CESAR LEONEL CHAMALE SICAN 201700634

UNIVERSIDAD SAN CARLOS DE GUATEMALA

FACULTA DE INGENIERIA

INGENIERIA EN CIENCIAS Y SISTEMAS

MANUAL TECNICO:

En este manual encontramos cada macro, procedure, y código en general utilizada en el programa, a un lado de las líneas encontrara el pseudocodigo o explicación de cada macro, procedure o proceso en general.

```
include macros2.asm ;archivo con los macros a utilizar
include archivos.asm ; incluimos macros para manipulacion de archivos

imprimir macro buffer ;imprime cadena
push ax
push dx

    mov ax, @data
    mov ds,ax
    mov ah,09h ;Numero de funcion para imprimir buffer en pantalla
    mov dx,offset buffer ;equivalente a que lea dx,buffer, inicializa en dx la po
sicion donde comienza la cadena
    int 21h

pop dx
pop ax
endm


escribirChar macro char
        mov ah, 02h
        mov dl, char
        int 21h
endm

posicionarCursor macro x,y
    mov ah,02h
    mov dh,x
    mov dl,y
    mov bh,0
    int 10h
endm
```

```
imprimirVideo macro caracter, color
    mov ah, 09h
    mov al, caracter ;al guarda el valor que vamos a escribir
    mov bh, 0
    mov bl, color ; valor binario rojo
    mov cx,1 ; este nos sirve para decirle que solo se imprima una vez la letra
    int 10h
endm

esDiptongo macro caracter1, caracter2 ;en el registor al va a tener un 1 si es un
 diptongo o un 0 si no lo es
    LOCAL salida, esA, esE, esO, esDip, regla1Dip, regla2Dip,esIr2, esDipR2
    mov al,0

    regla1Dip:
    esA:
        cmp caracter1,97 ;97 es a
        jne esE

        cmp caracter2,105; es i
        je esDip

        cmp caracter2, 117 ; es u
        je esDip

        jmp regla2Dip


    esE:

        cmp caracter1,101 ; es e
        jne esO

        cmp caracter2,105; es i
        je esDip

        cmp caracter2, 117; es u
        je esDip

        jmp regla2Dip


    esO:

        cmp caracter1,111 ; es o
```

```
        jne regla2Dip

        cmp caracter2,105 ; es i
        je esDip

        cmp caracter2, 117; es u
        je esDip

        jmp regla2Dip

    esDip:
        mov al,1
        inc esDipR1Contador
        inc esDipContadorGen
        jmp salida



;----------- TEGLA 2 DE DIPTONGOS
    regla2Dip:
        cmp caracter1, 105 ; es i
        JE esIr2
        cmp caracter1, 117; es u
        JE esIr2
        ;jmp regla3Dip
        jmp regla3Dip

        esIr2:
            cmp caracter2,97 ; es a
            JE esDipR2
            cmp caracter2,101 ; es e
            JE esDipR2
            cmp caracter2,111 ; es o
            JNE regla3Dip


        esDipR2:
            mov al,1
            inc esDipR2Contador
            inc esDipContadorGen

;----------------- REGLA 3 DE DIPTONGOS
    regla3Dip:
        ;vocales cerradas
        cmp caracter1, 105 ; es i
```

```
        JE esIr3
        cmp caracter1, 117 ; es u
        JE esUr3
        jmp salida

        esIr3:
            cmp caracter2, 117 ; es u
            JE esDipR3
        esUr3:
            cmp caracter2, 105 ; es i
            JNE salida

        esDipR3:
            mov al,1
            inc esDipContadorGen
            inc esDipR3contador


    salida:

endm

esTriptongo macro caracter1, caracter2, caracter3
    LOCAL salida, esAt, esEt, esOt,esTrip
    mov bl,0

    cmp caracter1,105; es i
    JE esIt
    cmp caracter1,117 ; es u
    JE esIt
    jmp salida
    esIt:
        cmp caracter2, 97 ;97 es a
        JE esAt
        cmp caracter2, 101 ; es e
        JE esEt
        cmp caracter2,111 ; es o
        JE esOt
        jmp salida


    esAt:
        cmp caracter3,105; es i
        JE esTrip
        cmp caracter3,117 ; es u
```

```
        JE esTrip
        jmp salida


    esEt:
        cmp caracter3,105; es i
        JE esTrip
        cmp caracter3,117 ; es u
        JE esTrip
        jmp salida

    esOt:

        cmp caracter3,105; es i
        JE esTrip
        cmp caracter3,117 ; es u
        JE esTrip
        jmp salida

    esTrip:
        mov bl,1
        inc esTripContador
        jmp salida




    salida:
