  21h

        mov  ah, 40h
        mov  bx, [filehndl]
        mov  cx, 8  ;STRING LENGTH.
        mov  dx, usr
```

Para cargar el archivo de juego se resetea la información de la ruta para que se pueda ingresar una totalmente nueva, se obtiene la ruta y se procede a leer el archivo. Luego de esto se procede a cargar el juego y cargar los datos fundamentales como las coordenadas del carro, el color de fondo, el mensaje de cabecera y demás.

```
;CARGA DE ARCHIVO
%macro user_Cargar 0
    mov ah, 09h
    mov dx, j_1
    int 21h

    resetPth
    getDatoPth
    leerNivel
%endmacro
```

```
%macro guardarDatosJuego 0
    resetTexto

    mov ah,3dh              ; abriendo un archivo
    mov al,0                ; indicando que lo estoy abriendo en modo lectura
    mov dx,f_pts            ;especifico la ruta del archivo
    int 21h
    jc %%fin
    mov bx,ax               ; handle del archivo lo copio a bx

    mov ah,3fh              ;funcion para leer archivo
    mov dx,texto            ; indico la variable en donde guardare lo leido
    mov cx,0ffh             ; numero de bytes a leer
    int 21h
    jc %%fin
    cmp ax,0                ; si ax = 0 significa que EOF
    jz %%fin


    mov si, 0
    mov bl, '$'
    %%cicloEscritura:
        %%ciclo:
            cmp texto[si], bl
            je %%llenado_enter
            inc si
            jmp %%ciclo
        %%llenado_enter:
            mov bl, 10
            mov texto[si], bl
            inc si
            mov di, 0
        %%escritura_user:
            mov dl, usr[di]
            mov texto[si], dl
            inc si
```

Juego.asm

Se utiliza la interrupción 10h con 13h para cargar el modo video, seguidamente se llama al macro **“dibujarTodo”** el cual se encarga de pintar la pantalla por primera vez.

Las siguientes veces la pantalla es refrescada por el macro **“set_Items”**

```
%macro load_Juego 0
    setData
    ;iniciar el modo video., 13h
    mov al,13h
    xor ah,ah
    int 10h

    ;posicionar directamente a la memoria de video
    mov ax,0A000H
    mov es,ax
    xor di,di

    dibujarTodo

    %%mainLoop:
        setHeader
        set_Items

        mov ax, [perdido]
        cmp ax, 0
        je %%finProg

        mov ax, [segAct]
        mov bx, [segMax]
        cmp ax, bx
        jae %%siguienteNivel
        ;====================delay============================

        Delay
```

```
    ;==================delay=========================

    Delay

    ;============leer el buffer de mi teclado =========
    HasKey      ; hay tecla?
    jz %%mainLoop

    GetCh           ; si hay una tecla presiona, cual tecla es la que se presiono

    cmp al,20h          ; es espacio ? ,si sí,  se sale
    jne %%Ps_           ;sino comprobar movimientos
    jmp %%finProg
%%Ps_:
    cmp al, 1bh ;ESC
    je %%Pausa


%%Mov1:
    cmp al, 'd'
    jne %%Mov2

    ; si llega aqui es la tecla d
    mov ax,[CoordX_car]
    cmp ax, 290
    jae %%mainLoop
    add ax, 10
    mov [CoordX_car],ax
    dibujarTodo
    jmp %%mainLoop

%%Mov2:
    cmp al, 'a'
    jne %%mainLoop
    mov ax,[CoordX_car]
```

En este macro, la información del juego es puesta para su funcionamiento inicial, esta incluye las coordenadas de las estrellas, coordenadas iniciales del carro, puesta de tiempo a cero, suma de niveles, etc. Este macro es reutilizado al subir de nivel.

