EDIN MONTENEGRO-201709311

MANUAL TECNICO

Ordenamiento en Assembler

Este programa en assembler contiene 5 archivos auxiliares con macros para que el archivo principal tengo una mejor visibilidad y llevar el orden de cada función.

Los nombres de los archivos usados son:

- File.asm
- Entrada.asm
- Macsor.asm
- Report.asm
- Vmacros.asm

MACROS EN FILE.ASM:

En este archivo se encuentran todas las macros con respecto a crear un fichero, dicho fichero se crea con el fin de generar el reporte de órdenes que se han realizado en formato ".xml", este archivo cuenta con dos macros:

- Macro Generar Reporte
  - Esta macro recibe 4 parámetros los cuales son ruta, nombre con que se desea crear el archivo, contenido de archivo, y el hand que es el nombre propio que genera asm

```
;genera reporte params: ruta, nombrefichero, #bytescontenido, contenido, handler
1 reference
generarReporte macro ruta, nomarchivo, contenido, handreport
    LOCAL inicio_
    1 reference
    inicio_:
    ;creacion de directorio
        mov ah, 39h
        lea dx, ruta
        int 21h
        ;creacion de archivo en carpeta
        mov ah,3ch
        mov cx,0
        lea dx, nomarchivo
        int 21h
        ;si resulta error en creacion
        jc ERROR7
        mov handreport, ax

        ;*******escribir encabezado******;hasta <ciclo>Primer semestre 2021</ciclo>
        mov ah,40h
        mov bx, handreport
        mov cx,381
        mov dx, offset encabezadoRep
        int 21h

        ;salto de linea
        mov ah,40h
        mov bx, handreport
        mov cx,1
        mov dx, offset nsalto
        int 21h
```

Se utiliza la interrupción 21h, con 39h, 3ch, 40h para crear, escribir y cerrar un fichero respectivamente

- Macro Obtener Bytes :

  Esta macro se utiliza con el fin de saber cuantas posiciones se debe desplazar cx para poder escribir el contenido en el reporte, recibe como parámetro el vector donde esta almacenado todo el historial (“contenido de reporte”).

```
;obtencion de caracteres validos para escribir
;params variable, contador
1 reference
obtenerBytes macro variable
    LOCAL inicio, fin
    xor cx,cx
    xor si,si
    mov cx,30000
    2 references
    inicio:
        cmp variable[si],'$'
        je fin
        inc si
        loop inicio
    11 references
    fin:
        ;se sale
72 references
endm
```

MACROS EN ENTRADA.ASM:

En este archivo se encuentran todas las macros con respecto a leer un fichero, dicho fichero se lee con el fin de obtener la entrada de números, que están en un archivo con extensión ".xml" .

- Macro "GetRuta"

  Esta macro se utiliza para obtener la ruta ingresada por el usuario además de guardar la extensio que se ingresa, para posteriormente verificar que sea ".xml"

```
;obtener la ruta ingresada param es entrada
1 reference
getRuta macro stringbuff, exten
    LOCAL ObtenerCaracter, Centinela, Extension
    xor si,si ;limpio indice de origen para evitar errores
    mov di,0
    2 references
    ObtenerCaracter:
        getCaracter ;ya esta en vmacros
        cmp al, 2eh ; si es un punto "."
        je Extension;brincar a extension
        ;if al == enter
        ;cmp al,0dh ;codigo ascii del salto de linea en hex
        ;then
        ;je Centinela ;si es igual al sato es el fin
        ;else
        mov stringbuff[si],al
        inc si ; si = si +1
        jmp ObtenerCaracter

    3 references
    Extension:
        mov stringbuff[si],al
        mov exten[di],al;aca se va contatenando el .extension
        inc si ; si = si +1
        inc di ; si = si +1
        getCaracter
        cmp al,0dh
        je Centinela
        jmp Extension
    2 references
    Centinela:
        mov al, 00h ;ingreso signo nulo en ascii
        mov stringbuff[si],al ;copio el fin en param
        mov exten[di],'$'
72 references
endm
```

- Macro “CompararExtension”

  Esta macro se utiliza para luego de haber ingresado la ruta del archivo que se desea leer y presionar la tecla “Enter” compare la extensión y verifique si es correcta, recibe dos parámetros los cuales son dos arreglos uno que es la extensión ingresada y el otro es la extensión que esta almacenada en el programa

  Que se usa para comparar

```
;valida que la extension sea .xml
1 reference
compararExtension macro ext_origin, ext_ingresada
    LOCAL INICIO,FIN
    xor si,si
    2 references
    INICIO:
        mov bh, ext_origin[si]
        mov bl, ext_ingresada[si]
        cmp bh, bl
        jnz ERROR4
        cmp bh, '$'
        jz FIN
        inc si
        jmp INICIO
    2 references
    FIN:
72 references
endm
```

