Universidad de San Carlos de Guatemala
Arquitectura de computadores y ensambladores 1
Escuela de Sistemas
Ing. Otto Rene Escobar Leiva
Aux Mario Pineda

Manual de Usuario:

Proyecto 2

Nombre: **Carnet**:
Brandon Oswaldo Yax Campos 201800534

Guatemala, 02 de mayo del 2022

Introducción:

El proyecto tuvo como objetivo la creación de un juego llamado galaga a través de la memoria de gráficos que en assembler se puede manejar, teniéndose además del juego un apartado para registro de usuarios, uno para login y un menu dependiendo de los tres tipos de usuarios que pueden existir (usuario normal, usuario admin, administrador general) teniendo entre las opciones que dichos menús proveen la posibilidad de manipular los datos que el juego registraba a través de un score en un archivo externo, pudiéndose consultar el top 10 de usuarios con mejor score y su partida, así como un apartado para visualizar métodos de ordenamiento, tomando como datos a analizar ya sea el score o el tiempo requerido para finalizar cada juego, además de proveer al administrador general la posibilidad de quitar bloqueos de usuarios asi como promover y quitar administrador. Se llegó a la conclusión que la memoria de gráficos facilita enormemente la necesidad de manipular objetos movibles en forma de píxeles, así como la facilidad que esta provee para guardar su estado en un archivo externo y realizar diversos gráficos con esta.

Objetivos

Objetivo General

- Que el desarrollador pueda aplicar los conocimientos adquiridos en el curso y que éste sea capaz de construir un sistema complejo mediante el lenguaje ensamblador haciendo uso de algoritmos creativos para solucionar los distintos requerimientos solicitados.

Objetivos Específicos

- Implementar soluciones creativas para algoritmos complejos.
- Manipular correctamente la memoria del sistema.
- Mezclar diferentes familias de funciones en interrupción.
- Aplicar instrucciones aritméticas a la solución de algoritmos.
- Aplicar instrucciones lógicas a la solución de algoritmos.
- Aprender a utilizar como entrada teclas auxiliares.
- Comprender y aplicar el manejo de memoria de video en ensamblador.

Código:

main: Codigo principal con el cual se llamará al resto de macros y procedimientos:

```
include macro.asm
.MODEL small
.STACK
.RADIX 16
.DATA
;APARTADO PARA LA DECLARACION DE VARIABLES Y LISTAS
mVariables
.CODE
;APARTADO PARA EL CODIGO
start:
    main proc
        call pFlujoProyecto2
    main endp
END start
```

pFlujoProyecto2: Procedimiento con el cual se procedera a llamar al resto de macros y procedimientos.

```
pFlujoProyecto2 proc
    call pAjustarMemoria
        call pBaseDatos
        call pLimpiarConsola
        mMostrarString mensajeI
         ;apartado de espera de un enter----------------------
            call pEspEnter
        ;-------------------------------------------------
        call pLimpiarConsola
        call pMenuPrincipal
    ;call pMenuOrd
    call pRetControl
    ret
pFlujoProyecto2 endp
```

pLImpiarconsola: con este método es posible limpiar la consola en los apartados donde sea necesario.

```
pLimpiarConsola proc
    push ax
    push bx
```

```
    push cx
    push dx
    ;Limpia la consola
    mov ax,0600h ; es igual a mov ah,06 (scroll up windows con el int
10)  y mov al,00
    mov bh, 07
    mov cx, 00000  ; es igual a mov ch,0   mov cl, 0 , filas y coumnas
de derecha a izquierda
    mov dx, 184FH  ;filas y columnas de both
    int 10
    ;posiciona el cursor en la pos 0
    mov ah, 02
    mov bh,0  ;numero de pagina
    mov dl,0   ;columna
    mov dh,0   ;fila
    int 10
    pop dx
    pop cx
    pop bx
    pop ax
    ret
pLimpiarConsola  endp
```

pMenuPrincipal: Menú principal con el cual se accedera ya se a login, a registrar o a salir del programa, cabe aclarar que las teclas Fn's en la laptop donde fue trabajado el proyecto para ser activadas es necesario apretar una tecla especial llamada Fn, por lo cual se llama dos veces a la interrupción ah 01-int 21, una para capturar el fn y otra para capturar el valor de la tecla auxiliar.

```
pMenuPrincipal proc
    ciclomenu:
        mov opcion,0
        mMostrarString Menu
        ;la laptop que se posee para trabajar esto necesita de
presionar una tecla antes de los FN
        ; para que los reconozca por tal motivo s   e hizo esto dos
veces para que se pudiera a trapar el valor de Fn
        mov ah,01
        int 21
```

```
        mov ah,01  ;atrapa la tecla fn
        int 21
        mov opcion,al
        cmp opcion,59t
        je Login
        cmp opcion,"<"
        je Register
        cmp opcion,"C"
        je salir
        mMostrarString opi
        jmp ciclomenu
    Login:
        call pLogin
        call pLimpiarConsola
        jmp ciclomenu
    Register:
        call pResetFlagsE ;RESETEA LAS BANDERAS QUE DETECTAN DE ERRORES
A LA HORA DE REGISTRAR
        call pRegistrar
        call pLimpiarConsola
        jmp ciclomenu
    salir:
        call pLimpiarConsola
    ret
pMenuPrincipal endp
```

LOGIN:

pLogin: procedimiento que presentara las opciones necesarias para logearse, teniendo como principal objetivo el capturar un usuario y contraseña, verificando en primer lugar si el usuario existe, para posteriormente revisar si la contraseña es correcta o no y revisar si el usuario ha sido bloqueado y por lo tanto aun con la contraseña correcta no le sea posible ingresar (cuando se equivoca 3 veces un usuario con su contraseña es bloqueado).

```
pLogin proc
    call pResetFlagsE
    mOpenFile2Write usersb ;abre el archivo de users
    cicloLogin:
    call pLimpiarConsola
```

```
    call pInidoc ;COLOCAR EL PUNTERO AL INICIO DEL DOCUMENTO
    mLimpiar UsuarioI,25,24 ;limpia el espacio para  almacenar el
usuario ingresado
    mLimpiar PasswordI,25,24 ;limpia el espacio donde se almacenara
temporalmente la contra ingresada
    mMostrarString msgLogin ;muestra mesaje de login
    mMostrarString msgexit
    ;captura de usuario y contraseña
    mMostrarString rU
    mCapturarString UsuarioI
    mMostrarString rP
    mCapturarPassword PasswordI
    ;ADMIN PRINCIPAL=====================================================
    cmp UsuarioI[0],09 ;tab=Exit  ;POR SI SE QUIERE SALIR DEL MENU
    je exitab
    mReadFile eleActual
    mEncontrarId UsuarioI
    cmp idEncontrado,0
    je noesadmin
    esadmin: ;ES EL ADMIN PRINCIPAL
        mHallarSimbolo 01 ;pasa al separador que esta alapar de
contraseña
        mReadFile eleActual;se salta el separador
        mEncontrarId PasswordI;verificar si la contraseña es la
correcta
        cmp idEncontrado,0
        je PasswordIncorrect
        ;USUARIO GENERAL
PRINCIPAL=======================================
        cAdminCor:;password de admin correcta
            call pQuitarbloqAdmin ;al ingresar la contraseña correcta
tanto nveces error y bloqueo se vuelven a su valor default
            call pCloseFile
            call pLimpiarConsola
            call pMenuAdmin
            jmp salir
    ;USUARIO O USUARIO
ADMIN================================================
    noesadmin: ;ENTONCES ES UN USUARIO NORMAL o ADMIN SECUNDARIO
        mUserExiste UsuarioI
        cmp existee,1 ;EXISTE USUARIO?
        jne Noexiste  ;NO EXISTE
```

```
        ;luego del metodo mUserExiste  estara posicionado justo en la
linea deseada
        mHallarSimbolo 01 ;Separador alapar de contraseña
        mReadFile eleActual ;Primer elemento de contraseña
        mEncontrarId PasswordI
        cmp idEncontrado,1
        jne PasswordIncorrect
        je PasswordCorrect
        ;EXISTE?
    PasswordCorrect:
        mHallarSimbolo 01 ;separador a la par de N veces repetido
        mHallarSimbolo 01 ;separador a la par de bloqueado o no
bloqueado
        mReadFile eleActual ;B o N
        cmp eleActual,"B" ;si esta bloqueado a pesar de tener buena la
contraseña, no ingresara
        je Ubloqueado
        mHallarSimbolo 01 ;separador a la par de admin o no admin  (A O
N)
        mReadFile eleActual ;A o N
        cmp eleActual,"A"
        je Usuarioadmin
        jne UsuarioNormal
    ;USUARIO NORMAL=====================================================
    UsuarioNormal:
        call pCloseFile
        call pLimpiarConsola
        call pMenuUser ;MUESTRA MENU CORRESPONDIENTE
        jmp salir
    ;USUARIO ADMIN======================================================
    Usuarioadmin:
        call pCloseFile
        call pLimpiarConsola
        call pMenuU_admin ;MUESTRA MENU CORRESPONDIENTE
        jmp salir
    Ubloqueado:
        mHallarSimbolo 01
        mReadFile eleActual ;A o N (admin o no admin)
        cmp eleActual,"A"
        jne noadminU
            call pPosAnterior ;A/n
            call pPosAnterior ;separador antes de A/n
            call pPosAnterior ;B/No bloqueado
```

```
            call pPosAnterior ;separador antes de B/n
            call pPosAnterior ;N VECES BLOQUEADO
            call pPosAnterior ;separador
            call pPosAnterior ;ultimo digito de la contraseña
            call pQuitarbloqAdmin
            jmp Usuarioadmin
        noadminU:
        mMostrarString msgUbloqueado
        call pespEnter
        jmp cicloLogin
    Adminbloqueado:
        call pDelay30
        jmp cicloLogin
    PasswordIncorrect:
        ;LE SUMA UNO AL NUMERO DE ERRORES
        mMostrarString msgPinc
        call pIncVEquivoco
        call pDarbloqueo
        call pEspEnter
        ;SABER SI ES ADMIN O NO, AQUI YA RECORRIO LA POSICION DE NVECES
ERROR Y LA DE BLOQUEO CON
        ;UN SOLO HALLAR SIMBOLO SE PASARIA AL APARTADO PARA SABER SI ES
ADMIN O NO
        mHallarSimbolo 01
        mReadFile eleActual
        cmp eleActual,"A"
        je PasswordIAdmin;SI ES ADMIN
        ;SI NO SEGUIR
        jmp cicloLogin
    PasswordIAdmin:
        call pPosAnterior ;A/n
        call pPosAnterior ;separador antes de A/n
        call pPosAnterior ;B/No bloqueado
        mReadFile eleActual
        cmp eleActual,"B"
        je Adminbloqueado
        jmp cicloLogin
    Noexiste:
        mMostrarString msgUnE ;MENSAJE USUARIO NO EXISTE
        call pEspEnter
        jmp cicloLogin
    ;NO EXISTE
    exitab:
```

```
        call pCloseFile
    salir:
    ret
pLogin endp
```

mUserExiste: macro para identificar si el usuario existe, y además posicionarlo en la fila del documento leído correcta para posteriormente tomar los datos de esta.

```
mUserExiste macro Username
    local Existe,Noexiste,salir,cicloexiste
    ;SE VERIFICA SI NO ES EL ADMIN
    mReadFile eleActual ;TOMA EL PRIMER VALOR DEL ARCHIVO
    mEncontrarId Username;lo primero en el documento de usuarios es el
admin, que siempre estara aca
    cmp idEncontrado,1 ; se encontro usuario?
    je Existe ;si se encontro se procede a decir que si existe el
usuario y se marcara como error
    cicloexiste: ;caso contrario se procedera a un ciclo de lectura del
archivo hasta hallar o un espacio o el id buscado
        mHallarSimbolo 0A  ;se salta hasta el enter hasta la posicion
donde esta 0A
        mReadFile eleActual ; se corre una vez el elemento
        cmp eleActual," " ;si hay un espacio es que ya se llego al fin
del documento y el usuario no existe
        ;CADA VEZ QUE SE CREA UN USUARIO SE ELIMINA EL ULTIMO ESPACIO
QUE DEJA LA CREACION DEL USUARIO ANTERIOR
        je Noexiste ; no existe usuario
        mEncontrarId Username ;si no es espacio lo que esta en esta
posicion fijo es un nombre de user, el user a registrar  es igual a
este?
        cmp idEncontrado,1 ; si, entonces existe
        je Existe
        jne cicloexiste
    Existe:
        mov existee,1 ;se reporta error pues existe usuario que se
intenta registrar
        mov eerror,1  ;se reporta error general al registro
        jmp salir
    Noexiste:
        mov existee, 0 ; no existe usuario, no hay error
    salir:
```

```
endm
```

REGISTRAR:

pRegistrar: Procedimiento con el objetivo principal de capturar un usuario y contraseña para luego verificar que cumpla con las características requeridas de estos, para hacer validas un posible registro.