endm


;   ALGORITMO PARA HIATO SIN TIILDES (PSEUDOCODIGO)
;   IF (CARACTER1= a OR e OR O)
;       IF (CARACTER2= a OR e OR O)
;           ES HIATO
;       PASAR A REGLA 2
;   PASAR A REGLA 2
;
; #regla2
;   IF (CARACTER1 = i)
;       IF (CARACTER2 = i)
;           es hiato
;       no es hiato
;   regla 3
;
```

```
;   IF (CARACTER1 = u)
;       IF (CARACTER2 = u)
;           es hiato
;       no es hiato
;   no es hiato
esHiato macro caracter1, caracter2
    LOCAL salida, esHia,hiatoRegla1,esHiatoAbierta,hiatoRegla2,esHiatoI,esHiatoU
    mov cl,0

    hiatoRegla1:
        cmp caracter1,97 ;97 es a
        JE esHiatoAbierta
        cmp caracter1, 101 ; es e
        JE esHiatoAbierta
        cmp caracter1,111 ; es o
        JE esHiatoAbierta
        jmp hiatoRegla2
    esHiatoAbierta:
        cmp caracter2,97 ;97 es a
        JE esHia
        cmp caracter2, 101 ; es e
        JE esHia
        cmp caracter2,111 ; es o
        JE esHia
        jmp hiatoRegla2

    hiatoRegla2:
        cmp caracter1,105; es i
        JE esHiatoI
        cmp caracter1,117 ; es u
        JE esHiatoU
        jmp salida

    esHiatoI:
        cmp caracter2,105; es i
        JE esHia
        jmp salida

    esHiatoU:
        cmp caracter2,117 ; es u
        JE esHia
        jmp salida

    esHia:
        mov cl,1
```

```
        inc esHiatoContGen



    salida:
endm


;-------------------------------------------------------------------------------
--------------

; CONTADORE DE PALABRAS
contadorPalabras macro

    local contarcaracteres,charcontados,noesespacio
    xor si,si          ;limpiar stack index
    mov pcontador,1     ;empezar a contar en 1
    contarcaracteres:
    cmp bufferInformacion[si],"$" ;fin de cadena
    je charcontados    ;si es igual sale del ciclo
    cmp bufferInformacion[si],32 ;compara caracter espacio
    jne noesespacio    ;si no es igual se va a esa etiqueta
    add pcontador,1     ;si es igual aumenta el contador
    noesespacio:
        inc si         ;aumenta el stack index para el siguiente caracter
        jmp contarcaracteres ;regresa al inicio del ciclo
    charcontados:      ;termino de contar
    ;print salto
    ;IntToString pcontador, avisoContaPrint
    ;print pcontador2
    ;imprimir_reg bx

    ;print salto
    xor si,si    ;limpia si por si se usa despues. No es necesario

endm
;-------------------------------------------------------------------------------
---------
salirMenu macro
    print msjcontinue
    getChar
    print salto
    cmp al, 120
    je exit
    jmp MenuOpciones
```

```
endm

;--------------------------------------------------------------------------------
-----------------
.model small

;----------------SEGMENTO DE PILA-----------------------------------------------
----------------
.stack

;----------------SEGMENTO DE DATO-----------------------------------------------
------------------

.data
msjEntrada db 0ah, 0dh, '| Universidad de San Carlos de Guatemala',0ah, 0dh,'| Ar
quitectura de Ensambladores y Computadores 1' , 0ah, 0dh, '| CESAR LEONEL CHAMALE
 SICAN      201700634' , 0ah, 0dh, '| Pr',160,'ctica 4',0ah, 0dh, '| Ingrese x si
 desea cerrar el programa' , '$'
msjComandos db 0ah, 0dh, '| COMANDOS DE APLICACION: ',0ah, 0dh,'| 1. -
abrir_',34,'ruta',34 , 0ah, 0dh, '| 2. -
contar_<diptongo | triptongo | hiato | palabra>' , 0ah, 0dh, '| 3. -
prop_<diptongo | triptongo | hiato > ',0ah, 0dh, '| 4. -
colorear' ,0ah, 0dh,'| 5. -reporte',0ah, 0dh,'| 6. -
diptongo_palabra',0ah, 0dh,'| 7. -hiato_palabra',0ah, 0dh,'| 8. -
triptongo_palabra', '$'

msjIngreseC db 0ah, 0dh, 'INGRESE EL NUMERO DE SU COMANDO: ' , '$'
msjcontinue db 0ah, 0dh, 'precione CUALQUIER tecla para continuar o x para salir:
 ' , '$'
ingreseruta db 0ah,0dh, 'Ingrese una ruta de archivo' , 0ah,0dh, 'Ejemplo: entrad
a.txt' , '$'
;------------------------------MESAJE PARA COMANDOS ----------------------------
------------------
msjComando1 db 0ah, 0dh, '-abrir_','$'
msjComando2 db 0ah, 0dh, '-contar_<','$'
msjComando3 db 0ah, 0dh, '-prop_<','$'