```
%macro setData 0
   mov ax, 0
   mov [t_mi], ax              ;MICROSEGUNDOS A 0
   mov [t_s], ax               ;SEGUNDOS A 0
   mov [t_m], ax               ;MINUTOS A 0
   mov [obs_tmpAnt], ax        ;TIEMPO ANTERIOR DE OBSTACULOS A 0
   mov [pre_tmpAnt], ax        ;TIEMPO ANTERIOR DE PREMIOS A 0

   mov [segAct], ax            ;TIEMPO ACTUAL A 0
   mov [segMax], ax            ;TIEMPO ACTUAL A 0

                               ;TODOS LOS PREMIOS EMPEZARAN EN LA PARTE DE ARRIBA
   mov [pre_CoordY1], ax
   mov [pre_CoordY2], ax
   mov [pre_CoordY3], ax
   mov [pre_CoordY4], ax
   mov [pre_CoordY5], ax
                               ;TODOS LOS OBSTACULOS EMPEZARAN EN LA PARTE DE ARRIBA
   mov [obs_CoordY1], ax
   mov [obs_CoordY2], ax
   mov [obs_CoordY3], ax
   mov [obs_CoordY4], ax
   mov [obs_CoordY5], ax

   mov ah, 3ah
   mov [dosP], ah              ;DOS PUNTOS A ':'

   mov ax, 155
   mov [CoordX_car], ax        ;CENTRAR EL CARRO EN X
   mov ax, 165
   mov [CoordY_car], ax        ;CENTRAR EL CARRO EN Y

   mov ax, [nv0]               ;INCREMENTAR NIVEL POR 1
   inc ax
   mov [nv0], ax
```

```
;-----------------------
%macro dibujarTodo 0
    setFondo
    DibujarCarro
    DibujarEstrellas
%endmacro
;-----------------------
```

```
%macro setHeader 0
    imprimir usr, 01H   ;NOMBRE USUARIO
    imprimir nv_, 0BH   ;Nivel
    imprimirNumeros nv0, 0DH    ;Nivel
    ImprimirPuntaje ptn, 16H    ;Numero nivel

    setTiempo
%endmacro
```

El tiempo es dividido en tres partes, microsegundos, segundos y minutos, estos son aumentados cada delay del programa.

```
%macro setTiempo 0
    mov ax, [segs]
    inc ax
    mov [segs], ax
    mov ax, [t_mi]
    inc ax
    mov [t_mi], ax
    cmp ax, 100
    jae %%mas_seg
    jmp %%fin
    %%mas_seg:
        mov ax, 0
        mov [t_mi], ax

        mov ax, [segAct]
        inc ax
        mov [segAct], ax

        mov ax, [t_s]
        inc ax
        mov [t_s], ax
        cmp ax, 60
        jae %%mas_min
        jmp %%fin
    %%mas_min:
        mov ax, 0
        mov [t_s], ax

        mov ax, [t_m]
        inc ax
        mov [t_m], ax
    %%fin:
        imprimirNumeros t_mi, 24H
        imprimir dosP, 24H
        imprimirNumeros t_s, 21H
        imprimir dosP, 21H
        imprimirNumeros t_m, 1EH
%endmacro
```

Este es un ejemplo de cómo se gráfica, se inicia con las coordenadas ax como X y bx como Y, a partir de estas se empieza a pintar de izquierda a derecha y arriba abajo. Los colores, altura y ancho del recuadro a pintar se pueden modificar. Lo mismo pasa con el carro. Cabe recordar que para brincar una fila completamente es necesario agregarle a di la cantidad de 320

```
;-------------------------------------------------------------------------
%macro setFondo 0
    mov ax, 15
    mov bx, 25
    mov cx,bx    ;coord x
    shl cx,8
    shl bx,6
    add bx,cx    ; bx = 320
    add ax,bx    ; sumar x a y
    mov di, ax

    mov ax, 27       ;COLOR DEL
    mov bx, 170      ;ALTURA DEL FONDO
    %%loopRow:       ;LOOP PARA DIBUJAR HACIA ABAJO
        mov dx, 290 ;ANCHO DEL FONDO
        push di
        %%loopCol:  ;LOOP PARA DIBUJAR A LA IZQUIERDA
            mov [es:di], ax     ;PINTAR EN ES+DI CON COLOR AX
            inc di
            dec dx
            jnz %%loopCol
        pop di
        add di, 320 ;NUEVA FILA
        dec bx
        jnz %%loopRow
%endmacro
```

```
%macro setCarro 1
    push ax
    mov di, ax
    mov si, 0
    %%loop:
        mov ax, %1[si]
        mov [es:di], ax     ;PINTAR EN ES+DI CON COLOR AX
        inc di
        inc si
        cmp si, 8
        jb %%loop
    pop ax
%endmacro
```