Características para hacer válido un registro:

1. Se solicitará el nombre de usuario y se harán lan siguientes validaciones:

1. No puede empezar por número.
2. La longitud debe validarse entre 8 y 15 caracteres.
3. El nombre de usuario no debe existir.
4. Los únicos caracteres especiales que puede contener serán: Guión (-) Guión bajo (_) Punto (.)

2. Se solicitará el password para el usuario y se harán las siguientes validaciones.

1. Debe tener al menos una mayúscula.
2. Debe tener al menos un número.
3. Debe tener al menos un caracter especial de los siguientes: Admiración (!) Mayor qué (>) Porcentaje (%) Punto y coma (;) Asterisco (*)
4. Longitud mínima de 16 caracteres.
5. Longitud máxima queda a discreción del desarrollador.

```
pRegistrar proc
    mov eerror,0
    mLimpiar UsuarioRegis,25,24
    mLimpiar PasswordRegis,25,24
    call pLimpiarConsola
```

```
mMostrarString msgRegister
mMostrarString rU
mCapturarString UsuarioRegis
mMostrarString rP
mCapturarPassword PasswordRegis
;RESTRICCIONES
mUserInicial
mSizeUser
mUserExisteR
mRequisitoCletra
mAMayus
mANum
mASigno
mSizePassword
;EXISTE ERROR?
mComparar eerror,1
je ErrorRegistro
jne noErrorRegistro
ErrorRegistro:
    mMostrarString ActionR
    ;NUMERO INICIAL
    mComparar numinicio,0
    je nNinicial
    yNinicial:;error posee un numero en su caracter incial
        mMostrarString msginitialbad
    nNinicial:; no posee error de este tipo

    mComparar largoe,0
    je nLerornea
    ;LONGITUD ERRONEA
    yLerronea: ; posee error
        mMostrarString msglengtherror
    nLerornea:; no posee error de este tipo

    ;USUARIO EXISTENTE
    mComparar existee,0
    je nUexist
    yUexist:;usuario existe
        mMostrarString msgUExist
    nUexist:; no posee error de este tipo

    ;CARACTERES ESPECIALES NO PERMITIDOS PRESENTES
    mComparar caractNp,0
```

```
        je nCnexist
        yCnexist: ; error carateres especiales no permitidos presentes
            mMostrarString msgCaractP
        nCnexist: ; no posee error de este tipo

        ;PASSWORD SIN AL MENOS UNA MAYUSCULA
        mComparar mayuse,0
        je nPsm
        yPsm:; Password sin mayuscula
            mMostrarString msgunaM
        nPsm:; no posee error de este tipo

        ;PASSWORD SIN AL MENOS UN NUMERO
        mComparar nume,0
        je nPsn
        yPsn: ; Password sin numero
            mMostrarString msgunN
        nPsn:; no posee error de este tipo

        ;PASWORD SI AL MENOS UN SIMBOLO ESPECIAL(!>%;*)
        mComparar sinCaractE,0
        je nPss
        yPss: ;password sin simbolos
            mMostrarString msgunS
        nPss:; no posee error de este tipo

        ;PASSWORD CON LONGITUD ERRONEA
        mComparar largoe2,0
        je nPlongitud
        yPlongitud:; hay error respecto a la longitud
            mMostrarString msglengtherror2
        nPlongitud:; no posee error de este tipo

        call pEspEnter
        jmp salir

    noErrorRegistro: ;registro sin error
        call pAlmacenaruser
        mMostrarString RUSucces
        call pEspEnter
    salir:
    ret
pRegistrar endp
```

mUserInicial: verifica si la primera letra del usuario a ingresar es un numero o no.

```
mUserInicial macro
    local iniNumero,iniLetra,salir
    mEnRango UsuarioRegis[0],30h,39h  ; el dato se encuentra entre 0-9 del codigo ascii?
    cmp enrango,0     ;no
    je iniLetra       ;inicia con una letra y otro caracter, no hay error
    iniNumero:        ;inicia con un numero, si hay error
        mov numinicio,1  ;marca con 1 la variable la variable global que indica un error sobre un numero inicial
        mov eerror,1 ;indica que hay un error en el usuario o en la contraseña por tal razon no se registra
        jmp salir

    iniLetra:
        mov numinicio,0 ;marca que no hubo error
    salir:
endm
```

mSizeUser: verifica que el tamaño del nombre de usuario sea el correcto (8-15 caracteres).

```
mSizeUser macro
    local ciclosize,comparaciones,sentenciagood,salir,sentenciabad
    push si
    mov contadoraux,0 ;inicializa variable que contendra el tamaño del nombre user
    mov si, 0
    ciclosize:
        cmp si, 25t ;si ya llego a 25? (el tamaño maximo de la variable)
        je comparaciones ; si, pase a comprobar el tamaño del nombre user con los rangos
        mComparar UsuarioRegis[si],"$" ;cuando llegue a $ es que llego al fin del nombre usuario
        je comparaciones ;si es asi pasa a comparar el tamaño con los margenes permitidos
```

```
        mSumardb contadoraux,1 ;si no, suma uno al tamaño del nombre
user
        inc si ;incrementa uno al index si
        jmp ciclosize ; repite el ciclo
    comparaciones:
        mEnRango contadoraux, 8t,15t ; el tamaño esta entre 8 -15  ?
        mComparar enrango,1  ; esta en rango?
        je sentenciagood     ;si, la sentencia es correcta
        jne sentenciabad     ;no, la sentencia no es correcta
    sentenciagood:
        mov largoe, 0        ; no hay error respecto al rango
        jmp salir            ; sale
    sentenciabad:
        mov largoe,1         ; si hay error respecto al rango
        mov eerror,1         ; si hay error en name user o password
    salir:
    pop si
endm
```

mUserExisteR: verifica si el usuario existe o no.

```
mUserExisteR macro
    local Existe,Noexiste,salir,cicloexiste
    ;SE VERIFICA SI NO ES EL ADMIN
    mOpenFile usersb  ;abre el archivo en modo lectura
    mReadFile eleActual ;TOMA EL PRIMER VALOR DEL ARCHIVO
    mEncontrarId UsuarioRegis;lo primero en el documento de usuarios es
el admin, que siempre estara aca
    cmp idEncontrado,1 ; se encontro usuario?
    je Existe ;si se encontro se procede a decir que si existe el
usuario y se marcara como error
    cicloexiste: ;caso contrario se procedera a un ciclo de lectura del
archivo hasta hallar o un espacio o el id buscado
        mHallarSimbolo 0A  ;se salta hasta el enter hasta la posicion
donde esta 0A
        mReadFile eleActual ; se corre una vez el elemento
        cmp eleActual," " ;si hay un espacio es que ya se llego al fin
del documento y el usuario no existe
        ;CADA VEZ QUE SE CREA UN USUARIO SE ELIMINA EL ULTIMO ESPACIO
QUE DEJA LA CREACION DEL USUARIO ANTERIOR
        je Noexiste ; no existe usuario
```

```
        mEncontrarId UsuarioRegis ;si no es espacio lo que esta en esta
posicion fijo es un nombre de user, el user a registrar  es igual a
este?
        cmp idEncontrado,1 ; si, entonces existe
        je Existe
        jne cicloexiste
    Existe:
        mov existee,1 ;se reporta error pues existe usuario que se
intenta registrar
        mov eerror,1  ;se reporta error general al registro
        call pCloseFile
        jmp salir
    Noexiste:
        mov existee, 0 ; no existe usuario, no hay error
        call pCloseFile
    salir:
endm
```

mEncontrarId: busca si en una posición actual del cursor en un archivo leído existe una cadena igual a una cadena en especifico.

```
mEncontrarId macro id
    local salir ,comparar,idhallado,idNohallado,finid
    push si
    mov idEncontrado,0 ;limpiar idEncontrado
    mov si,0
    comparar:
    mComparar id[si],"$" ;llego al final del id escogido
    je finid
    mComparar id[si],eleActual   ; la letra obtenida es igual al
elemento actual del id?
    jne idNohallado ; no, entonces no son el mismo id, id no se hallo
    ;si, se procede a seguir recorriendo
    mReadFile eleActual
    inc si
    jmp comparar
    finid: ;llego al fin del id, pero llego al fin del usuario leido en
el doc?
        cmp eleActual,"$"
        je idhallado ;si entonces si es ese el id
        jne idNohallado ; los id no son iguales
    idhallado:
        MovVariables eleActual,0;LIMPIEZA DE VARIABLE
```

```
        mov idEncontrado,1 ;INDICA SI EL ID FUE ENCONTRADO
        jmp salir
    idNohallado:
        MovVariables eleActual,0;LIMPIEZA DE VARIABLE
        mov idEncontrado,0 ;INDICA SI EL ID FUE ENCONTRADO
    salir:
    pop si
endm
```

mRequisitoCletra: verifica que cada letra del nickname del usuario este entre los rangos permitidos.

```
mRequisitoCletra macro
    local ciclo,sicumpleR,nocumpleR,salir
    push si
    mov si,0 ;inicializa si
    ciclo:
        cmp si,25t  ;si llego hasta 25? (el tamaño maximo deuna
variable)
        je sicumpleR ;si, entonces no paso ningun error por lo tanto
todas las letras en el username cumplen con las restricciones
        mComparar UsuarioRegis[si], "$" ; llego hasta $?
        je sicumpleR ;si llego hasta $ significa que no hay errores en
los caracteres del name user
        mEnRangoEsp UsuarioRegis[si] ;revisa que esta letra cumpla con
los requisitos de los margenes
        cmp enrango,1 ;esta en rango, entre los caracteres permitidos
        jne nocumpleR ;no, no lo esta
        inc si  ;si , si lo esta
        jmp ciclo
    sicumpleR:
        mov caractNp,0 ; no hay errores entre los caracteres permitidos
        jmp salir
    nocumpleR:
        mov caractNp,1 ;hay error y hay caracteres no permitidos
        mov eerror,1    ; error general
    salir:
    pop si
endm
```

mAmayus: Verifica que exista al menos una mayúscula en la contraseña.

```
mAMayus macro
    local ciclopassword, salir, tieneMayus,noTieneMayus
    push si
    mov si, 0 ; se inicializa si
    ciclopassword:
        cmp si,25t ; si llego a 25 es que no habia ningun caracter con
mayusculas
        je noTieneMayus
        mEnRango PasswordRegis[si],41h,5Ah  ; esta esta letra entre el
rango de las mayusculas
        cmp enrango,1 ;esta en el rango?
        je tieneMayus ; si  se pasa a indicar que si tiene mayusculas
por lo tanto no hay error
        inc si
        jmp ciclopassword
    tieneMayus:
        mov mayuse,0 ; se procede a decir que o hay error en la falta
de mayusculas
        jmp salir
    noTieneMayus:
        mov mayuse,1 ;falta mayusculas
        mov eerror,1 ;hay error en usuario o password
    salir:
    pop si
endm
```

mAnum: verifica que existe al menos un numero en la contraseña a ingresar.

```
mANum macro
    local ciclopassword, salir, tieneCaracter,noTieneCAracter
    push si
    mov si, 0 ; se inicializa si
    ciclopassword:
        cmp si,25t ; si llego a 25 es que no habia ningun caracter con
numero
        je noTieneCAracter ; se salta a indicar que hay error y no
cumple con tener al menos un numero
        mEnRango PasswordRegis[si],30h,39h  ;esta en el rango de
numeros?
```

```
        cmp enrango,1 ;Si
        je tieneCaracter ;se procede a indicar que existe al menos un
numero
        inc si  ;se busca si y se compara el siguiente elemento
        jmp ciclopassword
    tieneCaracter:
        mov nume ,0 ;hay al menos un numero
        jmp salir
    noTieneCAracter:
        mov nume,1 ;no hay numeros
        mov eerror,1 ;error
    salir:
    pop si
endm
```