msjComando4 db 0ah, 0dh, '-colorear','$'
msjComando5 db 0ah, 0dh, '-reporte','$'
msjComando6 db 0ah, 0dh, '-diptongo_','$'
msjComando7 db 0ah, 0dh, '-hiato_','$'
msjComando8 db 0ah, 0dh, '-triptongo_','$'
;--------------------------COMPARADORES A COMANDOS ------------------------------
---------------------
;MENSAJE CONTADOR COMPARADOR
;---de contador
```

```
msjContadorCompDi db 'diptongo>','$'
msjContadorCompTri db 'triptongo>','$'
msjContadorCompHi db 'hiato>','$'
msjContadorCompPa db 'palabra>','$'
msjTotalPalabras db 'cantidad todal de palabras: ','$'
msjTotalHiatos db 'cantidad todal de hiatos: ','$'

;---- de prop
;MENSAJE PROP
msjPropDi db 'diptongo>','$'
msjPropTri db 'triptongo>','$'
msjPropHi db 'hiato>','$'

;--------------------------------------------------------------------------------
----------------------
saltoLinea db 0Ah,0Dh,"$"
saludo db 0Ah,0Dh, "---------------------- ANALIZANDO TEXTO  -------------------
","$"
fin db 0Ah,0Dh, "Finalizando el programa.......", "$"
salto db 0ah,0dh, '$' ,'$'

texto db "Hola esto es unn peine y una aula","$"

fila db 0
columna db 0

;-------------------------------------------------SIGNOS DE PUNTUACION -----------
------------------
msjPunto1 db '.','$'
msjPorce db '%','$'
;--------------------------------------------------------------------------------
-----------------
;----------------AGREGADOS PARA COMANDOS ---------------------------------------
--------
comand db 0, '$'
;--------------- PARA ABRIR DOCUMENTO ------------------------------------------
--------
bufferentrada db 50 dup('$')
handlerentrada dw ?
bufferInformacion db 700 dup('$')

msjOpcionesArch db 0ah,0dh, '1. Mostrar informacion' , 0ah,0dh, '2. Cerrar archiv
o' , '$'
;-----------------MENSAJE DE ERROES
err1 db 0ah,0dh, 'Error al abrir el archivo puede que no exista' , '$'
```

```
err2 db 0ah,0dh, 'Error al cerrar el archivo' , '$'
err3 db 0ah,0dh, 'Error al escribir en el archivo' , '$'
err4 db 0ah,0dh, 'Error al crear en el archivo' , '$'
err5 db 0ah,0dh, 'Error al leer en el archivo' , '$'
errorComando db 0ah,0dh, "Error comando no existente" ,0ah,0dh, "$"

;---------------------- PARA CONTADORES ----------------------------------------
-------
pcontador dw 0
avisoContador1 db "aviso contador 1",'$'
;avisoContaPrint dw 0
;avisoContaPrint2 db "aviso print contador palabras ",'$'
contadorHiato dw 1
avisoContador2 db "aviso contador 2",'$'
contadoraux dw 0
avisoContador3 db "aviso contador 3",'$'
pcontadorResH db 15, '$'
;########################
contadorVer db 15, '$'
;########################
contadorDiptongo dw 0
avisoContador4 db "aviso contador 4",'$'
contadorTriptongo dw 0
avisoContador5 db "aviso contador 5",'$'
esDipR2Contador dw 0
avisoContador6 db "aviso contador 6",'$'
esDipR1Contador dw 0
avisoContador7 db "aviso contador 7",'$'
esDipContadorGen dw 0
avisoContador8 db "aviso contador 8",'$'
esDipR3contador dw 0
avisoContador9 db "aviso contador 9",'$'
esTripContador dw 0
avisoContador10 db "aviso contador 10",'$'
esHiatoContGen dw 0
avisoContador11 db "aviso contador 11",'$'
;#####################################
pruebaSimb db 2, '$'
;#####################################
textoaleer db "Texto de muestra con seis palabras siete ocho nueve dies",'$'

;----------------------AUX PARA ENTRADAS----------------------------------------
-------
auxEntrada db 0, '$'
```

```
auxContador db '$'

etiquetaPruebas db "llego xdd" , "$"
;----------------------
- aca hay un buggg########################################
;cuando creamos mas variables para imrprimir aca,, da simbolos raros

auxProp db 0, '$'
auxPalabra db 0, '$'




;----------------SEGMENTO DE CODIGO-----------------------------------------------
--------


.code
XOR_REG proc
    xor ax, ax
    xor bx, bx
    xor cx, cx
    xor dx, dx
    ret
XOR_REG endp
main proc

    mov ax,@data
    mov ds,ax

    Menu:
        print msjEntrada
        print saltoLinea
        getChar ; lee un caracter del teclado y lo guarda en al
