UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1
SEGUNDO SEMESTRE 2020
ING. OTTO ESCOBAR
TUTOR ACADÉMICO SECCIÓN A: HERBERTH ARGUETA


# MANUAL TECNICO: PROYECTO 2

NOMBRE: Juan José Ramos Campos
CARNET: 201801262
FECHA: 9/11/20

# Contenido

# macros.inc

Este es el archivo principal de la aplicación, cuenta con una serie de macros especiales que le permiten realizar diferentes cosas entre ellas:

## print

```
print macro cadena ; 
MOV ah,09h
MOV dx,@data
MOV ds,dx
MOV dx, offset cadena
int 21h
endm
```

Esta macro sirve para hacer un print en pantalla de una cadena que recibe como parámetro

## getText

```
getText macro buffer
LOCAL CONTINUE, FIN
PUSH SI
PUSH AX

xor si,si
CONTINUE:
getChar
cmp al,0dh
je FIN
mov buffer[si],al
inc si
jmp CONTINUE

FIN:
mov al, '$'
mov buffer[si],al

POP AX
POP SI
endm
```

Esta macro sirve para poder escribir desde el teclado, recibe como parámetro un arreglo de caracteres donde se almacenara cada carácter pulsado en el teclado

## getfecha y getHora

```
;OBTENER FECHA
getfecha macro
    MOV AH,2AH
    INT 21H
endm

;OBTENER HORA
gethora macro
    MOV AH,2CH
    INT 21H
endm
```

Esta macros sirven para obtener la fecha y hora de la terminal de Windows

## obtenerFechaHora

```
;OBTENER FECHA
obtenerFechaHora macro dia, mes, ano,
    XOR AX,AX
    XOR BX,BX
    XOR CX,CX
    getfecha
    MOV BX,CX
    convertirString1 BX,ano
    XOR BX,BX
    getfecha
    MOV BL,DH
    convertirString1 BX,mes
    getfecha
    MOV BL,DL
    convertirString1 BX,dia
    XOR BX,BX
    gethora
    MOV BL,CH
    convertirString1 BX,hora
    gethora
    MOV BL,CL
    convertirString1 BX,minuto
    gethora
    MOV BL,DH
    convertirString1 BX,segundo
endm
```

esta macro coloca en los arreglos correspondientes, los datos de fecha y hora obtenidos con la macro anterior.

## CovertirString

```
ConvertirString macro buffer
    LOCAL Dividir,Dividir2,FinCr3,NEGATIVO,FIN2,FIN
    xor si,si
    xor cx,cx
    xor bx,bx
    xor dx,dx
    mov dl,0ah
    test ax,1000000000000000
    jnz NEGATIVO
    jmp Dividir2

    NEGATIVO:···
    Dividir:
        xor ah,ah
    Dividir2:···
    FinCr3:···
    FIN:
endm
```

Esta macro convierte a string un numero

## ConvertirAscii

```
CovertirAscii macro numero
    LOCAL INICIO,FIN
    xor ax,ax
    xor bx,bx
    xor cx,cx
    mov bx,10   ;multiplicador 10
    xor si,si
    INICIO:···
    FIN:
endm
```

Esta macro, al contrario que la anterior, convierte un numero a string

## Clear

Esta macro sirve para limpiar los arreglos que se hayan declarado y utilizado

```
clear macro buffer
LOCAL INICIO, SALIR
    PUSH DI
    xor DI,DI
INICIO:
    CMP DI,20
    JE SALIR
    CMP buffer[DI], '$'
    JE SALIR

    MOV buffer[DI], '$'
    INC DI
    JMP INICIO
SALIR:
    XOR DI, DI
    POP DI
endm
```

# Archivo.inc

## Openfile

```
openFile macro ruta,handle
mov ah,3dh
mov al,10b
lea dx,ruta
int 21h
mov handle,ax
jc ErrorAbrir
endm
```