mAsigno: Verifica que la contraseña tenga al menos uno de los signos requeridos.

```
mASigno macro
    local ciclopassword, salir, tieneCaracter,noTieneCAracter
    push si
    mov si, 0
    ciclopassword:
        cmp si,25t ; si llego a 25 es que no habia al menos un caracter
especificado en el password
        je noTieneCAracter
        mComparar PasswordRegis[si],"!"
        je tieneCaracter
        mComparar PasswordRegis[si],">"
        je tieneCaracter
        mComparar PasswordRegis[si],"%"
        je tieneCaracter
        mComparar PasswordRegis[si],59t ;59t = puntoycoma
        je tieneCaracter
        mComparar PasswordRegis[si],"*"
        je tieneCaracter
        inc si
        jmp ciclopassword
    tieneCaracter:
        mov sinCaractE ,0
        jmp salir
    noTieneCAracter:
        mov sinCaractE,1
```

```
        mov eerror,1
    salir:
    pop si
endm
```

mSizePassword: Verifica que la contraseña tenga la longitud correcta (16 a 20 caracteres).

```
mSizePassword macro
    local ciclosize,comparaciones,sentenciagood,salir,sentenciabad
    push si
    mov contadoraux,0 ; se inicializa la variable que contendra el
tamaño del password
    mov si, 0
    ciclosize:
        cmp si,25t ; si llego a 25 (maximo tamaño para una password )
pasa a proceder a verificar el tamaño
        je comparaciones ;pasa a comparar con los margenes
        mComparar PasswordRegis[si],"$" ;llego hasta $, significa que
llego al fin del tamaño del password
        je comparaciones ;pasa a comparar con los margenes
        mSumardb contadoraux,1
        inc si
        jmp ciclosize
    comparaciones:
        mEnRango contadoraux, 16t,20t ;el tamaño esta entre 16 y 20
caracteres?
        mComparar enrango,1
        je sentenciagood ;si, longitud de password correcta
        jne sentenciabad ;no, longitud de password incorrecta
    sentenciagood:
        mov largoe2 , 0 ;no hay error en el rango
        jmp salir
    sentenciabad:
        mov largoe2 ,1 ;si hay eror en el rango
        mov eerror,1 ; hay error en usuario o contraseña
    salir:
    pop si
endm
```

MACROS Versátiles:

mEnRango: Verifica que un elemento se encuentre entre un rango especificado.

```
mEnRango macro dato,limif, limsup
    local enElrango,noEnelrango,salir
    ;ja >,jb <,  jbe<=
    mComparar dato,limif
    jb noEnelrango ; si es menor al limite inferior no esa en el rango
    mComparar dato,limsup
    jbe enElrango ; si es menor o igual al limite superior esta en el
rango
    ja noEnelrango; si es mayor no esta en el rango
    enElrango:
        MovVariables enrango,1
        jmp salir
    noEnelrango:
        MovVariables enrango,0
    salir:
endm
```

mMostrarString: Imprime un string con un “$” al final.

```
mMostrarString macro var
    push dx
    push ax
    xor ax,ax
    mov dx,ax
    lea dx, var
    mov ah, 09
    int 21
    pop ax
    pop dx
endm
```

String2Num: Convierte un string a decimal.

```
String2Num macro stringToRead,whereToStore,simbol2stop
    local readStringValue
    push ax
    push cx
    push dx
    push bx
```

```
    push si
    xor dx,dx ;limpia dx y la vuelve 0
    mov ax,dx ;ax = 0
    mov si,dx ;si = 0
    mov cx,dx
    mov bx, 0A ;bx=10
    ;mov si, offset stringtoRead
    readStringValue:
    mov cl,stringToRead[si]
    sub cl,30h ; se le resta 30 para convertirlo a un numero legible
(num de 0-9)
    mul bx    ; ax= ax*bx  se multiplica por 10 el valor actual de ax
    add ax,cx ; se suma a ax el valor de cx
    inc si
    cmp stringToRead[si],simbol2stop ; simbolo para saber cuando
finalizo lo relevante de la cadena y parar de convertir
    jne readStringValue
    mov whereToStore,ax
    pop si
    pop bx
    pop dx
    pop cx
    pop ax
endm
```

Num2String: Convierte un numero a un string legible.

```
Num2String macro numero, stringvar  ;stringvar: variable donde se
almacenara el numero
    local cNumerador,Convertir
    push ax
    push bx
    push dx
    push si

    mov contador,0
    mov bx,0A
    mov ax, numero
    cNumerador:   ;condicion de numerador
        xor dx,dx
        div bx
        push dx
        inc contador ;tamaño de la pila, aumenta al agregarse un valor
```

```
        cmp ax, 0 ;numerador es igual a 0?
        jne cNumerador
    mov si, offset stringvar; donde se almacenara el nuevo numero
    Convertir:
        pop dx  ;pop = pila.pop(ultimo valor)
        add dx,30h
        mov [si],dx
        inc si
        dec contador
        cmp contador,0
        jne Convertir
    pop si
    pop dx
    pop bx
    pop ax
endm
```

mCapturarString: Captura una cadena de un documento que se esta leyendo actualmente, en un espacio de este en especifico.

```
mCapturarString macro variableAlmacenadora
    local salir,capturarLetras,deletCaracter
    push ax
    push si
    mov si,0
    capturarLetras:
        mov ah,01h
        int 21h
        cmp al, 0dh ;es igual a enter?
        je salir ; una vez dado enter y capturado todo el nombre, pasar
al siguiente procedimiento
        cmp al, 08 ;es igual a retroceso?
        je deletCaracter
        mov variableAlmacenadora[si],al
        inc si
        jmp capturarLetras
    deletCaracter:
        cmp si,0
        je capturarLetras
        mov variableAlmacenadora[si],24
        dec si
        mMostrarString espacio ; " "
        mMostrarString retroceso ;"<-"
```

```
        jmp capturarLetras
    salir:
        pop si
        pop ax
endm
```

mLimpiar: Rellena variables con su símbolo inicial, limpiandolas en el proceso.

```
mLimpiar macro lista,numero,signo
    local salir,borrar
    push si
    mov si,0
    borrar:
        mov lista[si],signo
        inc si
        cmp si,numero
        je salir
        jne borrar
    salir:
    pop si
endm
```

mMovVariables: Permite mover variables entre variables, y brinda una mayor seguridad al realizar otros mov.

```
MovVariables macro var1,var2
    push dx
    mov dl,var2
    mov var1, dl ; SE INGRESA A LA NUEVA POSICION EL SIMBOLO ACTUAL
    pop dx
endm
```

mComparaStrings: Compara dos cadenas e indica si ambas son iguales o no a través de una variable llamada cadIguales como resultado (1 si son iguales, 0 no son iguales).

```
mCompararStrings macro var1, var2
    local salir,Iguales,noIguales,comparar,pfvar1,pfvar2
    push si
    mov cadIguales,0
```

```
    mov si,0
    comparar:
        mComparar var1[si],var2[si]
        je Pfvar1
        jne noIguales
    pfvar1:
        mComparar var1[si],"$" ;cadena llego al final?
        je pfvar2 ;tambien llego al final en la cadena 2?
        inc si
        jne comparar
    pfvar2:
        mComparar var2[si],"$"
        je Iguales ;si llego al final al mismo tiempo que var 1, son
iguales
        jne noIguales ;no son iguales
    Iguales:
        mov cadIguales,1
        jmp salir
    noIguales:
        mov cadIguales,0
        jmp salir
    salir:
    pop si
endm
```

mComparar: Permite comparar variables entre variables y brinda una mayor seguridad al comparar otros valores.

```
mComparar macro var1,var2
    push ax
    push bx
    xor ax,ax
    mov bx,ax
    mov al,var1
    mov bl,var2
    cmp al,bl
    pop bx
    pop ax
endm
```

mMoverVariablesDw: Mueve variables de tamaño word.

```
MovVariablesDw macro var1,var2
```

```
    push dx
    mov dx,0
    mov dx,var2
    mov var1, dx ; SE INGRESA A LA NUEVA POSICION EL SIMBOLO ACTUAL
    pop dx
endm
```

mCrearFile: Macro para crear archivos externos.

```
mCrearFile macro nameFile
    local falloCT,salidaCT,salir
    push ax
    mov cx,0
    lea dx, nameFile
    mov ah, 3C
    int 21
    jc falloCT
    mov handler, ax
    jmp salidaCT
    falloCT:
        mMostrarString savebad
        mov creacionCorrecta,0
        jmp salir
    salidaCT: ;sale del bucle
        ;mMostrarString savegood
        mov creacionCorrecta,1
        jmp salir
    salir:
    pop ax
endm
```

mWriteToFile: Permite escribir un string en un archivo actualmente abierto.

```
mWriteToFile macro palabra
    push ax
    push bx
    push cx
    push dx
    mov bx, handler
    mov cx, LENGTHOF palabra
    mov dx, offset palabra
    mov ah,40
```

```
    int 21
    pop dx
    pop cx
    pop bx
    pop ax
endm
```

mReadFile: Permite leer un carácter de un archivo leído actualmente, con cada llamada de esta macro aumenta la posición del cursor de lo actualmente leido en uno.

```
mReadFile macro varAlmacenadora
    push ax
    push bx
    push cx
    push dx
    mov bx, handler
    mov cx, 1
    lea dx, varAlmacenadora ; esto seria igual a:  mov dx, offset
lectura, "EN LA POSICION DE LECTURA GRABAR LO LEIDO"
    mov ah, 3F
    int 21
    mov posLectura, ax
    pop dx
    pop cx
    pop bx
    pop ax
endm
```

mOpenFile2Write: Permite abrir un archivo para que pueda ser leído y escrito.

```
mOpenFile2Write macro fileName
    local errorOpen,Opencorrecto,salidaOpen
    push ax
    push dx
    mov estadocarga,0
    mov al,2
    lea dx, fileName
    mov ah,3Dh
    int 21
```

```
    jc errorOpen
    mov handler, ax
    jmp Opencorrecto
    errorOpen:
        mMostrarString carbad
        mov estadocarga,0
        jmp salidaOpen
    Opencorrecto:
        ;mMostrarString cargood
        mov estadocarga,1
        jmp salidaOpen
    salidaOpen:
    pop dx
    pop ax
endm
```

mHallarSimbolo: busca en el documento hasta hallar un símbolo que sea el mismo que el requerido.

```
mHallarSimbolo macro simbolo
    local buscar,salir
    buscar:
        mReadFile eleActual
        cmp posLectura,0  ;"LLEGO AL FINAL DEL DOCUMENTO?"
        je salir; si llego, salir del metodo sino seguir comparando
        mComparar eleActual,simbolo ;buscando el simbolo buscado, si se
hallo ya no se manda al ciclo buscar y se sale
        jne buscar
    salir:
endm
```

mDelayt: Macro que crea un delay de n cantidad de segundos, dicho valor se debe de especificar.

```
mDelayt macro tiempo
    local ciclodelay,segundo,salir
    push ax
    push dx
    mov valort1,0
    mov auxt, 0 ;borrar
    mov contDb,0
```

```
    cmp tiempo,0 ;si el delay es de 0 salir por que indica que no hay
delay por hacer
    je salir
    mov ah,2Ch
    int 21h
    mov valort1,dh  ;VALOR 1 TOMA UN TIEMPO INICIAL
    ciclodelay:
        ;segunda toma de tiempo
        mov ah,2Ch
        int 21h
        mComparar valort1,dh  ;los tiempos son distintos (si es asi
paso un segundo)
        jne segundo ;SI ESE ES EL CASO PASA A UN APARTADO DE CUANDO
PASO 1 SEGUNDO
        jmp ciclodelay
        segundo:
            cmp contDb,tiempo ;CONTADOR ES IGUAL A EL TIEMPO REQUERIDO?
            je salir  ;SI, SALIR
            mov valort1,dh ;si no es asi, mover el tiempo actual a la
variable tiempo y repetir ciclo
            inc contDb; SE LE SUMA UNO AL CONTADOR
            jmp ciclodelay
    salir:
        pop dx
        pop ax
endm
```