        cmp al, 120 ; if (al == 120){va a brincar a la etiqueta salir}else{va a c
ontinuar con el programa}
        je exit
    MenuOpciones:
        print msjComandos
        print saltoLinea
        print msjIngreseC
        print saltoLinea
        getChar
        cmp al,49 ; NUMEOR 1 PARA ABRIR DOC
        je AbrirArchivo
```

```
        cmp al,50 ; NUMERO 2 PARA CONTAR
        je contar
        cmp al,51 ; NUMERO 3 PARA PROP
        JE PROPORCION
        cmp al,52 ; NUMERO 4 PARA COLOREAR
        JE COLOREAR
        jmp exit
    AbrirArchivo: ; PARA ABRIR DOC
        print ingreseruta
        print saltoLinea
        print msjComando1
        limpiar bufferentrada, 50 ,24h ; LIMPIAMOS EL ARRAY PARA LA RUTA, LIMPIAM
OS CON $
        obtenerRuta bufferentrada  ; OBTENEMOS LA RUTA DEL ARCHIVO DE ENTRADA
        abrir bufferentrada,handlerentrada ; LE MANDAMOS LA RUTA Y EL HANDLE QUE
SERA LA REFERENCIA AL FICHERO
        limpiar bufferInformacion, 700 ,24h ; LIMPIAMOS LA VARIABLE DONDE GUARDAM
OS LOS DATOS DEL ARCHIVO DE ENTRADA
        leer handlerentrada, bufferInformacion, 700 ;leemos el archivo
        ; para el tamaño se puede mandar as "SIZEOF bufferInformacion" en anbos d
e limpar pero da problema al ensamblar,
        ;aunque no error, si funciona, pero al automatizar el arranque del ejetuc
able en doss opcions no acepta bien el SIZEOF

        print msjcontinue
        getChar
        cmp al, 120 ; if (al == 120){va a brincar a la etiqueta salir}else{va a c
ontinuar con el programa}
        je exit

    AbrirArchivo2:
        print salto
        print msjOpcionesArch
        print salto
        print msjIngreseC
        getChar
        cmp al,31h
        je MostrarInformacion
        cmp al,32h
        je CerrarArchivo
        jmp AbrirArchivo


    MostrarInformacion:
        print salto
```

```
    print bufferInformacion
    print salto
    print msjcontinue
    getChar
    cmp al, 120
    je exit
    jmp AbrirArchivo2


CerrarArchivo:
    cerrar handlerentrada
    jmp MenuOpciones


Error1:
    print salto
    print err1
    getChar
    jmp Menu

Error2:
    print salto
    print err2
    getChar
    jmp Menu

Error3:
    print salto
    print err3
    getChar
    jmp Menu

Error4:
    print salto
    print err4
    getChar
    jmp Menu

Error5:
    print salto
    print err5
    getChar
    jmp Menu

ErrorNoExist:
```

```
        print errorComando
        jmp Menu

    contar:
        print salto
        print msjContadorCompDi
        print salto
        print msjContadorCompTri
        print salto
        print msjContadorCompHi
        print salto
        print msjContadorCompPa
        print salto
        print msjComando2
        obtenerTexto auxContador
        print auxContador
        ;COMPARACIONES
        ;------------PARA HIATOS
        mov cx,6  ; hiato> 6 posiciones
        mov AX, DS
        mov ES, AX

        LEA si, msjContadorCompHi
        LEA di, auxContador
        repe cmpsb
        JE HIATOc

        ;----------PARA TRIPTONGOS
        xor cx,cx
        mov cx,10  ;triptongo>
        lEA si, msjContadorCompTri
        LEA di, auxContador
        repe cmpsb ;Compare msjContadorCompTri:auxContador
        JE TRIPTONGOc

        ;----------PARA DIPTONGO   diptongo>

        xor cx,cx
        mov cx,9      ; numero de posiciones a comparar diptongo> = 9 posiciones
        LEA si, msjContadorCompDi
        LEA di, auxContador
        repe cmpsb
        JE DIPTONGOc

        ;----------PARA PALABRAS , msjContadorCompPa , palabra>
```

```
        xor cx,cx
        mov cx,8      ; numero de posiciones a comparar palabra> = 8 POSICIONES
        LEA si, msjContadorCompPa
        LEA di, auxContador
        repe cmpsb
        JNE ErrorNoExist

        print salto
        print msjTotalPalabras
        contadorPalabras
        IntToString pcontador,contadorVer ; convertimos contador a numeros
        print contadorVer
        print salto
        salirm:
        salirMenu

        HIATOc:
            mov ax,@data
            mov ds,ax