Esta macro se encarga de abrir un archivo según la path o nombre

## CloseFile

```
closeFile macro handle
mov ah,3eh
mov handle,bx
int 21h
endm
```

Se encarga de cerrar el archivo abierto

## CréateFile

```
createFile macro buffer,handle
mov ah,3ch
mov cx,00h
lea dx,buffer
int 21h
mov handle,ax
jc ErrorCrear
endm
```

Se encarga de crear un archivo según el tipo de extensión

## writeFile

```
writeFile macro numbytes,buffer,handle
    mov ah, 40h
    mov bx,handle
    mov cx,numbytes
    lea dx,buffer
    int 21h
    jc ErrorEscribir
endm
```

sirve para escribir dentro del archivo

# Alogin.inc

## compararUser

Esta macro sirve para el login, compara si el usuario enviado es el usuario administrador o susuario normal, si son de administrador, llama a la macro compararPassword para verificar la contraseña

```
COMPARARUSER MACRO user, pass
LOCAL IGUAL, DIFERENTE


    xor CX, CX
    mov cx,7   ;Determinamos la cantidad
    mov AX,DS  ;mueve el segmento datos
    mov ES,AX  ;Mueve los datos al segme
    lea si, user               ; Reset
    lea di, admin

    repe cmpsb  ;compara las dos cadenas
    je IGUAL ;Si fueron iguales
    jne DIFERENTE
    IGUAL:
        COMPARARPASS pass
    DIFERENTE:
        COMPARARUSERENLISTA user, pass
ENDM
```

## compararPass

Esta macro sirve para evaluar si la password enviada es de administrador, si lo es entonces se redirige al menú de aministrador, si no lo es, marca error de login

```
COMPARARPASS:

    xor CX, CX
    mov cx,4   ;Determinamos la cantidad de da
    mov AX,DS  ;mueve el segmento datos a AX
    mov ES,AX  ;Mueve los datos al segmento ex
    lea si, pass               ; Reset pointe
    lea di, usuariotemp

    repe cmpsb  ;compara las dos cadenas
    je IGUAL2 ;Si fueron iguales
    jne ERROR
    IGUAL2:
        print salto
        print usuarionoexiste
        print salto
        getChar
        JMP NUEVO_JUEGO

ERROR:
    print errorlogin
    JMP INGRESAR
```

## compararUserenLista

Esta macro se encarga de verificar si el usuario esta en registrado, en caso de que no es administrador, si no esta registrado marca error de login, si esta registrado lo envia a nuevo juego.

```
COMPARARUSERENLISTA MACRO user, pass
LOCAL INICIO, FIN, ERROR, GETUSER, VERIFICARUSER, IGUA
    MOV DI, 0
    MOV SI, 0
    clear usuariotemp
INICIO: ···
GETUSER: ···
VERIFICARUSER: ···
GETPASS: ···
ENDM
```

## compararRegistro

Esta macro se encarga de buscar un usuario dentro de la lista para verificar si existe o no y evitar que otro usuario tome su user, si ya existe retorna un mensaje de error de que el usuario no puede usar ese nombre de usuario.

```
COMPARARREGISTRO MACRO buffer
LOCAL INICIO, ENDF, REGRESAR, SIEXISTE, TERMINAR
    push CX
    push AX
    push SI
    push DI
    xor DI, DI
    xor SI, SI
    mov SI, 0
    MOV DI, 0
INICIO: ···
SIEXISTE: ···
REGRESAR: ···
ENDF:
    print usuarionoexiste
    POP DI
    POP SI
    POP AX
    POP CX
    JMP REGISTRAR2
TERMINAR:
    POP DI
    POP SI
    POP AX
    POP CX
    clear usuario
ENDM
```