```
%macro DibujarCarro 0
    mov ax, [CoordX_car]
    mov bx, [CoordY_car]
    mov cx,bx   ;coord x
    shl cx,8
    shl bx,6
    add bx,cx   ; bx = 320
    add ax,bx   ; sumar x a y
    mov di, ax

    setCarro car_f1
    add ax, 320
    setCarro car_f2
    add ax, 320
    setCarro car_f3
    add ax, 320
    setCarro car_f4
    add ax, 320
    setCarro car_f5
    add ax, 320
    setCarro car_f6
    add ax, 320
    setCarro car_f7
    add ax, 320
    setCarro car_f8
    add ax, 320
    setCarro car_f9
    add ax, 320
    setCarro car_f10
    add ax, 320
    setCarro car_f11
    add ax, 320
    setCarro car_f12
    add ax, 320
    setCarro car_f13
    add ax, 320
```

```
%macro DibujarEstrellas 0
    pre_Mover pre_CoordX1, pre_CoordY1
    pre_Mover pre_CoordX2, pre_CoordY2
    pre_Mover pre_CoordX3, pre_CoordY3
    pre_Mover pre_CoordX4, pre_CoordY4
    pre_Mover pre_CoordX5, pre_CoordY5

    obs_Mover obs_CoordX1, obs_CoordY1
    obs_Mover obs_CoordX2, obs_CoordY2
    obs_Mover obs_CoordX3, obs_CoordY3
    obs_Mover obs_CoordX4, obs_CoordY4
    obs_Mover obs_CoordX5, obs_CoordY5
%endmacro
```

La colisión de las estrellas y el auto se verifica por medio de sus coordenadas, si estas son intersecadas se toma como una colisión y se procede a sumar o restar puntos, dependiendo del objeto con el que colisiono.

```
;-------------------------------------------------------------------
%macro verCosilion 2    ;1-X  2-Y
    mov cl, 0           ;CONDICION INCIIAL, SI ES 0 NO HAY, 1 SI HAY COLISICION

    mov bx, [%2]        ;OBTENEMOS Y SUPERIOR DE LA ESTRELLA
    add bx, 9           ;OBTENEMOS Y INFERIOR DE LA ESTRELLA
    mov dx, [CoordY_car]    ;OBTENEMOS Y DEL CARRO
    cmp bx, dx
    jb %%fin            ;SI NO HA LLEGADO A Y DEL CARRO TERMINA

    mov bx, [%1]        ;OBTENEMOS X DERECHA DE LA ESTRELLA
    mov dx, [CoordX_car]    ;OBTENEMOS X DERECHA DEL CARRO

                        ;PRIMERO VERIFICAMOS SI LA ESQUINA DERECHA DE LA ESTRELLA
                        ;TOCA CON LA ESQUINA IZUIQERDA DEL CARRO
    add bx, 9
    cmp bx, dx
    jb %%fin
                        ;AHORA COMPARAMOS LA ESQUINA IZQUIERDA DE LA ESTRELLA
                        ;CON LA ESQUINA DERECHA DEL CARRO
    sub bx, 9
    add dx, 9
    cmp bx, dx
    ja %%fin

    mov cl, 1           ;HAY COLISION
    mov ax, 185         ;PARA ELIMINAR LA ESTRELLA DEL FONDO

    %%fin:
%endmacro
```

La velocidad de caída de las estrellas esta dada por este macro, el cual simplemente verifica los microsegundos y al cabo de cada cierto múltiplo de un número, este permitirá el movimiento de las estrellas

```
;-----------------------------
%macro set_Items 0
    mov ax, [t_mi]
    mov si, [vel]
    mov cx, 0
    %%ciclo:
    cmp cx, 100
    jae %%fin
    cmp ax, cx
    jne %%aumento
        setPremios
        setObstaculos
        jmp %%fin
    %%aumento:
        add cx, si
        jmp %%ciclo
    %%fin:
%endmacro
```

Admin.asm

Para el administrador se utiliza el mismo concepto que con los usuarios para poder abrir los archivos, la única diferencia es el uso de una matriz de 12x10 para almacenar los datos del top 10, la cual se va llenando con forme se lee el archivo de puntaje de todos los usuarios.