            print salto
            print msjTotalPalabras



            IntToString esHiatoContGen , contadorVer ; convertimos contador a num
eros
            print contadorVer
            print salto
            salirMenu

        TRIPTONGOc:
            mov ax,@data
            mov ds,ax
            print etiquetaPruebas
            print salto
            print msjContadorCompTri
            jmp exit

        DIPTONGOc:
            mov ax,@data
            mov ds,ax
            print salto
            print etiquetaPruebas
            print salto
            print msjContadorCompDi
```

```
        jmp exit


PROPORCION:
    print salto
    ;print msjPropDi
    ;print salto
    ;print msjPropHi
    ;print salto
    ;print msjPropTri
    ;print salto
    print msjComando3
    obtenerTexto auxProp
    print auxProp
    print salto
    ;COMPARACIONES
    ;------------PARA HIATOS
    mov cx,6  ; hiato> 6 posiciones
    mov AX, DS
    mov ES, AX

    LEA si, msjPropHi
    LEA di, auxProp
    repe cmpsb
    JE HIATOp

    ;----------PARA TRIPTONGOS
    xor cx,cx
    mov cx,10  ;triptongo>
    lEA si, msjPropTri
    LEA di, auxProp
    repe cmpsb ;Compare msjContadorCompTri:auxContador
    JE TRIPTONGOp

    ;----------PARA DIPTONGO   diptongo>

    xor cx,cx
    mov cx,9      ; numero de posiciones a comparar diptongo> = 9 posiciones
    LEA si, msjPropDi
    LEA di, auxProp
    repe cmpsb
    JE DIPTONGOp
```

```
        jmp exit


        HIATOp:
        mov ax,@data
        mov ds,ax
        ;print salto
        ;print msjPropHi
        print salto

        ;call XOR_REG
        xor si,si
        mov ax ,esDipContadorGen ; ax = contadorhiato
        mov bx,100 ;            ax * 100
        mul bx                  ;   *
        contadorPalabras        ; llamar al metodo que cuanto cuantas todas las p
alabras
        mov bx,pcontador ;      resultado de conteo -> bx = resultado
        div bx                  ; resultado de multiplicacion dividido por result
ado de conteo
        ;print salto
        ;imprimir_reg ax
        ;print salto
        ;imprimir_reg dx
        ;print salto
        ;limpiar contadoraux,15,0 ;limpiamos el contador
        ;mov si, ax
        ;imprimir_reg ax
        ;print salto
        mov contadoraux,ax ; movemos ax a contador
        ;print contadoraux
        ;print salto
        ;print pruebaSimb
        ;print salto
        IntToString contadoraux,contadorVer ; convertimos contador a numeros
        print contadorVer
        call XOR_REG
        ;print msjPunto1
        cmp dx,0
        JE terminarProp

        ;decimalProp:
            ;print salto
            print msjPunto1
            mov contadoraux,dx
```

```
            IntToString contadoraux,contadorVer
            print contadorVer
            ;print msjPorce



        terminarProp:
            ;print msjEspacio
            print msjPorce
            print salto


        jmp Menu

        TRIPTONGOp:
        mov ax,@data
        mov ds,ax
        print salto
        print etiquetaPruebas
        print msjPropTri
        print salto
        jmp Menu
        DIPTONGOp:
        mov ax,@data
        mov ds,ax
        print salto
        print msjPropDi
        print salto
        jmp Menu
COLOREAR:

    ;----------CON ESTO DE ENTRA A MODO VIDEO-------------------
    ;al inicializar el modo video le decimos que empiece en la posicion 0
    ; se inicializa modo video con una resolucion de 80x25
    mov ah, 0
    mov al, 03h
    int 10h
    ;-------------------------------------------------------------
    imprimir saludo
    imprimir saltoLinea
    ;-------------------------------------------------------------------------------
------------------
    ;por que ya imprimimos las palabras de arrbia por eso usamos lo siguiente par
a guardar la posicion de
    ;fila y columna
```

```
    mov ah, 03h ; con el 03h le decimos que analice despues del texto anterior
    mov bh, 00h
    int 10h ;dh guarda el valor de la ultima posicion fila y dl guarda la ultima