InsertarUser

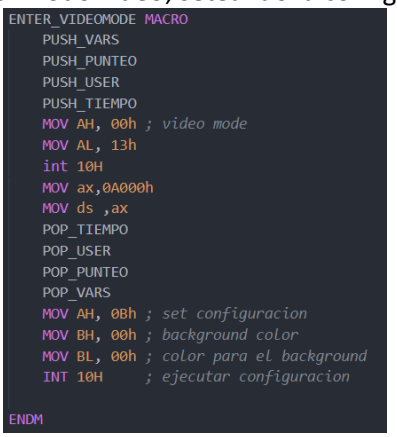
En caso de que el usuario este disponible, esta macro se encarga de guardarlo en el arreglo de usuarios.

```
INSERTARUSER MACRO user, password
LOCAL INICIO, INSERTUSERNAME, INSERTDOSPUNTOS, INSERTPASSWORD, TERMINAR
MOV listausuarios[0], ';'
MOV DI, 0
MOV SI, 0
INICIO:
    CMP listausuarios[SI], '$'
    JE INSERTUSERNAME
    INC SI
    JMP INICIO
INSERTUSERNAME: ···
INSERTDOSPUNTOS: ···
INSERTPASSWORD: ···
TERMINAR: ···
ENDM
```

# Ajuego.in

## Enter_videomode

Esta macro se encarga de entrar al modo video, seteando la configuración necesaria para jugar

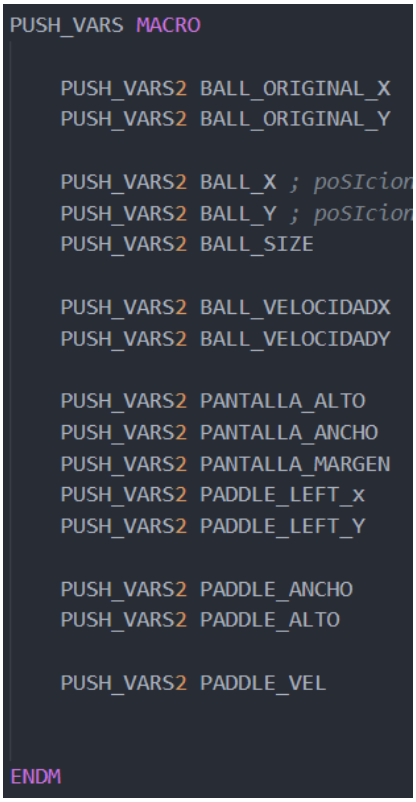


```
ENTER_VIDEOMODE MACRO
    PUSH_VARS
    PUSH_PUNTEO
    PUSH_USER
    PUSH_TIEMPO
    MOV AH, 00h ; video mode
    MOV AL, 13h
    int 10H
    MOV ax,0A000h
    MOV ds ,ax
    POP_TIEMPO
    POP_USER
    POP_PUNTEO
    POP_VARS
    MOV AH, 0Bh ; set configuracion
    MOV BH, 00h ; background color
    MOV BL, 00h ; color para el background
    INT 10H     ; ejecutar configuracion

ENDM
```

## PushVars

Esta macro se encarga de hacer push de todas las variables necesarias para el juego, dado que al entrar en modo video, la referencia a estas se pierde, es necesario guardarlas



```
PUSH_VARS MACRO

    PUSH_VARS2 BALL_ORIGINAL_X
    PUSH_VARS2 BALL_ORIGINAL_Y

    PUSH_VARS2 BALL_X ; posIcion
    PUSH_VARS2 BALL_Y ; posIcion
    PUSH_VARS2 BALL_SIZE

    PUSH_VARS2 BALL_VELOCIDADX
    PUSH_VARS2 BALL_VELOCIDADY

    PUSH_VARS2 PANTALLA_ALTO
    PUSH_VARS2 PANTALLA_ANCHO
    PUSH_VARS2 PANTALLA_MARGEN
    PUSH_VARS2 PADDLE_LEFT_x
    PUSH_VARS2 PADDLE_LEFT_Y

    PUSH_VARS2 PADDLE_ANCHO
    PUSH_VARS2 PADDLE_ALTO

    PUSH_VARS2 PADDLE_VEL


ENDM
```