mDelaytCenti: Macro que crea un delay de n cantidad de centisegundos, dicho valor se debe de especificar.

```
mDelaytCenti macro tiempo
    local ciclodelay,centisegundo,salir
    push ax
    push dx
    push bx
    push cx
    mov valort1,0
    mov auxt, 0 ;borrar
    mov contDb,0
    mov ah,2Ch
    int 21h
    mov valort1,dl  ;VALOR 1 TOMA UN TIEMPO INICIAL
```

```
    ciclodelay:
        ;segunda toma de tiempo
        mov ah,2Ch
        int 21h
        mComparar valort1,dl  ;los tiempos son distintos (si es asi
paso un centisegundo)
        jne centiSegundo ;SI ESE ES EL CASO PASA A UN APARTADO DE
CUANDO PASO 1 centisegundo
        jmp ciclodelay
        centiSegundo:
            call pTimeOrd ;SI SE QUIERE USAR ESTA MACRO PARA CUALQUIER
OTRA COSA, BORRAR ESTA LINEA
            mComparar contDb,tiempo ;CONTADOR ES IGUAL A EL TIEMPO
REQUERIDO?
            je salir  ;SI, SALIR
            mov valort1,dl ;si no es asi, mover el tiempo actual a la
variable tiempo y repetir ciclo
            inc contDb; SE LE SUMA UNO AL CONTADOR
            jmp ciclodelay
    salir:
        pop cx
        pop bx
        pop dx
        pop ax
endm
```

mDrawPixel: macro para pintar un pixel en la ventana, con el modo vídeo ya activado.

```
mDrawPixel macro line,column,color
    push ax
    push bx
    push dx
    push si
    xor ax,ax
    xor bx,bx
    xor dx,dx
    xor si,si

    ;formula para pintar un pixel de la matriz video = ((linea-1) *
320) + (columna-1)
    mov ax,line
    dec ax
```

```
    mov bx, 320t
    mul bx
    ;en ax ya tengo el resultado del primer parentesis
    add ax, column
    dec ax

    mov si, ax
    mov bl,color
    mov es:[si],bl

    pop si
    pop dx
    pop bx
    pop ax
endm
```

mDrawRectangulo: Macro para dibujar un rectángulo en el modo video.

```
mDrawRectangulo macro x,y,ancho,alto,color
    local lineasup,barraslat,lineainf
    push cx
    push bx
    xor cx,cx
    xor bx,bx
    mov bx,y  ;auxiliar que tendra almacenada la variable y
    mov cordx,x
    mov cordy,y

    mov cx,ancho
    lineasup: ;se grafica la linea superior, imprimiendo y aumentando
las columnas para generar una linea
        mDrawPixel cordx,cordy,color
        inc cordy
        loop lineasup
    mov cordy,bx ; se regresa cordy a su valor original
    inc cordx ;se pasa a la siguiente fila

    mov cx,alto ; se hara el siguiente procedimiento hasta que se
cumpla el alto establecido
    barraslat: ;se grafican las barras laterales
        mDrawPixel cordx,cordy,color
        mSumarDw cordy,ancho
        dec cordy
```

```
        mDrawPixel cordx,cordy,color
        mov cordy,bx ;una vez hecho las dos impresiones siempre volver
al valor original
        inc cordx
        loop barraslat
    mov cx,ancho
    lineainf:
        mDrawPixel cordx,cordy,color
        inc cordy
        loop lineainf
    mov cordx,0
    mov cordy,0
    pop bx
    pop cx
endm
```

mImprimirLetreros: Macro para imprimir strings en modo video.

```
mImprimirLetreros macro letrero,fila,columna,color
    push ax
    push bx
    push cx
    push dx
    push bp
    mov al,1   ;MODO DE IMPRESION CON COLOR (1), SIN COLO(0)
    mov bh,0   ;PAGINA
    mov bl,color  ;COLOR  (PALETA VGA 1t-255t)
    mov cx,LENGTHOF letrero ;tamaño del letrero
    mov dl,columna ;columna
    mov dh,fila ;fila
    call pDataS_ES ;se puede realiar esto o el procedimiento de abajo
siempre y cuando ds tenga el valor de @data
    ;push ds  ;meto el valor de ds a la pila (que contiene el data
segment)
    ;pop es  ;dicho valor se saca de la pila y se asigna a "es"
    mov bp,offset letrero
    mov ah,13h
    int 10h
    call pMemVideoMode;SE VUELVE A COLOCAR LA MEMORIA DE VIDEO EN ES
    pop bp
    pop dx
    pop cx
    pop bx
```

```
    pop ax
endm
```

mWaitKey: Espera a que se presione una tecla y la lee.

```
mWaitKey macro key
    local ciclo
    push ax
    ciclo:
    mov ah, 00  ;Espera a que se presione una tecla y la lee
    int 16h
    cmp al,key
    jne ciclo
    pop ax
endm
```

mCapturarStringDoc: Captura un string en la posición actual leida del documento abierto.

```
mCapturarStringDoc macro variableAlmacenadora
    local salir,capturarString
    push si
    mov si,0
    capturarString:
        MovVariables variableAlmacenadora[si],eleActual
        inc si
        mReadFile eleActual
        cmp eleActual,1 ;es igual a 1 ASCII (no es igual al 1 decimal
no afecta a los numeros)?
        je salir  ; si, terminar de capturar
        cmp eleactual,00
        je salir
        cmp eleActual,0A ;es igual a enter tipo1
        je salir  ; si, terminar de capturar
        cmp eleActual," " ;los 0's impresos en un documento externo se
vuelven espacios
        je salir  ; si, terminar de capturar
        jmp capturarString
    salir:
    pop si
endm
```

mDrawBarra: Dibuja una barra para ser usada en los ordenamientos y aspectos visuales.

```
mDrawBarra macro x,y,alto,ancho,color
    local cicloAncho,cicloAlto,no0x,no0y
    push ax
    push dx
    push cx

    movVariablesDw cordx,x
    movVariablesDw cordy,y
        cmp cordx,0
        jne no0x
        mov cordx,1
        no0x:
        cmp cordy,0
        jne no0y
        mov cordy,1
        no0y:
    ;ancho de x1 a x2
    ;alto de y1 a y2
    mov ax,x
    mov dx,y
    mov cx,ancho
    cicloAncho:
    push cx
        mov cx, alto
        cicloAlto:
            mDrawPixel cordx,cordy,color
            inc cordx
        loop cicloAlto
        mov cordx,ax
    pop cx
    inc cordy
    loop cicloAncho
    mov cordy, dx
    pop cx
    pop dx
    pop ax
endm
```

mMoverAFila: Le indica al programa en que fila del documento abierto se debe de colocar.

```
mMoverAFila macro fila
    local ciclo,salir
    call pInidoc ;POSICIONA EL CURSOR EN EL INICIO DEL DOCUMENTO LEIDO
    push cx
    cmp fila,0
    je salir
    mov cx,fila
    ciclo:
        mHallarSimbolo 0A
    loop ciclo
    salir:
    pop cx
endm
```

mCapturarFilaDoc: Captura la fila completa de un documento como string en una variable.

```
mCapturarFilaDoc macro varAlmacenadora
    local salir,capturarString,noseparador,separador
    push si
    mov si,0
        mReadFile eleactual
    capturarString:
        cmp eleactual,01 ;separador
        jne noseparador
        MovVariables varAlmacenadora[si],09h ;tabulacion
        jmp separador
        noseparador:
        MovVariables varAlmacenadora[si],eleActual
        separador:
        inc si
        mReadFile eleActual
        cmp eleActual,0A ;es igual a enter tipo1
        je salir  ; si, terminar de capturar
        cmp eleActual,0dh ;es igual a enter tipo1
        je salir  ; si, terminar de capturar
        jmp capturarString
    salir:
    pop si
endm
```

MENÚS:

Menu de Admin:

```
pMenuAdmin proc
    push ax
    mOpenFile2Write usersb
    ciclomenu:
    call pResetFlagsE
    mMostrarString msgMenuAdmin ;MENU ADMIN
    call pImprimirUser ;IMPRIMIR NOMBRE DE USUARIO ACTUAL
    mMostrarString MenuAdmin  ;MUESTRA EL MENU
    mov opcion,0
    ;la laptop que se posee para trabajar esto necesita de presionar
una tecla antes de los FN
    ; para que los reconozca por tal motivo se hizo esto dos veces para
que se pudiera a trapar el valor de Fn
    mov ah,01
    int 21;atrapa la tecla fn
    mov ah,01  ;atrapa otras teclas
    int 21
    mov opcion,al
    cmp opcion, 59t
    je unlockUser
    cmp opcion, "<"
    je darAdmin
    cmp opcion, "="
    je quitarAdmin
    cmp opcion, "?"
    je Bublesort
    cmp opcion, "@"
    je heapsort
    cmp opcion, "A"
    je Timsort
    cmp opcion, "C"
    je salir
    mMostrarString opi
    jmp ciclomenu
    unlockUser:
        call pQuitarbloqueo
        call pLimpiarConsola
        jmp ciclomenu
    darAdmin:
        call pDarAdmin
```

```
        call pLimpiarConsola
        jmp ciclomenu
    quitarAdmin:
        call pQuitarAdmin
        call pLimpiarConsola
        jmp ciclomenu
    Bublesort:
        call pMenuOrd
        jmp ciclomenu
    heapsort:
        jmp Bublesort ;CAMBIAR ESTA LINEA SI DICHO ORDENAMIENTO HUBIERA
SIDO TRABAJADO
    Timsort:
        jmp Bublesort ;CAMBIAR ESTA LINEA SI DICHO ORDENAMIENTO HUBIERA
SIDO TRABAJADO
    salir:
        pop ax
    ret
pMenuAdmin endp
```

Menu de Usuario:

```
pMenuUser proc
    menuUser:
    ;MENU DE USUARIO
    mMostrarString msgMenuU  ;MENU DE USUARIO NORMAL
    call pImprimirUser ;IMPRIMIR NOMBRE DE USUARIO ACTUAL
    mMostrarString MenuUsuario ;IMPRESION DE MENU USUARIO
    mov opcion,0
    ;la laptop que se posee para trabajar esto necesita de presionar
una tecla antes de los FN
    ; para que los reconozca por tal motivo se hizo esto dos veces para
que se pudiera a trapar el valor de Fn
    mov ah,01
    int 21
    mov ah,01  ;atrapa la tecla fn
    int 21
    mov opcion,al
    cmp opcion, "<"
    je game
    cmp opcion, "="
    je totalscorboard
```

```
    cmp opcion, "?"
    je myscorboards
    cmp opcion, "C"
    je salir
    mMostrarString opi
    jmp menuUser
    game:
        call pGame ;llama al juego
        call pAlmacenarScore ;almacena el score
        jmp menuUser
    totalscorboard:
        call pShowtop10
        jmp menuUser
    myscorboards:
        call pShowMytop10
        jmp menuUser
    salir:
    ret
pMenuUser endp
```

Menú de Usuario Admin:

```
pMenuU_admin proc
    ciclomenu:
    mMostrarString msgMuA   ;TITULO: MENU DE USUARIO ADMIN
    call pImprimirUser ;IMPRIMIR NOMBRE DE USUARIO ACTUAL
    mMostrarString MenuUsuarioAdmin ;OPCIONES DEL MENU DE USUARIO ADMIN
    mov opcion,0
    ;la laptop que se posee para trabajar esto necesita de presionar
una tecla antes de los FN
    ; para que los reconozca por tal motivo se hizo esto dos veces para
que se pudiera a trapar el valor de Fn
    mov ah,01
    int 21
    mov ah,01  ;atrapa la tecla fn
    int 21
    mov opcion,al
    cmp opcion, 59t
    je unlockUser
    cmp opcion, "<"
    je totalscorboard
    cmp opcion, "="
```

```
    je myscorboards
    cmp opcion, ">"
    je game
    cmp opcion, "?"
    je Bublesort
    cmp opcion, "@"
    je heapsort
    cmp opcion, "A"
    je Timsort
    cmp opcion, "C"
    je salir
    mMostrarString opi
    jmp ciclomenu
    unlockUser:
        call pQuitarbloqueo
        call pLimpiarConsola
        jmp ciclomenu
    totalscorboard:
        call pShowtop10
        jmp ciclomenu
    myscorboards:
        call pShowMytop10
        jmp ciclomenu
    game:
        call pGame ;llama al juego
        call pAlmacenarScore ;almacena el score
        jmp ciclomenu
    Bublesort:
        call pMenuOrd
        jmp ciclomenu
    heapsort:
        jmp Bublesort ;CAMBIAR ESTA LINEA SI DICHO ORDENAMIENTO HUBIERA
SIDO TRABAJADO
    Timsort:
        jmp Bublesort ;CAMBIAR ESTA LINEA SI DICHO ORDENAMIENTO HUBIERA
SIDO TRABAJADO
    salir:
    ret
pMenuU_admin endp
```

pDelay30t: Procedimiento de 30 segundos que imprime el tiempo que lleva cada segundo.

```
pDelay30 proc
    push ax
    push dx
    mov valort1,0
    mov contadort,0
    ;SE TOMA EL VALOR DE T1
    mov ah,2Ch
    int 21h
    mov valort1,dh  ;VALOR 1 TOMA UN TIEMPO INICIAL
    ciclodelay:
        mov ah,2Ch
        int 21h
        mComparar valort1,dh ;EL CICLO SE REPETIRA HASTA QUE SEAN
DISTINTOS
        jne segundo ;ES DISTINTO POR LO CUAL YA CAMBIO DE SEGUNDO
        jmp ciclodelay
        segundo:
            mLimpiar StringNumT,4,24 ;SE LIMPIA EL STRING QUE
ALMACENARA EL SEGUNDO
            Num2String contadort,StringNumT ;SE PASA EL CONTADOR ACTUAL
A STRING
            mMostrarString StringNumT ;SE IMPRIME EL STRING DEL
CONTADOR
            cmp contadort,30t ;CONTADOR ES IGUAL A 30?
            je salir  ;SI, SALIR
            MovVariables valort1,dh ;NO, ENTONCES VALORT1=auxt (que
contiene el valor2 sin el efecto de mod)
            inc contadort ; SE LE SUMA UNO AL CONTADOR
            jmp ciclodelay
    salir:
        pop dx
        pop ax
    ret
pDelay30 endp
```