posicion de la columna
    ;-------------------------------------------------------------------------
-----------------
    ; -----------ACTUALIZAMOS POSICIONES
    mov fila, dh
    mov columna, dl
    mov si, 0
    mov di, 0

ciclo1:
    call XOR_REG
    ;-------------------------------------------------------------------------
-----------
    ;posicionar al cursor donde corresponde
    posicionarCursor fila, columna

    esTriptongo bufferInformacion[si], bufferInformacion[si+1], bufferInformacion
[si+2]
    esDiptongo bufferInformacion[si], bufferInformacion[si+1]
    esHiato bufferInformacion[si], bufferInformacion[si+1]

    cmp bl,0
    JNE esTripPrint
    cmp al,0
    JNE esDipPrint
    cmp cl,0
    JNE esHiatoPrint
    JMP letra

    esDipPrint:
        ;pintamos el diptongo
        imprimirVideo bufferInformacion[si], 0010b ;imprimos verde
        inc columna ;aumenta la posicion del cursor
        inc si

        posicionarCursor fila, columna


        imprimirVideo bufferInformacion[si], 0010b ;imprimos verde
        jmp siguiente

        ;letra:
```

```
        ;   imprimirVideo bufferInformacion[si], 1111b ;imprimos blanco
        ;   jmp siguiente
;--------------------------------------------------------------------------------
------
    esTripPrint:
        ;pintamos el diptongo
        imprimirVideo bufferInformacion[si], 1110b ;imprimos amarillo
        inc columna ;aumenta la posicion del cursor
        inc si

        posicionarCursor fila, columna


        imprimirVideo bufferInformacion[si], 1110b ;imprimos amarillo
        inc columna ;aumenta la posicion del cursor
        inc si

        posicionarCursor fila, columna
        imprimirVideo bufferInformacion[si],1110b ;imprimos amarillo
        jmp siguiente
;--------------------------------------------------------------------------------
-------
    esHiatoPrint:
    ;pintamos el hiato
        imprimirVideo bufferInformacion[si], 0100b ;imprimos rojo
        inc columna ;aumenta la posicion del cursor
        inc si

        posicionarCursor fila, columna


        imprimirVideo bufferInformacion[si], 0100b ;imprimos rojo
        jmp siguiente


;--------------------------------------------------------------------------------
-------
    letra:
        imprimirVideo bufferInformacion[si], 1111b ;imprimos blanco
        jmp siguiente
;--------------------------------------------------------------------------------
------

    siguiente:
```

```
    inc columna ;aumenta la posicion del cursor
    inc si



    cmp columna, 80d
    jl noSalto
        mov columna,0
        inc fila
    noSalto:

    cmp bufferInformacion[si], 36d ; $
    jne ciclo1


    inc fila
    mov ah,02h
    mov dh,fila
    mov dl,0
    mov bh,0
    int 10h

    JMP Menu

exit:
    close

main endp
end main
```

DESCRIPCION DE MACROS USADAS EN PROGRAMA

```
print macro buffer ;imprime cadena
push ax
push dx
    mov ax, @data
    mov ds,ax
    mov ah,09h ;Numero de funcion para imprimir buffer en pantalla
    mov dx,offset buffer ;equivalente a que lea dx,buffer, inicializa en dx la po
sicion donde comienza la cadena
    int 21h
pop dx
pop ax
endm

getChar macro  ;obtiene el caracter y se almacena el valor en el registro al

    mov ah,01h ; se guarda en al en codigo hexadecimal
    int 21h

endm


close macro ;cierra el programa
    mov ah, 4ch ;Numero de funcion que finaliza el programa
    xor al,al
    int 21h
endm


getChar macro  ;obtiene el caracter y se almacena el valor en el registro al

    mov ah,01h ; se guarda en al en codigo hexadecimal
    int 21h

endm

obtenerTexto macro buffer
```

```
LOCAL ObtenerChar, endTexto
    xor si,si ; xor si,si = mov si,0

    ;while (caracter != "/n"){
        ;buufer[i] = caracter
        ;i++
        ;}
    ObtenerChar:
        getChar
        cmp al,0dh ; ascii de salto de linea en hexa
        je endTexto

        ; si = 1

        mov buffer[si],al ;mov destino, fuente arreglo[1] = al
        ; si = 2
        inc si ; si = si + 1 // si++