```
%macro admin_Pts 0
    resetTexto
    resetAlt

    mov ah,3dh          ; abriendo un archivo
    mov al,0            ; indicando que lo estoy abriendo en modo lectura
    mov dx,f_pts        ;especifico la ruta del archivo
    int 21h
    jc %%fin
    mov bx,ax           ; handle del archivo lo copio a bx

    mov ah,3fh          ;funcion para leer archivo
    mov dx,texto        ; indico la variable en donde guardare lo leido
    mov cx,0ffh         ; numero de bytes a leer
    int 21h
    jc %%fin
    cmp ax,0            ; si ax = 0 significa que EOF
    jz %%fin


    mov si, 0
    mov ax, 0
    mov cx, 0
    %%cicloLectura:
        mov dl, texto[si]
        cmp dl, '$'
        je %%fin_lectura
            mov di, 0
        %%lectura_user:     ;LEER USUARIO 7 ESPACIOS
            mov dl, texto[si]
            mov ar_linea[di], dl
            inc di
            inc si
            cmp di, 7
            jb %%lectura_user
        %%lectura_nivel:    ;LEER NIVEL 1 ESPACIO
```

```
mov si, 0
mov ax, 0
mov cx, 0
%%cicloLectura:
    mov dl, texto[si]
    cmp dl, '$'
    je %%fin_lectura
        mov di, 0
    %%lectura_user:     ;LEER USUARIO 7 ESPACIOS
        mov dl, texto[si]
        mov ar_linea[di], dl
        inc di
        inc si
        cmp di, 7
        jb %%lectura_user
    %%lectura_nivel:    ;LEER NIVEL 1 ESPACIO
        inc si
        mov dl, texto[si]
        mov ar_linea[di], dl
        inc di  ;sale con 8
        add si, 2
    %%lectura_puntaje:  ;LEER PUNTAJE 4 ESPACIOS
        mov dl, texto[si]
        mov ar_linea[di], dl
        inc di
        inc si
        cmp di, 12
        jb %%lectura_puntaje

    push si
    mov si, 8
    mov di, 0
    %%altura_Max:
        mov dl, ar_linea[si]
        mov dh, alt_Aux[di]
        cmp dl, dh
```

Este macro es simplemente para mostrar las opciones de ordenamiento disponibles

```
%macro ordenamiento 0
    setAlturaNum
    mov [alt_max], al

    load_grafica

    %%mensajeOrdenamiento:
        salto
        mov dx, g_1
        int 21h
        mov dx, g_2
        int 21h
        mov dx, g_3
        int 21h
        getChar
        cmp al, '1'
        je %%siguiente
        cmp al, '2'
        je %%siguiente
        errorOrd
        jmp %%mensajeOrdenamiento
    %%siguiente:
        mov cl, al
        sub cl, '0'
    %%mensajeVelocidad:
        salto
        mov dx, v_1
        int 21h
        getChar
        cmp al, '0'
        jb %%error_
        cmp al, '9'
        ja %%error_
        jmp %%siguiente2
        %%error_:
        errorVel
```

Para guardar el top 10, se utiliza la matriz de 12x10 y se extrae dato por dato, estos son puestos en el archivo y cada vez que se llene un dato entero, por ejemplo, usuario, se procede a colocar una coma y el siguiente dato, en este caso sería el nivel.

```
%macro guardarTop10Puntos 0
    mov ah,3dh          ; abriendo un archivo
    mov al,0            ; indicando que lo estoy abriendo en modo lectura
    mov dx,f_rep        ;especifico la ruta del archivo
    int 21h
    jc %%fin
    mov bx,ax           ; handle del archivo lo copio a bx

    resetTexto

    mov si, 0
    mov di, 0
    mov bl, 10              ;ENTER
    mov bh, ','             ;COMA
    %%ciclo_Titulo:
        mov dl,t_ep[di]
        mov texto[si], dl
        inc si
        inc di
        cmp si, 41
        jb %%ciclo_Titulo
    mov texto[si], bl

    inc si
    mov di, 0
    mov ch, 0
    mov al, 1
    mov ah, [tcant]
    %%ciclo_Data:
        cmp al, ah
        ja %%continuar

        inc ch
        cmp ch, 10
        jb %%menor_diez
            mov dl, 31h
            mov texto[si], dl
```

Grafica.asm

Para la grafica se creo un macro apartado para cargarla dado que se utilizará en diversas ocasiones esta función y para diferentes fines, por lo tanto, se decidió separa el código lo máximo posible

```
%macro inic_Graph 0
    ;iniciar el modo video., 13h
    mov al,13h
    xor ah,ah
    int 10h