PROCEDIMIENTOS PARA LEER UN DOCUMENTO DE FORMA MÁS EXTENSA:

pInidoc: coloca el cursor del documento abierto en el inicio del mismo.

```
pInidoc proc
    push ax
```

```
    push bx
    push cx
    push dx
    mov al,0
    mov bx,handler
    mov cx,0
    mov dx,0
    mov ah,42h
    int 21
    pop dx
    pop cx
    pop bx
    pop ax
    ret
pInidoc endp
```

pFinaldoc: coloca el cursor del documento abierto en el final del mismo.

```
pFinaldoc proc
    push ax
    push bx
    push cx
    mov al,2
    mov bx,handler
    mov cx,-1
    mov dx,-1
    mov ah,42h
    int 21
    pop cx
    pop bx
    pop ax
    ret
pFinaldoc endp
```

pPosAnterior: Moverse una posición anterior de la posición actual.

```
pPosAnterior proc
    mov al,1
    mov bx,handler
    mov cx,-1
    mov dx,-1
```

```
    mov ah,42h
    int 21
    ret
pPosAnterior endp
```

pClosFile: Cierra el archivo leído.

```
pCloseFile proc
    push bx
    push ax
    mov bx, handler
    mov ah, 3Eh
    int 21
    pop ax
    pop bx
    ret
pCloseFile endp
```

PROCEDIMIENTOS PARA EL JUEGO, ORDENAMIENTOS Y MANIPULACIÓN DEL PROGRAMA:

pDarbloqueo: Da un bloqueo al usuario que se halla equivocado mas de 3 veces al ingresar en el login.

```
pDarbloqueo proc
    ;ya que dar bloqueo va despues de incrementar veces que se equivoco
solo se busca una vez el separador
    cmp eleActual,33h
    jne nobloquear
    mHallarSimbolo separador
    mWriteToFile BloqueoU
    jmp salir
    nobloquear: ;si no se bloquea al menos se debe de desplazar la
misma cantidad de espacios de como
    ;si se hubiera bloqueado
        mHallarSimbolo separador
        mReadFile eleActual
    salir:
    ret
pDarbloqueo endp
```

pQuitarbloqueo: Permite a los administradores quitar el bloqueo de los usuarios que se equivocaron 3 veces al ingresar en el login.

```
pQuitarbloqueo proc
    call pLimpiarConsola
    mMostrarString msgUnlockUserT ;IMPRIMIR TITULO DEL APARTADO
    call pImprimirUser ;IMPRIMIR USUARIO ACTUAL
    mLimpiar Umoderado,25,24
    mOpenFile2Write usersb
    call pResetFlagsE
    mMostrarString usDesBloq
    mCapturarString Umoderado; CAPTURAR STRING DE USUARIO
    call pExisteUserM
    cmp existee,0 ;no existe el usuario ingresado?
    je Unoexiste ; no existe, entonces marca error y se sale
    ;SI EXISTE, ENTONCES EL ARCHIVO YA SE ENCUENTRA POSICIONADO EN EL
ESPACIO DESEADO
    mHallarSimbolo separador; contraseña
    mHallarSimbolo separador ; n veces error
    mHallarSimbolo separador; B/n
    mReadFile eleActual
    cmp eleActual, "N" ; no bloqueado
    je noBloqAnt
    call pPosAnterior ; separador antes de B/n
    call pPosAnterior ;n veces error
    call pPosAnterior ; separador antes de n veces eror
    mWriteToFile Nequivdef
    mHallarSimbolo 01
    mWriteToFile Bloqdef
    mMostrarString Udesbloqueado
    call pEspEnter
    jmp salir
    noBloqAnt:
        mMostrarString Unoblock ;el usuario no estaba bloqueado
        call pEspEnter
        jmp salir
    Unoexiste:
    mMostrarString MsgUnE ;usuario no existe
    call pEspEnter
    jmp salir
    salir:
    call pCloseFile
    ret
```

```
pQuitarbloqueo endp
```

pDarAdmin: Asciende a admin a un usuario normal.

```
pDarAdmin proc
    call pLimpiarConsola
    mMostrarString msgPromoteAdmin;IMPRIMIR TITULO DEL APARTADO
    call pImprimirUser ;IMPRIMIR USUARIO ACTUAL
    mLimpiar Umoderado,25,24
    mOpenFile2Write usersb
    call pResetFlagsE
    mMostrarString usDarAdmin
    mCapturarString Umoderado; CAPTURAR STRING DE USUARIO
    call pExisteUserM
    cmp existee,0 ;no existe el usuario ingresado?
    je Unoexiste ; no existe, entonces marca error y se sale
    ;SI EXISTE, ENTONCES EL ARCHIVO YA SE ENCUENTRA POSICIONADO EN EL
ESPACIO DESEADO
    mHallarSimbolo separador; contraseña
    mHallarSimbolo separador ; n veces error
    mHallarSimbolo separador; B/n
    mHallarSimbolo separador; separador antes de Admin/No admin
    mReadFile eleActual
    cmp eleActual, "A" ; el elemento es admin
    je AdminAnt
    call pPosAnterior ; separador antes de Admin/No admin
    mWriteToFile AdminU
    mMostrarString Udadoadmin
    call pEspEnter
    jmp salir
    AdminAnt:
        mMostrarString Uadmin ;el usuario ya era admin
        call pEspEnter
        jmp salir
    Unoexiste:
    mMostrarString MsgUnE ;usuario no existe
    call pEspEnter
    jmp salir
    salir:
    call pCloseFile
    ret
pDarAdmin endp
```

pQuitarAdmin: Quita el admin a un usuario que previamente lo era.

```
pQuitarAdmin proc
    call pLimpiarConsola ;LIMPIA LA CONSOLA
    mMostrarString msgDemoteAdmin ;IMPRIMIR TITULO DEL APARTADO
    call pImprimirUser ;IMPRIMIR USUARIO ACTUAL
    mLimpiar Umoderado,25,24 ;Limpiar la variable donde se almacenara
el usuario a quitar admin
    mOpenFile2Write usersb ;ABRE EL ARCHIVO DE USUARIOS PARA LEER Y
ESCRIBIR
    call pResetFlagsE ;RESETEA TODAS LAS BANDERAS

    mMostrarString usQuitarAdmin ;MENSAJE PARA QUITAR ADMIN
    mCapturarString Umoderado; CAPTURAR STRING DE USUARIO
    mCompararStrings Umoderado,nameAdminG
    cmp cadIguales, 1
    je Admin_G

    call pExisteUserM ;EXISTE USUARIO?
    cmp existee,0 ;no existe el usuario
    je U_noexiste ; no existe, entonces marca error y se sale
    ;SI EXISTE, ENTONCES EL ARCHIVO YA SE ENCUENTRA POSICIONADO EN EL
ESPACIO DESEADO
    mHallarSimbolo separador; contraseña
    mHallarSimbolo separador ; n veces error
    mHallarSimbolo separador; B/n
    mHallarSimbolo separador; separador antes de Admin/No admin
    mReadFile eleActual
    cmp eleActual, "N" ; el elemento es admin para poder degradarlo?
    je AdminAnt ;el elemento no es admin, por tal motivo no es posible
degradarlo
    call pPosAnterior ; separador antes de Admin/No admin
    mWriteToFile admindef
    mMostrarString UquitAdmin
    call pEspEnter
    jmp salir
    AdminAnt:
        mMostrarString uNoAdmin ;el usuario no era admin
        call pEspEnter
        jmp salir
    U_noexiste:
        mMostrarString MsgUnE ;usuario no existe
        call pEspEnter
        jmp salir
```

```
    Admin_G:;se intento quitar el admin al admin general
        mMostrarString msgQuitAdminG
        call pEspEnter
        jmp salir
    salir:
    call pCloseFile
    ret

pQuitarAdmin endp
```

pTextMode:Coloca el programa en modo texto, el cual es el default de dosbox.

```
pTextMode proc
    push ax
    mov ax, 03
    int 10h
    pop ax
    ret
pTextMode endp
```

pVideoMode: Coloca el programa en modo video, para graficar.

```
pVideoMode proc
    push ax
    mov ax, 13
    int 10h
    pop ax
    ret
pVideoMode endp
```

pMemVideoMode: Enlaza a “es” la direccion de la memoria de video, para poder pintar los bytes de esta.

```
pMemVideoMode proc
    push dx
    mov dx, 0A000
    mov es,dx
    pop dx
    ret
pMemVideoMode endp
```

JUEGO

pMovimientoGame: Es el corazón del juego, le permite al juego moverse a 100 fps.

```
pMovimientoGame proc
    mov auxfpsT,0
    reset:
        call pConfigIni
    fps: ;ciclo que provoca un movimiento cada centisegundo
        mov ah,2Ch
        int 21
        cmp dl, auxfpsT
        je fps
    mov auxfpsT, dl
    call pDrawCleansCorazones
    call pDrawCorazones
    cmp nivelGame,4 ; si se finalizo el 3 nivel, nivelgame llegara a 4
indicando que finalizo el juego
    je gameover
    cmp liNave,0 ;si la vida de la nave llego a 0 es game over
    je gameover
    ;MOVIMIENTO

    call pLevel
    call pScore
    call pTimeGame
    ;MOVIMIENTOS DE LA NAVE
    call pMovNave
    call pDrawNaveBorr
    call pDrawNave
    cmp exitGame,1 ; si luego de una pausa se selecciono en salir del
juego
    je salir

    ;IMPRESION DE ENEMIGOS-NIVELES
    cmp printEnemyE,1 ;SI YA SE IMPRIMIO NO VOLVER A IMPRIMIR
    je yaimpresoEnemy
    ;SE ESPERA A QUE EL USUARIO PRESIONE ESPACIO PARA PODER IMPRIMIR
LOS ENEMIGOS Y SEGUIR CON EL RESTO DEL JUEGO
        call pEspInicial
        ;mDelayt 1t  ;DELAY ANTES DE EMEPEZAR CADA NIVEL, ESTE DELAY ES
DE 1 SEGUNDO
```

```
    call pDrawEnemigos ;SE IMPRIME ENEMIGOS
    mov printEnemyE,1 ;SE MARCA QUE YA SE IMPRIMIO
    ;COORDENADAS INICIALES PARA EL MOVIMIENTO DE ENEMIGOS
        mov ce_x,45t
        mMultiplicacionDw ce_x,nivelGame
        mov ce_y, 140t
        mov filaIgame,45t  ;auxiliar que contendra la fila actual
recorrida
        mMultiplicacionDw filaIgame,nivelGame
    yaimpresoEnemy:
    call pMovEnemys ;mover enemigos

    ;MOVIMIENTO DE BALAS
    ;BALA 1
        cmp estD1,0 ;no se tiene permitido imprimir la bala
        je sinAccion
        ;si se tiene permitido disparar la bala
            call pMovbala
        sinAccion:
        cmp nivelGame,1
        je fps
    ;BALA 2
        cmp estD2,0
        je sinAccion2
        ;si se tiene permitido disparar la bala
            call pMovbala2
        sinAccion2:
        cmp nivelGame,2
        je fps
    ;BALA 3
        cmp estD3,0
        je sinAccion3 ; si esta en 0 nos e tiene permitido mover la
bala
            call pMovbala3;si se tiene permitido mover la bala
        sinAccion3:
        jmp fps
    gameover:
        mImprimirLetreros letGover,8t,23t,15t ;imprimir letrero de game
over
        mImprimirLetreros letEsp,12t,18t,15t ;mensaje indicando accion
a realizar
        mWaitKey " "
    salir:
```

```
    ret
pMovimientoGame endp
```