## PopVars

Esta macro es lo contrario que lo anterior, una vez en modo video, se hace un POP de las variables necesarias para poder acceder a ellas y jugar con normalidad

```
POP_VARS MACRO

    POP_VARS2 PADDLE_VEL

    POP_VARS2 PADDLE_ALTO
    POP_VARS2 PADDLE_ANCHO

    POP_VARS2 PADDLE_LEFT_Y
    POP_VARS2 PADDLE_LEFT_x

    POP_VARS2 PANTALLA_MARGEN
    POP_VARS2 PANTALLA_ANCHO
    POP_VARS2 PANTALLA_ALTO

    POP_VARS2 BALL_VELOCIDADY
    POP_VARS2 BALL_VELOCIDADX


    POP_VARS2 BALL_SIZE
    POP_VARS2 BALL_Y
    POP_VARS2 BALL_X
    POP_VARS2 BALL_ORIGINAL_Y
    POP_VARS2 BALL_ORIGINAL_X
    MOV AUX_TIEMPO, 0
ENDM
```

## POP_Punteo y PUSH_punteo

Ambas macros, sirven al igual que las anteriores, para guardar el punteo y luego sacarlo cuando se entra en modo video, dado que se puede perder la referencia

```
PUSH_PUNTEO MACRO
    MOV AL, CONTADOR_PUNTOS
    MOV AH, 0
    PUSH AX
ENDM


POP_PUNTEO MACRO
LOCAL INICIO, FIN
    POP AX
    MOV CONTADOR_PUNTOS, AL
    MOV AH, 0
    ConvertirString puntaje
ENDM
```

## PUSH_Usuario y POP_Usuario

Ambas macros, sirven al igual que las anteriores, para guardar el usuario logueado y luego sacarlo cuando se entra en modo video, dado que se puede perder la referencia

```
PUSH_USER MACRO
LOCAL INICIO, FIN
MOV DI, 0
INICIO:
    MOV AL, usuario[DI]
    MOV AH, 0
    PUSH AX
    INC DI
    CMP DI, 8
    JE FIN
    JMP INICIO
FIN:
endm


POP_USER MACRO
LOCAL INICIO, FIN
MOV DI, 7
INICIO:
    POP AX
    MOV usuario[DI], AL
    DEC DI
    CMP DI, -1
    JE FIN
    JMP INICIO
FIN:
ENDM
```

## Draw_Block

Esta macro sirve para pintar los bloques en la pantalla

```
;============================ DIBUJAR LOS BLOQUES
DRAW_BLOCK macro
    local LOOP_FOR,NEXT_LOOP

        MOV BX, 0
        MOV DI, 6736 ;pixel de inicio = esquina superior izq
        MOV SI, 0
        MOV DX, 5 ; color
        MOV CX, 3FC0h

        LOOP_FOR: ···
        NEXT_LOOP:
            loop LOOP_FOR
    endm
```

## Draw_Borde

Esta macro sirve para pintar los bordes en la pantalla

```
;================================ DIBUJAR EL BORDE
DRAW_BORDE MACRO
LOCAL INICIO, FIN
local SUPERIOR, INFERIOR, IZQUIERDO, DERECHO, NEXT_LOOP,
endm
```

## Draw_User

Dibuja el nombre de usuario en la parte superior de la pantalla

```
;============================== DIBUJA
DRAW_USER MACRO
LOCAL INICIO, FIN
    MOV DI, 0
    MOV CONTADOR_LOOPS, 2
INICIO:
    MOV AL, usuario[DI]
    LOOP_USER AL, CONTADOR_LOOPS
    INC DI
    INC CONTADOR_LOOPS
    CMP DI, 8
    JE FIN
    JMP INICIO
FIN:
ENDM
```