        jmp ObtenerChar

    endTexto:

        mov al,24h ; asci del singo dolar $
        mov buffer[si], al

endm

; en este macro limpiamos el arreglo con $
limpiar macro buffer, numbytes, caracter
LOCAL Repetir
push si
push cx

    xor si,si
    xor cx,cx
    mov cx,numbytes ; le pasamos a cx el tamaño del arreglo
    ;si = 0
    Repetir:
        mov buffer[si], caracter ; le asignamos el caracter que estamos mandando
        inc si ; si ++
        Loop Repetir ; se va a repetir hasta que cx sea 0
pop cx
pop si
endm

obtenerRuta macro buffer
```

```
LOCAL ObtenerChar, endTexto
    xor si,si ; xor si,si = mov si,0

    ObtenerChar:
        getChar
        cmp al,0dh ; ascii de salto de linea en hexa
        je endTexto
        mov buffer[si],al ;mov destino, fuente
        inc si ; si = si + 1
        jmp ObtenerChar

    endTexto:
        mov al,00h ; CARACTER NULO
        mov buffer[si], al
endm

abrir macro buffer,handler

    mov ah,3dh ;funcion para abrir fichero
    mov al,02h ;010b Acceso de lectura/escritura. 010b
    lea dx,buffer ;carga la dirección de la fuente (buffer) a dx
    int 21h ;ejecutamos la interrupción
    jc Error1 ;salta si el flag de acarreo = 1
    mov handler,ax ;sino hay error  en ax devuelve un handle para acceder al fich
ero

endm


cerrar macro handler

    mov ah,3eh
    mov bx, handler
    int 21h
    jc Error2
    mov handler,ax

endm

leer macro handler,buffer, numbytes

    mov ah,3fh ;interrupción para leer
    mov bx,handler ;copiamos en bx el handler,referencia al fichero
    mov cx,numbytes ;numero de bytes a leer, tamaño del arreglo que guarda el con
tenido
```

```
    lea dx,buffer ;carga la dirección de la variable buffer a dx
    int 21h
    jc  Error5
    ;en el buffer se guarda la información

endm

crear macro buffer, handler

    mov ah,3ch ;función para crear fichero
    mov cx,00h ;fichero normal
    lea dx,buffer ;carga la dirección de la variable buffer a dx
    int 21h
    jc Error4
    mov handler, ax ;sino hubo error nos devuelve el handler

endm

escribir macro handler, buffer, numbytes

    mov ah, 40h ;función de escritura del archivo
    mov bx, handler ;en bx copiamos el handler,
    mov cx, numbytes ;numero de bytes a escribir
    lea dx, buffer ;carga la dirección de la variable buffer a dx
    int 21h ;ejecutamos la interrupción
    jc Error3

endm

ContarUnidades macro numero
LOCAL Mientras, FinMientras

xor si,si ;registro de 16 bits
xor bx,bx

    Mientras:
    mov al,numero[si]
    cmp al,24h ;cuando encontremos $ dejamos el contador
    je FinMientras
    inc si
    jmp Mientras

    FinMientras:
    ;si el contador vale 2 el numero es > 9 si el contador vale 1 el numero <10
    mov bx,si ;bl nos indica si son unidades o si son decenas
```

```
    ; como sabemos que el numero  <2 sabemos que el resultado de guarda en bl
endm


TextoAEntero macro texto ;en el registo al se va a guardar el numero convertido
LOCAL Unidades, Decenas, Final
xor al,al
ContarUnidades texto
;bl me indica si el numero tiene unidades o tiene decenas

cmp bl,1
je Unidades
cmp bl,2
je Decenas
jmp Final

    Unidades:
        ;texto[0] -> numero de unidades en ascii
        mov al,texto[0]
        sub al, 48 ; al = al - 48
        jmp Final


    Decenas:

        ;texto[0]-> numero de decenas en ascii
        ;texto[1]-> numero de unidades en ascii
        mov al,texto[0]
        sub al,48 ; ya tengo las decenas en su valor decimal 45 -> 4
        mov bl,10
        mul bl ; al = al * bl
        ;al = 40

        xor bl,bl
        mov bl, texto[1]
        sub bl, 48 ; bl = 5

        add al,bl ;  al = 40 +5

        jmp Final

    Final:

endm
```

```
sizeNumberString macro string
LOCAL LeerNumero, endTexto
    xor si,si ; xor si,si = mov si,0
    xor bx,bx

    LeerNumero:
        mov bl,string[si] ;mov destino, fuente
        cmp bl,24h ; ascii de signo dolar
        je endTexto
        inc si ; si = si + 1
        jmp LeerNumero

    endTexto:
        mov bx,si

endm