pDrawCorazon: Procedimiento para dibujar un corazon.

```
pDrawCorazon proc
    push ax
    push dx
    mov ax,corazonx
    mov dx, corazony
    ;punta corazon
    mDrawPixel corazonx,corazony,39t ;rojo
    ;fila de arriba
    dec corazonx
    dec corazony
    mDrawFila corazonx,corazony,39t,3t
    ;fila de arriba
    dec corazonx
    mDecVar corazony,4t
    mDrawFila corazonx,corazony,39t,5t
    ;fila de arriba
    dec corazonx
    mDecVar corazony,6t
    mDrawFila corazonx,corazony,39t,7t
    ;fila de arriba
    dec corazonx
    mDecVar corazony,7t
    mDrawFila corazonx,corazony,39t,7t
    ;fila de arriba
    dec corazonx
    mDecVar corazony,6t
    mDrawFila corazonx,corazony,39t,2t
    inc corazony
    mDrawFila corazonx,corazony,39t,2t
    mov corazonx,ax
    mov corazony,dx
    pop dx
    pop ax
    ret
pDrawCorazon endp
```

pDrawEnemigo1: Procedimiento para dibujar el enemigo tipo 1.

```
pDrawEnemigo1 proc
    ;punta sur del enemigo
    push ax
    push dx
    mov ax, ce_x
    mov dx, ce_y
    mDecVar ce_y,4
    mDrawPixel ce_x,ce_y,01
    mIncVar ce_y,7
    mDrawPixel ce_x,ce_y,01
    ;fila anterior
    dec ce_x
    mDecVar ce_y,7
    mDrawPixel ce_x,ce_y,01
    inc ce_y
    inc ce_y
    mDrawPixel ce_x,ce_y,01
    inc ce_y
    inc ce_y
    inc ce_y
    mDrawPixel ce_x,ce_y,01
    inc ce_y
    inc ce_y
    mDrawPixel ce_x,ce_y,01
    ;fila anterior
    dec ce_x
    mDecVar ce_y,6
    mDrawFila ce_x,ce_y,01,2t
    inc ce_y
    inc ce_y
    mDrawFila ce_x,ce_y,01,2t
    ;fila anterior
    dec ce_x
    mDecVar ce_y,7
    mDrawFila ce_x,ce_y,01,8t
    ;fila anterior
    dec ce_x
    mDecVar ce_y,7
    mDrawPixel ce_x,ce_y,01
    inc ce_y
    inc ce_y
    mDrawFila ce_x,ce_y,01,2t
```

```
    inc ce_y
    mDrawPixel ce_x,ce_y,01
    ;fila anterior
    dec ce_x
    mDecVar ce_y,4t
    mDrawFila ce_x,ce_y,01,4t
    ;antenas
    dec ce_x
    mDecVar ce_y,5
    mDrawPixel ce_x,ce_y,01
    mIncVar ce_y,5t
    mDrawPixel ce_x,ce_y,01
    mov ce_x,ax
    mov ce_y,dx
    pop dx
    pop ax
    ret
pDrawEnemigo1 endp
```

pDrawEnemigo2: Procedimiento para dibujar el enemigo tipo 2.

```
pDrawEnemigo2 proc
    push ax
    push dx
    ;mov ce_x,40t
    ;mov ce_y,140t
    mov ax, ce_x
    mov dx, ce_y
    ;parte sur del enemigo
    mDecVar ce_y,4t
    mDrawPixel ce_x,ce_y,2t
    inc ce_y
    inc ce_y
    mDrawPixel ce_x,ce_y,2t
    inc ce_y
    inc ce_y
    inc ce_y
    mDrawPixel ce_x,ce_y,2t
    inc ce_y
    inc ce_y
    mDrawPixel ce_x,ce_y,2t
    ;fila anterior
    dec ce_x
```

```
mDecVar ce_y, 6t
mDrawFila ce_x,ce_y,2t,2
inc ce_y
inc ce_y
mDrawFila ce_x,ce_y,2t,2
;fila anterior
dec ce_x
mDecVar ce_y, 7t
mDrawFila ce_x,ce_y,2t,8t
;fila de los ojos
dec ce_x
mDecVar ce_y, 8t
mDrawPixel ce_x,ce_y,2t
mIncVar ce_y,3
mDrawFila ce_x,ce_y,2t,2t
mIncVar ce_y,2
mDrawPixel ce_x,ce_y,2t
;fila anterior
dec ce_x
mDecVar ce_y, 6t
mDrawFila ce_x,ce_y,2t,6t
;Antenas
dec ce_x
mDecVar ce_y, 5t
mDrawPixel ce_x,ce_y,2t
inc ce_y
inc ce_y
inc ce_y
mDrawPixel ce_x,ce_y,2t
;Antenas2
dec ce_x
mDecVar ce_y, 4t
mDrawPixel ce_x,ce_y,2t
mIncVar ce_y,5t
mDrawPixel ce_x,ce_y,2t
;fila anterior
dec ce_x
mDecVar ce_y, 4t
mDrawPixel ce_x,ce_y,2t
inc ce_y
inc ce_y
inc ce_y
mDrawPixel ce_x,ce_y,2t
```

```
    mov ce_x,ax
    mov ce_y,dx
    pop dx
    pop ax

    ret
pDrawEnemigo2 endp
```

pDrawEnemigo3: Procedimiento para dibujar el enemigo tipo 3.

```
pDrawEnemigo3 proc
    push ax
    push dx
    ;mov ce_x,50t
    ;mov ce_y,140t
    mov ax, ce_x
    mov dx, ce_y
    ;punta sur del enemigo
    dec ce_y
    mDrawFila ce_x,ce_y,44t,2t
    ;fila anterior
    dec ce_x
    mDecVar ce_y,3
    mDrawFila ce_x,ce_y,44t,4t
    ;fila anterior
    dec ce_x
    mDecVar ce_y,5
    mDrawFila ce_x,ce_y,44t,6t
    ;fila anterior
    dec ce_x
    mDecVar ce_y,7
    mDrawFila ce_x,ce_y,44t,2t
    inc ce_y
    mDrawFila ce_x,ce_y,44t,2t
    inc ce_y
    mDrawFila ce_x,ce_y,44t,2t
    ;fila anterior
    dec ce_x
    mDecVar ce_y,8
    mDrawPixel ce_x,ce_y,44t
    inc ce_y
    inc ce_y
    mDrawFila ce_x,ce_y,44t,4t
```

```
    inc ce_y
    mDrawPixel ce_x,ce_y,44t
    ;fila anterior
    dec ce_x
    mDecVar ce_y,4
    mDrawFila ce_x,ce_y,44t,2t
    ;fila anterior
    dec ce_x
    mDecVar ce_y,4
    mDrawFila ce_x,ce_y,44t,2t
    inc ce_y
    inc ce_y
    mDrawFila ce_x,ce_y,44t,2t
    ;fila anterior
    dec ce_x
    mDecVar ce_y,6
    mDrawPixel ce_x,ce_y,44t
    mIncVar ce_y, 5t
    mDrawPixel ce_x,ce_y,44t
    mov ce_x,ax
    mov ce_y,dx
    pop dx
    pop ax
    ret
pDrawEnemigo3 endp
```

pDrawNave: Procedimiento para dibujar la nave principal.

```
pDrawNave proc
    push cx
    push ax
    push dx
    mov ax,cNave_x
    mov dx, cNave_y
    ;CAÑON PRINCIPAL
    mDrawPixel cNave_x,cNave_y,39t
    inc cNave_x
    mDrawPixel cNave_x,cNave_y,15t
    ;CUERPO
    inc cNave_x
    dec cNave_y
    mDrawFila cNave_x,cNave_y,15t,3t
    inc cNave_x
```

```
mDecVar cNave_y,4t
mDrawFila cNave_x,cNave_y,15t,5t
inc cNave_x
mDecVar cNave_y,6t
mDrawFila cNave_x,cNave_y,15t,7t
;VENTANAS PRINCIPALES NAVE
inc cNave_x
mDecVar cNave_y,7t
mDrawFila cNave_x,cNave_y,15t,2t
mDrawFila cNave_x,cNave_y,39t,3t
mDrawFila cNave_x,cNave_y,15t,2t
;siguiente parte y parte de las ventanas
inc cNave_x
mDecVar cNave_y,9t
mDrawFila cNave_x,cNave_y,15t,4t
mDrawPixel cNave_x,cNave_y,39t
inc cNave_y
mDrawPixel cNave_x,cNave_y,15t
inc cNave_y
mDrawPixel cNave_x,cNave_y,39t
inc cNave_y
mDrawFila cNave_x,cNave_y,15t,4t
;otra parte
inc cNave_x
mDecVar cNave_y,12t
mDrawFila cNave_x,cNave_y,15t,13t

;otra parte
inc cNave_x
mDecVar cNave_y,13t
mDrawFila cNave_x,cNave_y,15t,2t
inc cNave_y
mDrawFila cNave_x,cNave_y,15t,7t
inc cNave_y
mDrawFila cNave_x,cNave_y,15t,2t

;OTRA PARTE
inc cNave_x
mDecVar cNave_y,14t
mDrawFila cNave_x,cNave_y,15t,2t
inc cNave_y
inc cNave_y
mDrawFila cNave_x,cNave_y,39t,2t
```

```
    mDrawFila cNave_x,cNave_y,15t,3t
    mDrawFila cNave_x,cNave_y,39t,2t
    inc cNave_y
    inc cNave_y
    mDrawFila cNave_x,cNave_y,15t,2t
    ;11 filas
    inc cNave_x
    mDecVar cNave_y,11t
    mDrawFila cNave_x,cNave_y,39t,2t
    inc cNave_y
    inc cNave_y
    inc cNave_y
    mDrawFila cNave_x,cNave_y,39t,2t
    mov cNave_x,ax
    mov cNave_y,dx
    cmp nivelGame,1
    je salir
    ;caño izquierdo
        mIncVar cNave_x,5t
        mDecVar cNave_y,7t
        mDrawPixel cNave_x,cNave_y,39t
        mov cx,3t
        canonizq:
            inc cNave_x
            mDrawPixel cNave_x,cNave_y,15t
        loop canonizq
        mov cNave_x,ax
        mov cNave_y,dx
        cmp nivelGame,2
        je salir
    ;cañon derecho
        mIncVar cNave_x,5t
        mIncVar cNave_y,7t
        mDrawPixel cNave_x,cNave_y,39t
        mov cx,3t
        canonder:
            inc cNave_x
            mDrawPixel cNave_x,cNave_y,15t
        loop canonder
        mov cNave_x,ax
        mov cNave_y,dx

    salir:
```

```
    pop dx
    pop ax
    pop cx
    ret
pDrawNave endp
```