## Draw_Nivel

Dibuja el nivel actual en la parte superior de la pantalla

```
;======================== DIBUJAR
DRAW_NIVEL MACRO
    ;REPOSICIONAR EL CURSOR
    MOV AH, 02H
    MOV BH, 0       ;Página de vídeo
    MOV DH, 1h  ;Línea donde situar
    MOV DL, 0Fh     ;Columna donde s
    INT 10H
    ; ESCRIBIR 'N'
    MOV AH, 09H
    MOV AL, 78  ; Código del carácte
    MOV BH, 0   ; Página de vídeo do
    MOV BL, 0Fh ; Atributo ó color q
    MOV CX, 1   ; Cantidad de veces
    INT 10h

    ;REPOSICIONAR EL CURSOR
    MOV AH, 02H
    MOV BH, 0       ;Página de vídeo
    MOV DH, 1h  ;Línea donde situar
    MOV DL, 10h     ;Columna donde s
    INT 10H
    ; ESCRIBIR NIVEL
    MOV AH, 09H
    MOV AL, 49  ; Código del carácte
    MOV BH, 0   ; Página de vídeo do
    MOV BL, 0Fh ; Atributo ó color q
    MOV CX, 1   ; Cantidad de veces
    INT 10h
ENDM
```

## Draw_Puntos

Dibuja el punjate en la parte superior de la pantalla

```
;========================= DIBUJAR LOS PUNTOS
AUMENTAR_PUNTOS MACRO
    INC CONTADOR_PUNTOS
ENDM

DRAW_PUNTOS MACRO
LOCAL INICIO, FIN, CORREGIR, NORMAL
    LOOP_PUNTOS '0', 14h

    MOV AL, CONTADOR_PUNTOS
    CMP AL, 9
    JG CORREGIR
    JMP NORMAL

CORREGIR:
    LOOP_TIEMPO puntaje[0], 15h
    LOOP_TIEMPO puntaje[1], 16h
    JMP FIN
NORMAL:
    LOOP_TIEMPO puntaje[1], 15h
    LOOP_TIEMPO puntaje[0], 16h

FIN:

ENDM
```

## Draw_Timer

Se encarga de dibujar el timer e irlo actualizando con el pasar del tiempo

```
DRAW_TIMER MACRO
LOCAL INICIO, CEROCERO, NORMAL, MOSTAR_MINUTO, AUMETAR_MINUTO, AUMENTAR_HORA, MOSTAR_HOR
AUMETAR_MINUTO: ···
MOSTAR_MINUTO: ···
CORREGIR: ···
NORMAL:   ···
AVANZAR: ···
AUMENTAR_HORA: ···
MOSTAR_HORA: ···
ENDM
```

# Amain.in

## Nuevo_Juego

Esta macro es la mas importante, se encarga de prácticamente toda la funcionalidad del juego, primero accede al modo video, pushea las variables, les hace pop y dibuja todos los elementos del juego para su funcionalidad.

```
NUEVO_JUEGO:


    INIT:
        ENTER_VIDEOMODE
        DRAW_USER
        DRAW_BORDE
        DRAW_TIMER
        DRAW_NIVEL
        DRAW_PUNTOS
        DRAW_BLOCK
        CALL PINTBALL
        CALL DRAWPADDLES

    INICIO_JUEGO: ···
    VERIFICAR_TIEMPO: ···
    SALIR:
        ENTER_VIDEOMODE
        DRAW_BORDE
        DRAW_MENSAJE
        getChar
        ;regresara a modo texto
        mov ax,0003h
        int 10h
        call main
    JMP FIN
```

## Draw_Paddle

Se encarga de dibujar la barra horizontal en  la posición x y indicada

```
DRAWPADDLES PROC near
    MOV CX, PADDLE_LEFT_x
    MOV DX, PADDLE_LEFT_Y

    DRAW_PADDLE_LEFT_HORIZONTAL:
        MOV AH, 0Ch ;config para escribir pixel
        MOV AL, 0FH ; choose color
        MOV BH, 00h ; set the page number
        INT 10H         ; execute configuration
        INC CX      ; CX = CX + 1
        MOV AX, CX
        SUB AX, PADDLE_LEFT_x
        CMP AX, PADDLE_ALTO
        JNG DRAW_PADDLE_LEFT_HORIZONTAL
        MOV CX, PADDLE_LEFT_x
        INC DX
        MOV AX, DX
        SUB AX, PADDLE_LEFT_Y
        CMP AX, PADDLE_ANCHO
        JNG DRAW_PADDLE_LEFT_HORIZONTAL
    RET
DRAWPADDLES ENDP
```