    ;posicionar directamente a la memoria de video
    mov ax,0A000H
    mov es,ax
    xor di,di
%endmacro

%macro load_grafica 0
    inic_Graph

    imprimir t_p2, 0AH  ;Nivel
    setAncho
    setMargen
    dibujarBarras

    %%precionarBarra:
        xor ax, ax
        getChar
        cmp al, ' '
        jne %%precionarBarra

    mov ax,3h       ; funcion para el modo texto
    int 10h
%endmacro
```

Estos macros ayudan a encontrar la posición X y Y de cada barra en la gráfica, con una formula la cual depende del ancho de cada barra y su alto, los cuales son fácilmente obtenidos dividiendo el tamaño máximo de la pantalla por la cantidad de barras que habrán.

```
;*******************************************************
%macro setXY 0
    mov cx,bx    ;coord x
    shl cx,8
    shl bx,6
    add bx,cx    ; bx = 320
    add ax,bx    ; sumar x a y
    mov di, ax
%endmacro
%macro getXY 0          ;Se tiene ancho, alto, al final ax -> X   bx -> Y
                        ;SE SABE QUE Xin = 15   Yin = 25
                        ;PARA X = 15 + X*5 + (X-1)*ANCHO
    pop cx
    push cx

    xor ax, ax
    mov al, 4
    mul cx

    mov bx, ax
    xor ax, ax
    dec cx
    mov al, [ancho]
    mul cx

    add bx, 19
    add ax, bx
                        ;PARA Y = 140 + 25 - ALTURA
    xor dx, dx
    mov dl, [altura]
    mov bx, 140
    add bx, 30
    sub bx, dx

%endmacro
```

```
%macro setAncho 0        ;(290 - [(X+1)*5])/X
    mov cl, [tcant]      ; X
    mov ax, 290
    mov dx, 0
    div cl
    mov dx, ax


    inc cl               ; X+1
    mov al, 5
    mul cl               ;(X+1)*5
    dec cl
    div cl

    sub dl, al
    mov [ancho], dl
%endmacro
%macro setAltura 0       ;(140*ALTURA)/ALTURA_MAXIMA
    mov al, [altura]
    mov bl, 140
    mul bl

    mov bl, [alt_max]
    div bl

    mov [altura], al
%endmacro
%macro setColor 0        ;color
    mov cl, [altura]
    %%rojo:
        cmp cl, 20
        ja %%azul
        mov bl, 4
        jmp %%pintar
    %%azul:
        cmp cl, 40
        ja %%amarillo
```

Este macro define los márgenes para la gráfica de barras

```
%macro setMargenSupInf 0
    mov ax, 29      ;COLOR DEL
    mov dx, 290 ;ANCHO DEL FONDO
    %%margen:   ;LOOP PARA DIBUJAR A LA IZQUIERDA
        mov [es:di], ax     ;PINTAR EN ES+DI CON COLOR AX
        inc di
        dec dx
        jnz %%margen
%endmacro
%macro setMargenDerIzq 0
    mov ax, 29      ;COLOR DEL
    mov bx, 170     ;ALTURA DEL FONDO
    %%margen:       ;LOOP PARA DIBUJAR HACIA ABAJO
        mov [es:di], ax     ;PINTAR EN ES+DI CON COLOR AX
        add di, 320 ;NUEVA FILA
        dec bx
        jnz %%margen
%endmacro
%macro setMargen 0
    mov ax, 15          ;MARGEN ARRIBA
    mov bx, 25
    setXY
    setMargenSupInf

    mov ax, 15          ;MARGEN ABAJO
    mov bx, 195
    setXY
    setMargenSupInf

    mov ax, 15          ;MARGEN IZQUIERDA
    mov bx, 25
    setXY
    setMargenDerIzq