pMovBala3: Procedimiento para mover la bala tipo 3, las demás procedimientos para las balas tipo 1 y 2 parten de esta restando funcionalidades.

```
pMovbala3 proc
    push ax
    push dx
    push cx
    mov cx,3t
    movnormal:
        cmp bala3x,5t ;si llega al tope de la pantalla se detiene
        je  finmovimiento
        push cx
        ;OBTENER EL VALOR DE UN PIXEL
        dec bala3x
        mov cx,bala3y ;column
        dec cx
        mov dx,bala3x ;fila
        dec dx  ;dec dx no tiene que ver con el procedimiento para
dibujar la bala
        dec dx  ;si embargo se decrementa dos veces para saber como es
el pixel  a la bala dos posiciones arriba
        mov ah, 0Dh
        int 10h
        cmp al,1t; si es igual al enemigo tipo 1 lo destruye y la bala
sigue el recorrido con el daño de esta restada en 1
        je DestEnemigot1
        cmp al,2t ;si es igual al enemigo tipo 2 lo destruye y y la
bala sigue el recorrido con el daño de esta restada en 2
        je DestEnemigot2
        cmp al,44t ;si es igual al enemigo tipo 3 lo destruye y la bala
desaparece
        je DestEnemigot3
        call pDrawBala3 ;pinta la bala
        mIncVar bala3x,3t ;para borrar el rastro de la bala se
posiciona en el ultimo pixel pintado de esta
```

```
        mDrawPixel bala3x,bala3y,0t ;se pinta de negro
        inc dx
        inc dx
        mov bala3x,dx
        pop cx
    loop movnormal
    jmp salir
    DestEnemigot1:
        pop cx
        mDrawNaveEdestruida bala3x,bala3y
        mSumarDw scoreG, 100t
        cmp damageb3,1 ; si el daño de la bala es de 1 (desaparece la
bala)
        je finmovimiento
            dec damageb3 ;si es de 2 o mas se le resta 1 al daño de la
bala y sigue su camino
            mLimpiarDisparo bala3x,bala3y ;borrar bala
            jmp salir ;no desaparece la bala
    DestEnemigot2:
        pop cx
        cmp damageb3,1 ; si el daño de la bala es menor a 1 ya no tiene
el daño necesario para destruir al enemigo 2
        je finmovimiento ;se borra la nave
        mDrawNaveEdestruida bala3x,bala3y
        mSumarDw scoreG, 200t
        cmp damageb3,2 ; si el daño de la bala es de 2 (desaparece la
bala)
        je finmovimiento
            dec damageb3
            dec damageb3 ;si es de 3 se le resta 2 al daño de la bala y
sigue su camino
            mLimpiarDisparo bala3x,bala3y ;borrar bala
            jmp salir ;no desaparece la bala 2
    DestEnemigot3:
        pop cx
        cmp damageb3,2 ; si el daño de la bala es menor o igual que 2
ya no tiene el daño necesario para destruir al enemigo 3
        jle finmovimiento ;se borra la bala
        mDrawNaveEdestruida bala3x,bala3y ;si tiene los 3 justos si
destruye la nave y suma al score
        mSumarDw scoreG, 500t
        jmp finmovimiento
    finmovimiento:
```

```
        mov damageb3,3t
        mLimpiarDisparo  bala3x,bala3y ;borrarbala
        mov estD3,0 ;estado disparo 3
    salir:
    pop cx
    pop dx
    pop ax
    ret
pMovbala3 endp
```

pMovEnemys: Variable que permite mover los enemigos de uno en uno para realizar el ataque kamikaze a la nave principal.

```
pMovEnemys proc
    push cx
    cmp estEnem,3
    je filaene3
    cmp estEnem,2
    je filaene2
    cmp estEnem,1
    je filaene1
    jmp salir ;SE MUEVE EL ESTADO PARA PASAR AL NIVEL 2
    filaene3:
        mov cx,nivelGame
        movi3:
            call pDestEnemA ;el enemigo fue destruido con anterioridad?
            cmp DestEnemA, 1 ;si entonces saltar a fin de movimiento
            je finMov3
            movVariablesDw borrXenemy, ce_x
            movVariablesDw borrYenemy, ce_y
            mDrawEborrado borrXenemy,borrYenemy
            call pColision
            cmp colisionE,1; si colisiono con la nave principal
            je finMov3
            cmp ce_x,196t ;si llego al margen inferior de la pantalla
            je finMov3
            inc ce_x
            call pDrawEnemigo3
        dec cx
        jne movi3
        jmp salir
        finMov3:
            call pDrawEborradoU
```

```
            movVariablesDw ce_x,filaIgame ;fila actual
            mSumarDw ce_y,28t
            cmp ce_y,336t ;comparar con la ultima posicion que puede
tener una nave enemgia
            jb  salir ;si es menor al margen salir y seguir graficando
de forma normal
            mRestaDw ce_x,15t
            movVariablesDw filaIgame, ce_x ; se actualiza la fila
actual a usar
            mov ce_y,308t
            mov estEnem,2
    filaene2:
        mov cx,nivelGame

        movi2:
            call pDestEnemA ;el enemigo fue destruido con anterioridad?
            cmp DestEnemA, 1 ;si entonces saltar a fin de movimiento
            je finMov2
            movVariablesDw borrXenemy, ce_x ; con las filas
actualizadas
            movVariablesDw borrYenemy, ce_y ;con la columna actualizada
            mDrawEborrado borrXenemy,borrYenemy
            call pColision
            cmp colisionE, 1 ; si la nave enemiga choco con la nave
principal
            je finMov2
            cmp ce_x,196t
            je finMov2
            inc ce_x
            call pDrawEnemigo2
        dec cx
        jne movi2
        jmp salir
        finMov2:
            call pDrawEborradoU
            movVariablesDw ce_x,filaIgame ;se vuelve a reestablecer x
en la fila actual
            mRestaDw ce_y,28t ;se resta 28 a la columna actual
            cmp ce_y,140t ; si es menor a 140t es que ya se movieron
los 7 enemigos
            jae salir ;si es mayor o igual al margen salir y seguir
graficando de forma normal
```

```
            mRestaDw ce_x,15t ;si es menor entonces pasar a la
siguiente fila de enemigos
            movVariablesDw filaIgame,ce_x ; se actualiza la fila actual
            mov ce_y,140t
            mov estEnem,1
    filaene1:
        mov cx,nivelGame

        movi1:
             call pDestEnemA ;el enemigo fue destruido con
anterioridad?
            cmp DestEnemA, 1 ;si entonces saltar a fin de movimiento
            je finMov
            movVariablesDw borrXenemy, ce_x ;toma la fila del enemigo
            movVariablesDw borrYenemy, ce_y ;toma la fila del enemigo
            mDrawEborrado borrXenemy,borrYenemy ;pinta un cuadro negro
en dicha poisicon
            call pColision ;verifica si el enemigo no choco con la nave
principal
            cmp colisionE,1 ; si la nave choco significa el fin del
movimiento de dicha nave
            je finMov
            cmp ce_x,196t ;llego al margen inferior de la pantalla
            je finMov; si llego al final de la pantalla, es su fin de
movimiento
            inc ce_x ;se incrementa su fila de 1 en 1
            call pDrawEnemigo1 ;se manda a dibujar el enemigo 1
        dec cx
        jne movi1
        jmp salir
        finMov:
            call pDrawEborradoU
            movVariablesDw ce_x,filaIgame
            mSumarDw ce_y,28t
            cmp ce_y,336t ; si es mayor a 336t es que ya se movieron
los 7 enemigos
            jb salir ;si es menor al margen salir y seguir graficando
de forma normal
            mRestaDw ce_x,15t
            movVariablesDw filaIgame,ce_x
            cmp ce_x, 0
            jne SeguirMoviendo
    FinalizarMovEnemigos:
```

```
        mov estEnem,3 ;incrementa en uno el nivel
        inc nivelGame
        mov printEnemyE,0 ;para indicarle al programa que debe de
volver a imprimir enemigos
        jmp salir
    SeguirMoviendo:
        movVariablesDw ce_x,filaIgame
        mov ce_y,140t
        mov estEnem,3
    salir:
    pop cx
    ret
pMovEnemys endp
```

pTimeGame: Cronómetro del juego.

```
pTimeGame proc
    inc cengameN
    cmp cengameN,100t ;cuando llegue a 100 el contador de centisegundos
volvera a 0 y se le sumara 1 a los segundos caso contrario solo sumara
uno y se saldra
    jne salir

    mov cengameN,0 ;centisegundos vuelve a 0
    inc seggameN ;se aumenta a uno los segundos
    inc segGameReporte; se aumenta en uno los segundos para el reporte
    cmp seggameN,60t;cuando llegue a 60 volvera a 0 los segundos y se
le sumara uno a los minutos
    jne salir

    mov seggameN,0t
    inc mingameN
    salir:
        mLimpiar cengameS,2,0
        mLimpiar seggameS,2,0
        mLimpiar mingameS,2,0
        Num2String cengameN,cengameS
        Num2String seggameN,seggameS
        Num2String mingameN,mingameS
        mImprimirLetreros mingameS,12t,4t,15t
        mImprimirLetreros dospuntosg,12t,6t,15t
        mImprimirLetreros seggameS,12t,7t,15t
        mImprimirLetreros dospuntosg,12t,9t,15t
```

```
        mImprimirLetreros cengameS,12t,10t,15t
    ret
pTimeGame endp
```

pPauseGame: Procedimiento para generar una pausa en medio del juego.

```
pPauseGame proc
    call pGuardarMatrizVideo ; guardar el estado de la matriz de video
para posteriormente cargarla sin los letreros
    mov exitGame, 0
    mImprimirLetreros letPause,5t,25t,15t
    mImprimirLetreros letRen,12t,20t,15t
    mImprimirLetreros letExit,15t,20t,15t
    ciclo:
        mov ah, 00  ;Espera a que se presione una tecla y la lee
        int 16h
        cmp al, 27t ;escape
        je exitG
        cmp al, " " ;espacio
        je salir
        jmp ciclo  ;estara en un ciclo si no es o espacio o escape
    exitG:
        mov exitGame,1
    salir:
        call pCargarMatrizVideo ;cargar la matriz de video guardada
luego de los letreros
    ret
pPauseGame endp
```

pGuardarMatrizVideo: procedimiento para guardar el estado actual de la matriz de video en un archivo externo.

```
pGuardarMatrizVideo proc
    push cx
    push ax
    push si
    mCrearFile matrizgraph
    mOpenFile2Write matrizgraph
    mov si,0
    mov cx, 64000t
    copiarmatriz:
```

```
        mov bl, es:[si]
        mov eleactualG,bl
        mWriteToFile eleactualG
        inc si
    loop copiarmatriz
    call pCloseFile
    pop si
    pop bx
    pop cx
    ret
pGuardarMatrizVideo endp
```

pCargarMatrizVideo: Cargar el estado guardado de la matriz de video en un archivo externo para volver a este..

```
pCargarMatrizVideo proc
    push cx
    push ax
    push si
    mOpenFile2Write matrizgraph
    mov si,0
    mov cx, 64000t
    copiarmatriz2:
        mReadFile eleactualG
        mov bl,eleactualG
        mov  es:[si],bl
        inc si
    loop copiarmatriz2
    call pCloseFile
    pop si
    pop bx
    pop cx
    ret
pCargarMatrizVideo endp
```

ORDENAMIENTOS:

pMoveOrdenamiento: Corazón de los ordenamientos, permite de modo default mover los gráficos a 100 fps con una delay escogido bajo.

```
pMoveOrdenamiento proc
    push ax
```

```
    push dx
    mov auxfpsT,0
    reset:
        call pConfigInicOrd
        call pDrawBarras
        call pOrdMando
    fps: ;ciclo que provoca un movimiento cada centisegundo
        mov ah,2Ch
        int 21
        cmp dl, auxfpsT
        je fps
    mov auxfpsT, dl
    call pTimeOrd
    cmp EstOrd,0t
    je sinAccion
        mDrawBarra 17t,0t,170t,8t,0t;borrar flechas de pasos anteriores
        call pBubbleSort
        cmp EstOrd,0t
        je exit
    sinAccion:
    jmp fps
    exit:
        mImprimirLetreros msgPressEnd,24t,7t,15t
        ciclo: ;SALE HASTA QUE SE PRESIONE "FIN"
        mov ah, 00  ;Espera a que se presione una tecla y la lee
        int 16h
        cmp al,"O"
        je exit2
        cmp al,"1"
        je exit2
        jmp ciclo
        exit2:
        pop dx
        pop ax
    ret
pMoveOrdenamiento endp
```