## Reset_PosicionPelota

Se encarga de resetar la posición de inicio de la pelota cuando esta cae sobre el borde inferior, es decir, cuando se pierde una vida

```
;======== RESETEA LA POSICION DE LA PELOTA CUANDO LLEGA AL MARGEN DE ABAJO
RESET_POSICION_PELOTA proc near
    MOV BALL_VELOCIDADX, 02h ; VELOCIDAD EN X DE LA PELOTA
    MOV BALL_VELOCIDADY, 02h  ; VELOCIDAD EN Y DE LA PELOTA
    MOV AX, BALL_ORIGINAL_X
    MOV BALL_X, AX
    MOV AX, BALL_ORIGINAL_Y
    MOV BALL_Y, AX

    RET
RESET_POSICION_PELOTA endp
```

## MOVEBALL

Otra de las macros mas importantes, se encarga del movimiento de la pelota en la pantalla, así como controlar las colisiones con los bordes, los bloques y la barra

```
MOVERBALL proc near

    BLOQUES:        ...
    NORMAL: ...
    NEG_VELOCIDAD_Y:
        AUMENTAR_PUNTOS
        NEG BALL_VELOCIDADY
        JMP NORMAL

    CHECK_COLISION:
        ENTER_VIDEOMODE
        ;NEG BALL_VELOCIDADY

    RET

    NEG_VELOCIDAD_X:
        NEG BALL_VELOCIDADX
        RET
    RESET_POSICION:
        CALL RESET_POSICION_PELOTA
        RET

    COLICION_ARRIBA:
        ;NEG BALL_VELOCIDADX
        NEG BALL_VELOCIDADY
        RET


MOVERBALL endp
```

## MOVEPADDLE

Otra de las macros mas importantes, se encarga de mover la barra de forma horizontal controlando que no pase de los márgenes establecidos

```
MOVEPADDLES PROC near


    ;VERIFICAR SI SE PRESIONO UNA TECLA, SI NO, SAL
    MOV AH, 01h
    INT 16h
    JZ CHECK_MOVIMIENTO ; SI ZF = 1, entonces JZ->



    ; SI SE PRECIONO VERIFICAR QUE TECLA FUE; AL =
    MOV AH, 00h
    INT 16h
    ;SI ES 'A' IQUIERDA, 'D' DERECHA
    CMP AL, 61h ; -> 'a'
    JE MOVE_IZQUIERDA
    CMP AL, 41h ; -> 'A'
    JE MOVE_IZQUIERDA


    CMP AL, 64h ; -> 'd'
    JE MOVE_DERECHA
    CMP AL, 44h ; -> 'D'
    JE MOVE_DERECHA


    CMP AL, 1Bh ; -> 'ESC'
    JE INIT
```

## PINTBall

Se encarga de pintar la pelota en la posición xy indicada, tanto en la posición de inicio y durante todo su movimiento

```
PINTBALL proc near

    MOV CX, BALL_X ; set column x
    MOV DX, BALL_Y ; set column y

    DIV_PELOTA_HORIZONTAL:
        MOV AH, 0Ch ;config para escribir pixel
        MOV AL, 0FH ; choose color
        MOV BH, 00h ; set the page number
        INT 10H         ; execute configuration
        INC CX      ; CX = CX + 1
        MOV AX, CX
        SUB AX, BALL_X
        CMP AX, BALL_SIZE
        JNG DIV_PELOTA_HORIZONTAL
        MOV CX, BALL_X
        INC DX
        MOV AX, DX
        SUB AX, BALL_Y
        CMP AX, BALL_SIZE
        JNG DIV_PELOTA_HORIZONTAL

    ret
PINTBALL endp
```