    mov ax, 305         ;MARGEN DERECHA
    mov bx, 25
```

```
%macro dibujarBarra 0
    mov ax, [color]      ;COLOR DEL
    mov bx, [altura]         ;ALTURA DEL FONDO
    %%loopRow:       ;LOOP PARA DIBUJAR HACIA ABAJO
        mov dx, [ancho] ;ANCHO DEL FONDO
        push di
        %%loopCol:  ;LOOP PARA DIBUJAR A LA IZQUIERDA
            mov [es:di], ax      ;PINTAR EN ES+DI CON COLOR AX
            inc di
            dec dx
            jnz %%loopCol
        pop di
        add di, 320 ;NUEVA FILA
        dec bx
        jnz %%loopRow
%endmacro
%macro dibujarBarras 0       ;BARRAS PAPA, BARRAS APRENDELAS :U
    mov dx, [tcant]
    mov cx, 0                ;0-12-24-32-...-120
    mov bx, 1                ;0-1-2-...-10
    %%cicloLectura:
        cmp dx, 0
        je %%fin
        push dx
        push cx
        push bx
                                 ;OBTENCION DE LA ALTURA Y COLOR
        mov si, cx
        add si, 8
        mov di, 0
        %%cicloAltura:
            mov dl, ar_desor[si]
            mov alt_Aux[di], dl
            inc si
            inc di
```

Ordenamiento.asm

Se pondrá un ejemplo con el ordenamiento de burbuja ya que el Quicksort funciona de la misma manera. La función de esto es simple, se ordena la matriz de 12x10 que contiene los datos a ordenar, esta es ordenada por medio del algoritmo ya definido y por cada movimiento que esta haga, por cada reemplazo o sustitución se procede a encender un sonido, seguido de pintar nuevamente la pantalla, esperar un tiempo, apagar el sonido y proseguir con el ordenamiento.

```
%macro ordenarBubble 0
    setDataSort
    inic_Graph

    imprimir t_orB, 04H
    ordenamientoEspacio

    ordenar_PintarBubble
    mov cl, [as_des]
    cmp cl, 1
    je %%siguiente
    ordenarBubble_Des
    jmp %%finProg
    %%siguiente:
    ordenarBubble_Asc
    %%finProg:
    imprimir t_orB, 04H
    ordenamientoEspacio
    mov ax,3h       ; funcion para el modo texto
    int 10h
%endmacro
```

```
%macro ordenarBubble_Des 0
    mov si, 1           ;I = 1
    %%For_I:
        mov cx, [tcant]     ;T
        cmp si, cx
        jnb %%fin
            mov di, cx
            dec di              ;J = T- 1
            %%For_J:
            cmp di, si
            jb %%finFor_I
            %%codigo:       ;ar_desor[] <  ar_linea[]
                push si
                push di

                xor ax, ax
                mov al, 12
                mul di
                mov di, ax
                mov si, 0
                %%llenarLinea:  ; J normal
                    mov dl, ar_desor[di]
                    mov ar_linea[si], dl
                    inc si
                    inc di
                    cmp si, 12
                    jb %%llenarLinea

                sub di, 24
                add di, 8
                mov si, 8
                %%comparar:
                    mov dl, ar_desor[di]
                    mov dh, ar_linea[si]
                    cmp dh, dl
                    ja %%finComparar
                    cmp dh, dl
                    jb %%cambio
```

```
%macro ordenar_PintarBubble 0
    %%ciclo:
        mov al, [segs]
        mov ah, [segMax]
        cmp ah, al
        jb %%fin
        imprimir t_orB, 04H
        ordenamientoDibujo
        Delay
        jmp %%ciclo
    %%fin:
        mov al, 0
        mov [segs], al
%endmacro
```

```
%macro sonidoAsignar 0
    mov di, 0
    mov si, 8
    %%cicloAltura:
        mov dl, ar_liaux[si]
        mov alt_Aux[di], dl
        inc si
        inc di
        cmp di, 4
        jb %%cicloAltura
    setAlturaNum

    mov al, [altura]
    %%rojo:
        cmp al, 20
        ja %%azul
        mov cx, 100
        jmp %%fin
    %%azul:
        cmp al, 40
        ja %%amarillo
        mov cx, 300
        jmp %%fin
    %%amarillo:
        cmp al, 60
        ja %%verde
        mov cx, 500
        jmp %%fin
    %%verde:
        cmp al, 80
        ja %%cxanco
        mov cx, 700
        jmp %%fin
```