pRDatosOrdPuntos: Permite guardar los 20 primeros datos del archivo score en un array asi como la posición de estos.

```
pRDatosOrdPuntos proc
    push si
    call pLimpiarArraySort
```

```
    mov auxDw,0
    mov CDatos,0
    mov NumactualDoc,0
    ;NAMEUSER -01- NIVEL -01- PUNTOS -01- TIEMPO ENTER ESPACIO
    mOpenFile2Write scoresb
    call pInidoc
    mov si, 0
    ciclo:
    mReadFile eleactual
    cmp eleactual," "
    je salir
    mHallarSimbolo 01
    mHallarSimbolo 01
    cmp punOtiempo,1 ;SE ESCOGIO LA METRICA DEL TIEMPO? SI ES ASI
MOVERSE UNA SEPARACION MAS
    jne notiempo;SI NO ES ASI NO MOVERSE MAS DE LA POSICION ACTUAL
        mHallarSimbolo 01
    notiempo:
    mReadFile eleactual
    mLimpiar NumactualDocS,6t,"$"
    mCapturarStringDoc NumactualDocS ;captura el numero en esta
variable
    String2Num NumactualDocS,NumactualDoc,"$"; convierte el numero
string a numero decimal
    MovVariablesDw datosOrd[si], NumactualDoc ;mmueve el numeor a esta
posicion de arreglo
    movVariablesDw indexDato[si],auxDw ; index
    inc auxDw
    mHallarSimbolo 0A
    inc si
    inc si
    inc CDatos
    cmp CDatos,20t ;si son mas de 20 termina de recopilar
    je salir
    jmp ciclo
    salir:
    call pCloseFile
    pop si
    ret
pRDatosOrdPuntos endp
```

pLimpiarAraySort: Limpia los array a usar para los ordenamientos.

```
pLimpiarArraySort proc
    push cx
    push bx
    mov cx,20t
    mov bx,0
    ciclo:
        mov [indexDato+bx],0
        mov [datosOrd+bx],0
        add bx,2
    loop ciclo
    pop bx
    pop cx
    ret
pLimpiarArraySort endp
```

pDrawBarras: dibuja el estado actual de los array con los datos cargados en un diagrama de barras.

```
pDrawBarras proc
    push cx
    push si
    push ax
    cmp CDatos,0
    je salir
    mov cx, CDatos ;se repetira el ciclo la n cantidad de datos tomados
    mov si,0  ; se inicia si
    mov brEspOy, 0; para el borrado
    mov x_barra , 16t ;empezara en el pixel 16t y fila 3 de un string
    mov y_barra , 50t ;empezara a graficar cada barra desde aqui
    mov altoBarra, 140t  ; 140t es el espacio de filas de pixeles
libres  para graficar sin contar los rotulos
    mDivisionDw altoBarra,CDatos  ;se divide entre la cantidad de datos

    cicloBarras:
        mDrawBarra x_barra,brEspOy,altoBarra,318t,0t ;BORRA LOS
MOVIMIENTOS ANTERIORES DE CADA LINEA
        push x_barra;se guarda x
        ;SE DIVIDE LA POSICION ACTUAL ENTRE 8 PARA IMPRIMIR EL STRING
DEL VALOR DE LA BARRA
        mDivisionDw x_barra,8t
        mov ax,x_barra
        mov filaLetreroOrd,al
        mLimpiar DatOrsb,6t,0
```

```
        Num2String datosOrd[si],DatOrsb
        mImprimirLetreros DatOrsb,filaLetreroOrd,1t,15t
        pop x_barra;se recupera el valor inicial de la barra

        movVariablesDw anchoBarra, datosOrd[si] ;se obtiene el ancho de
la barra tomando el valor actual del  array de datos
        ;-------------------------------------PUNTAJE O
TIEMPO--------------------------------
        cmp punOtiempo,0
        je puntaje
        jne tiempo
        puntaje:
        mDivisionDw anchoBarra,65t  ;se dividira por esto para que la
barra se reduzca un 80 veces su valor en decimal y pueda caber en la
pantalla
        jmp graficarBarra
        tiempo:
        cmp anchoBarra,269t
        jbe NoSobrepasaAncho;<=
        mov anchoBarra,267t
        NoSobrepasaAncho:

;-------------------------------------------------------------------------
---------------
        graficarBarra:
        mDrawBarra x_barra,y_barra,altoBarra,anchoBarra,15t ;se grafica
la barra
        mIncVar x_barra, altoBarra ;se desplaza hacia abajo la barra n
pixeles iguales al tamaño de cada barra
        inc x_barra ;se le suma uno para dejar un espacio vacio, EN
ESTE MOMENTO YA SE ENCUENTRA EN LA FILA ACTUAL INDICADA SE DIVIDE POR 8
        inc si
        inc si
    dec cx
    jne cicloBarras
    salir:
    pop ax
    pop si
    pop cx
    ret
pDrawBarras endp
```

pShowtop10: Permite visualizar el top 10 de mejores scores.

```
pShowtop10 proc
    push ax
    push bx
    push cx
    call pLimpiarConsola
    mMostrarString msgTop10 ;IMPRIMIR TITULO DEL APARTADO
    call pImprimirUser ;IMPRIMIR USUARIO ACTUAL
    mov punOtiempo,0
    ;capturar Datos
    call pRDatosOrdPuntos
    ;burble, ORDENAR LOS DATOS
    call pBubbleSortAuto
    ;print
    mMostrarString msgTitleRep
    mMostrarString sepRepOrden
    mov auxDw,0
    mov bx,0
    mov ax, 0
    mov cx,CDatos
    mOpenFile2Write scoresb
    cicloTop10:
            mMoverAFila [indexDato+bx]
            mLimpiar filaScore,25t,0
            mCapturarFilaDoc filaScore
            ;rank
            mov auxDw,ax
            inc auxDw
            mLimpiar auxString,4t,0
            Num2String auxDw,auxString
            mMostrarString auxString
            mMostrarString filaScore
            mMostrarString msgEnter
        add bx,2
        inc ax
        cmp ax,10t
        je finCiclo10
    dec cx
    jne cicloTop10
    finCiclo10:
    call pCloseFile
    call pEspEnter
    call pLimpiarConsola
```

```
    pop cx
    pop bx
    pop ax
    ret
pShowtop10 endp
```

pShowMyTop10: Permite visualizar los 10 mejores scores de un jugador.

```
pShowMytop10 proc
    push ax
    push bx
    push cx
    push dx
    call pLimpiarConsola
    mMostrarString msgMyTop10 ;IMPRIMIR TITULO DEL APARTADO
    call pImprimirUser ;IMPRIMIR USUARIO ACTUAL
    mov punOtiempo,0
    ;capturar Datos
    call pRDatosOrdPuntos
    ;burble, ORDENAR LOS DATOS
    call pBubbleSortAuto
    ;print
    mMostrarString msgTitleRep
    mMostrarString sepRepOrden
    mov auxDw,0
    mov bx,0
    mov ax, 0
    mov cx,CDatos
    mOpenFile2Write scoresb
    cicloTop10:
            mMoverAFila [indexDato+bx]
            mLimpiar filaScore,25t,0
            mCapturarFilaDoc filaScore
            call pFilaUScore  ;procedimiento que compara la variable
UserNameG con el principio de la fila
            cmp cadIguales,1 ;si son iguales se procede a imprimir la
fila
            jne nofilaUser ; si no son iguales no se imprime nada
            ;EL USER EN LA FILA ES IGUAL, SE PROCEDE A IMPRIMIR LA FILA
                mov auxDw,ax
                inc auxDw
```

```
                mLimpiar auxString,4t,0
                Num2String auxDw,auxString
                mMostrarString auxString
                mMostrarString filaScore
                mMostrarString msgEnter
                inc dx
                cmp dx,10t  ;si ya se imprimio 10 veces un dato, se
deja de imprimir mas escores
                je finCiclo10
        nofilaUser:
        add bx,2
        inc ax
    dec cx
    jne cicloTop10
    finCiclo10:
    call pCloseFile
    call pEspEnter
    call pLimpiarConsola
    pop dx
    pop cx
    pop bx
    pop ax
    ret
pShowMytop10 endp
```

pBubbleSort: Ordena y ayuda a graficar un diagrama de barras mientras se está realizando dicho ordenamiento.

```
pBubbleSort proc
    push ax
    push dx
    push cx
    push bx
    cmp CDatos,1
    jbe StopOrden
    mCompararDw nRepeticiones,CDatos
    je StopOrden
        mCompararDw nRepeticiones2,CDatos
        je StopCiclo
            mov bx,indexCiclo
            mov ax, [datosOrd+bx]
            cmp ascDec,0
            je ascendenteG
```

```
            jmp descendenteG
            ascendenteG:
                cmp ax, [datosOrd+bx+2]
                jae noswap ;si el dato 1 es mas grande al dato 2, no se
mueve y se queda de primero
                jmp swap
            descendenteG:
                cmp ax, [datosOrd+bx+2]
                jbe noswap ;si el dato 1 es mas grande al dato 2, no se
mueve y se queda de primero
            swap:
            ;swap
            mov dx,[datosOrd+bx+2]
            mov [datosOrd+bx+2],ax
            mov [datosOrd+bx],dx
            mDelaytCenti velocity;VELOCIDAD DEL DELAY
            call pInterCambioB
            ;MOVER INDEX
            call pMoverIndex
            noswap:
            mDelaytCenti velocity;VELOCIDAD DEL DELAY
            call pDrawBarraBubble
                mIncVar x_barra, altoBarra ;se desplaza hacia abajo la
barra n pixeles iguales al tamaño de cada barra
                inc x_barra ;se le suma uno para dejar un espacio
vacio, EN ESTE MOMENTO YA SE ENCUENTRA EN LA FILA ACTUAL INDICADA SE
DIVIDE POR 8
                inc indexCiclo
                inc indexCiclo
            call pDrawBarraBubble
        inc nRepeticiones2
    jmp salir
    StopCiclo:
        call pResetVarOrd
        inc nRepeticiones
        jmp salir
    StopOrden:
        call pDrawBarras
        mov EstOrd,0
    salir:
    pop bx
    pop cx
    pop dx
```

```
    pop ax
    ret
pBubbleSort endp
```

pBubbleSortAuto: Método de Ordenamiento Bubble, el cual ordena más no ayuda a gráficar como el mostrado anteriormente.

```
pBubbleSortAuto proc
    push ax
    push dx
    push cx
    push bx
    cmp CDatos,1t
    je salir
    mov cx,CDatos
    nRepeat:
        push cx
        mov cx, CDatos
        dec cx
        mov bx,0
        compEvery: ;comparar dato con cada dato del arreglo
            mov ax, [datosOrd+bx]
            cmp ax, [datosOrd+bx+2]
            jae noswap ;si el dato 1 es mas grande al dato 2, no se
mueve y se queda de primerz
            ;swap
            mov dx,[datosOrd+bx+2]
            mov [datosOrd+bx+2],ax
            mov [datosOrd+bx],dx
            ;MOVER INDEX
            call pMoverIndex
            noswap:
            add bx,2
        dec cx
        jne compEvery
        pop cx
    loop nRepeat
    salir:
    pop bx
    pop cx
```

```
    pop dx
    pop ax
    ret
pBubbleSortAuto endp
```

pFechaTime: Obtiene la fecha y hora actual y los almacena en variables como strings.

```
pFechaTime proc
    push bx
    push cx
    xor bx,bx
    ;dia,mes,anio,hora,min,segun
    mov ah,2Ah
    int 21h
    mov year,cx  ;valor numerico de año
    mov bl, dh   ;valor numerico de mes
    mov month,bx
    mov bl, dl   ;valor numerico de dia
    mov day,bx

    mov ah,2Ch
    int 21h
    mov bl, ch  ;valor numerico de horas
    mov hours,bx
    mov bl, cl   ;valor numerico de minutos
    mov minutes,bx
    mov bl, dh   ;valor numerico de segundos
    mov seconds,bx

    ;CONVERSION DE NUMEROS A VARIABLES
    Num2String year, anio
    Num2String month, mes
    Num2String day, dia
    Num2String hours, hora
    Num2String minutes,min
    Num2String seconds,segun
    pop cx
    pop bx
    ret
pFechaTime endp
```

Equipo Requerido:

- 2 gb ram
- procesador de 2 nucleos 2.3 ghz
- Dosbox 0.74-3
- Visual Code

CONCLUSIONES

- A la hora de manipular archivos externos es necesario limpiar los registros utilizados, si en algún momento llegamos a realizar una operación como la división luego de abrir dichos archivos con el programa.
- La mala utilización de registros puede dar errores como el paro del funcionamiento de un programa.
- Los lenguajes de alto nivel facilitan mucho todo lo concerniente a la programación permitiendo que nos despreocupemos del uso correcto de los registros.
- Las macros son una funcionalidad que permiten en gran medida el ahorro de líneas de código siendo estas reutilizables en otros proyectos.
- No se pueden mover dos variables al mismo tiempo, debe de existir un registro intercesor entre los dos.
- La memoria de gráficos permite manipular de forma mas sencilla los pixeles a colores para que el programa realice distintas reacciones dependiendo de estos, además de permitirse guardar esta en archivos externos.