StringToInt macro string
LOCAL Unidades,Decenas,salir, Centenas

    sizeNumberString string; en la variable bl me retorna la cantidad de digitos
    xor ax,ax
    xor cx,cx

    cmp bl,1
    je Unidades

    cmp bl,2
    je Decenas

    cmp bl,3
    je Centenas



    Unidades:
        mov al,string[0]
        sub al,30h
        jmp salir

    Decenas:
```

```
        mov al,string[0]
        sub al,30h
        mov bl,10
        mul bl


        xor bx,bx
        mov bl,string[1]
        sub bl,30h

        add al,bl

        jmp salir

;bl 1111 1111 ->255
; 0 - 999
; bx 1111 1111 1111 1111 -> 65535
    Centenas:
        ;543
        mov al,string[0] ;[0] -> 53 -> 5 en ascii
        sub al,30h;  -> 53-48 = 5 => Ax=5 => Ax-Ah,Al
        mov bx,100;  -> bx = 100
        mul bx; -> ax*bx = 5*100 = 500
        mov cx,ax; cx = 500
        ;dx = Centenas

        xor ax,ax ; ax = 0
        mov al,string[1] ;[1] -> 52 -> 4 en ascii
        sub al,30h ; -> 52-48 = 4 => Ax=4 => Ax-Ah,Al
        mov bx,10 ;  -> bx = 10
        mul bx ; -> ax*bx = 4*10 = 40

        xor bx,bx
        mov bl,string[2] ;[2] -> 51 -> 3 en ascii
        sub bl,30h ; -> 51-48 = 3 => Ax=3 => Ax-Ah,Al

        add ax,bx ; ax = 3 + 40
        add ax,cx ; ax = 43 + 500 = 543

        jmp salir

    salir:


endm
```

```
imprimir_reg macro reg  ;imprimir registros
    push dx ;bkup de dx
    push ax
    mov ah, 02h
    mov dx, reg   ;asigno a dx el reg 16 bits
        ;add dx,30h  ;sumamos para q salga el numero tal cual porq en consola +-
30h
    int 21h
    ;call XOR_REG    ;reset de registros a,b,c,d
    pop ax
    pop dx ;bkup de dx
endm

IntToString macro num, number ; ax 1111 1111 1111 1111 -> 65535
    LOCAL Inicio,Final,Mientras,MientrasN,Cero,InicioN
    push si
    push di
    limpiar number, 15 ,24h
    mov ax,num ; ax = numero entero a convertir 23
    cmp ax,0
    je Cero
    xor di,di
    xor si,si
    jmp Inicio

;ax = 123

    Inicio:

        cmp ax,0 ;ax = 0
        je Mientras
        mov dx,0
        mov cx,10
        div cx ; 1/10 = ax = 0 dx = 2
        mov bx,dx
        add bx,30h ; 1 + 48 = ascii
        push bx
        inc di  ; di = 3
        jmp Inicio

    Mientras:
        ;si = 0 , di = 3
        cmp si,di
        je Final
```

```
        pop bx
        mov number[si],bl
        inc si
        ;si = 2 di = 3
        jmp Mientras

    Cero:
        mov number[0],30h
        jmp Final

    Final:
        pop di
        pop si


endm
```

MACROS USADAS PARA LA CREACION DE ARCHIVOS

```
;============= Abrir archivo==========================
OpenFile macro buffer,handler
    local erro,fini
    mov AX,@data
    mov DS,AX
    mov ah,3dh
    mov al,02h
    lea dx,buffer
    int 21h
    ;jc Erro ; db con mensaje que debe existir en doc maestro
    mov handler,ax
    mov ax,0
    ;jmp fini
    erro:
    ;Print TItuloErrorArchivo
    mov ax,1
    fini:
endm
;============== MACRO CERRAR ARCHIVO==============
CloseFile macro handler
;mov checkopenfile,1
    mov AX,@data
    mov DS,AX
```

```
    mov ah,3eh
    mov bx,handler
    int 21h
    ;jc Error2 ; db con mensaje que debe existir en doc maestro
endm

;========== MACRO LEER ARCHIVO==========
ReadFile macro handler,buffer,numbytes
    mov AX,@data
    mov DS,AX
    mov ah,3fh
    mov bx,handler
    mov cx,numbytes ; numero maximo de bytes a leer(para proyectos hacerlo gigant
e)
    lea dx,buffer
    int 21h
;jc Error4 ; db con mensaje que debe existir en doc maestro
endm

; pendiente el de crear escribir
;======================== MACRO CREAR ARCHIVO (any extension) ==================
CreateFile macro buffer,handler
    mov AX,@data
    mov DS,AX
    mov ah,3ch
    mov cx,00h
    lea dx,buffer
    int 21h
    ;jc Error4
    mov bx,ax
    mov ah,3eh
    int 21h
endm
; ========================= MACRO ESCRIBIR EN ARCHIVO YA CREADO =================

WriteFile macro handler,buffer,numbytes
    mov AX,@data
    mov DS,AX
    mov ah,40h
    mov bx,handler
    mov cx,numbytes
    lea dx, buffer
    int 21h
endm
```