```
%macro sonidoIniciar 0
 %%STARTSOUND: ;CX=FREQUENCY IN HERTZ. DESTROYS AX & DX
   CMP CX, 014H
   JB %%STARTSOUND_DONE
   ;CALL STOPSOUND
   IN AL, 061H
   ;AND AL, 0FEH
   ;OR AL, 002H
   OR AL, 003H
   DEC AX
   OUT 061H, AL     ;TURN AND GATE ON; TURN TIMER OFF
   MOV DX, 00012H   ;HIGH WORD OF 1193180
   MOV AX, 034DCH   ;LOW WORD OF 1193180
   DIV CX
   MOV DX, AX
   MOV AL, 0B6H
   PUSHF
   CLI ;!!!
   OUT 043H, AL
   MOV AL, DL
   OUT 042H, AL
   MOV AL, DH
   OUT 042H, AL
   POPF
   IN AL, 061H
   OR AL, 003H
   OUT 061H, AL
   %%STARTSOUND_DONE:
%endmacro
%macro sonidoTerminar 0
    IN AL, 061H
   AND AL, 0FCH
   OUT 061H, AL
%endmacro
```

Extra.asm

Son los textos que se utilizan mayormente o que su tamaño es considerable pero la importancia no lo es. Por esta razón se optó por crear este archivo y escribir en el todos los encabezado o saltos que se utilizaran.

```
%macro salto 0
    mov ah, 09h
    mov dx, slt
    int 21h
    mov dx, pts
    int 21h
%endmacro
```

```
%macro mostrarMenu_Usuario 0
    mostrarEncabezado
    mov dx, u_1
    int 21h
    mov dx, u_2
    int 21h
    mov dx, u_3
    int 21h
%endmacro

%macro mostrarMenu_Admin 0
    mostrarEncabezado
    mov dx, a_1
    int 21h
    mov dx, a_2
    int 21h
    mov dx, a_3
    int 21h
%endmacro
```

Error.asm

Incluye los errores que pueden llegar a existir y los mensajes a mostrar al usuario

```
%macro errorMain 0
    mov ah, 09h
    mov dx, er_Main
    int 21h
%endmacro

%macro errorOrd 0
    mov ah, 09h
    mov dx, er_Ord
    int 21h
%endmacro

%macro errorVel 0
    mov ah, 09h
    mov dx, er_Vel
    int 21h
%endmacro

%macro errorJuego 0
    mov ah, 09h
    mov dx, er_Jue
    int 21h
%endmacro

%macro errorAdmin 0
    mov ah, 09h
    mov dx, er_Adm
    int 21h
%endmacro
```

GetData.asm

Se incluye todos los macros orientados a la obtención de información por medio de la entrada del usuario, así como la limpieza de los vectores utilizados para almacenarlas.

```
%macro getChar 0
    mov ah, 01h
    int 21h
%endmacro

%macro getDatoUsr 0
    xor si, si
    mov dl, ' '
    %%ciclo:
        getChar
        cmp si, 7
            jae %%fin_ciclo
        cmp al, 0dh
            je %%fin_cicloEnter
        mov usr[si], al
        inc si
        jmp %%ciclo

    %%fin_cicloEnter:
        cmp si, 7
        jae %%fin_ciclo
        mov usr[si], dl
        inc si
        jmp %%fin_cicloEnter

    %%fin_ciclo:
        salto
%endmacro
```

```
%macro getDatoPsw 0
    xor si, si
    mov cl, 0
    mov dl, ' '
    %%ciclo:
        getChar
        cmp si, 4
            jae %%fin_ciclo
        cmp al, 0dh
            je %%fin_cicloEnter
        cmp al, 48
            jb %%noNum
        cmp al, 57
            ja %%noNum
        jmp %%continua
        %%noNum:
            mov cl, 1
        %%continua:
        mov psw[si], al
        inc si
        jmp %%ciclo

    %%fin_cicloEnter:
        cmp si, 4
        jae %%fin_ciclo
        mov psw[si], dl
        inc si
        jmp %%fin_cicloEnter

    %%fin_ciclo:
        salto
%endmacro
```

```
%macro resetPsw 0
    xor si, si
    xor al, al
    %%ciclo:
        cmp si, 4
            jae %%fin_ciclo
        mov psw[si], al
        inc si
        call %%ciclo
    %%fin_ciclo:
%endmacro

%macro resetTexto 0
    xor si, si
    mov al, '$'
    %%ciclo:
        cmp si, 255
            jae %%fin_ciclo
        mov texto[si], al
        inc si
        call %%ciclo
    %%fin_ciclo:
%endmacro

%macro resetPth 0
    xor si, si
    mov al, 0
    %%ciclo:
        cmp si, 255
            jae %%fin_ciclo
        mov pth[si], al
        inc si
        call %%ciclo
    %%fin_ciclo:
%endmacro
```