- Macro “Reconocer Archivo”
  Esta macro se utiliza para hacer un análisis léxico de la entrada y poder almacenar los números que vienen en dicha entrada, para ello a través de una especie de switch case a bajo nivel se van descartando los símbolos que no interesan, la macro recibe como parámetro el vector en donde se almaceno toda la información de la entrada

```
;analizador de entrada
;almacena los numeros que estan en el xml
;params: vectorEntrada
1 reference
reconocerArchivo macro VectorIn
    LOCAL switcher,caso0,caso1,caso2,buscarPadre,get_Id,add_Numero,sal_tar,final_rec

    ;limpio contador de numeros
    mov TotalNumeros,0
    ;limpio vectores en donde almaceno numeros de entrada
    limpiarResuVec NumerosEntrada, SIZEOF NumerosEntrada, 24h
    limpiarResuVec NumerosEntradaCopia, SIZEOF NumerosEntradaCopia, 24h
    ;---
    xor si,si
    mov indice[0],'0'
    mov si,0

    2 references
    switcher:
        cmp indice[0],'0';este reconoce el padre
        je caso0
        cmp indice[0],'1';este reconoce el ID de cada numero
        je caso1
        cmp indice[0],'2';aqui se reconocen numeros y se almacenan
        je caso2
        jmp final_rec

    ;este caso ayuda a saltar el padre
    2 references
    caso0:
        cmp VectorIn[si],'<'
        je buscarPadre
        jmp sal_tar

    ;reconoce id de cada numero
    ;ayuda a ignorar esos caracteres hasta '>'
    ;luego viene el numero
    2 references
    caso1:
```

```
caso1:
    cmp VectorIn[si],'<'
    je get_Id
    jmp sal_tar

;aqui se reconocen los numeros que vienen dentro de
;>11< y se concatenan y se agregan a vector
2 references
caso2:
    cmp VectorIn[si],'<'
    je sal_tar
    cmp VectorIn[si],48;si es <0
    jb sal_tar
    cmp VectorIn[si],57;si es >9
    ja sal_tar
    jmp add_Numero



;aqui se ingnoran todos los caracteres
;hasta encontrar '>'
2 references
buscarPadre:
    SaltarChars VectorIn
    ;imprimir bufferNombre
    ;getCaracter;;;;;;;;;;;;desde aqui esta bien hay que seguir validado
    mov indice[0],'1'; aca se manda para reconder el ID de operacion
    jmp sal_tar;PARA INC SI Y RETORNAR A SWITCH

;se ignoran caracter de id's
2 references
get_Id:
    cmp VectorIn[si+1],'/';VERIFICAR SI EL SIG CHAR ES UN '/' ya que puede ser el cierre del id </numero>
    je sal_tar;SI LO ES SOLO SE RESTA Y NO SE AGREGA COMO ID

    SaltarChars VectorIn
    mov indice[0],'2';para pasar al caso 2
    jmp sal_tar;inc si y retorna switch

;aqui almaceno numero por numero de entrada
```

MACROS EN MACSOR.ASM:

En este archivo se encuentran todas las macros relacionadas al modo video que se utilizan para animar a través de barras el ordenamiento.

- Macro “setVelocidad”

  Esta macro ayuda a establecer la velocidad de animación que ingresa el usuario a través del teclado, esta animación se retrasa a través de un delay por software, este delay es otra macro que mas adelante se detalla. Esta macro recibe como parámetro la velocidad que se almacena al ingresar el usuario la misma

```
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;%%%%%%%%%%%%%%%%%%%%%%%%%%%% ESTABLECE VELOCIDAD %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 references
setvelocidad macro velRep
    LOCAL vel0,vel1,vel2,vel3,vel4,vel5,vel6,vel7,vel8,vel9,errorr,Fiiiin
    getCaracter
    cmp al, '0'
    je vel0
    cmp al, '1'
    je vel1
    cmp al, '2'
    je vel2
    cmp al, '3'
    je vel3
    cmp al, '4'
    je vel4
    cmp al, '5'
    je vel5
    cmp al, '6'
    je vel6
    cmp al, '7'
    je vel7
    cmp al, '8'
    je vel8
    cmp al, '9'
    je vel9
    jmp errorr

    2 references
    vel0:
        mov velocidad,1600
        mov msgVelocidadRelativa,30h
        mov velRep,30h
    jmp Fiiiin
    2 references
    vel1:
        mov velocidad,1500
        mov msgVelocidadRelativa,31h
        mov velRep,31h
    jmp Fiiiin
```

- Macro “EsperarBarra”

  Esta macro se utiliza en el modo video, luego de que al usuario se le ah presentado la disposición inicial de la animación de la entrada en forma de barras, con esta macro se espera a que el usuario ingrese el carácter 57 que hace referencia al keyboard scancode de la barr a espaciadora, hasta no presionar la barra no se sale de este ciclo y por lo tanto no se puede continuar con el ordenamiento

```
6 references
esperarBarra macro
    LOCAL ciclo,fiins
    punteroDSaDatos
    3 references
    ciclo:
        getCaracterSinMostrar
        cmp ah,57;se debe de presionar tecla space de lo contrario sigue en ciclo
        je fiins
    jmp ciclo

    2 references
    fiins:
72 references
endm
```

- Macro “ModoVideo”

  Esta macro se encarga de utilizar la interrupción 10h para pasar al modo video

```
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Modo Video %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

;-----------pantalla modo video 320*200
;       fil   col      fil    col
;ancho (0  ,   0  ) a ( 0  , 319 )
;largo (0  ,   0  ) a (199 ,  0  )
;ecuacion: 320*120+160

;modo_video
2 references
modoVideo macro
    mov ax,0013h
    int 10h
    mov ax,0A000h
    mov ds,ax
72 references
endm
```

- Macro “ModoTexto”
  Esta macro se encarga de regresar al modo texto y mover el datasegment a ds

```
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Modo Texto %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

6 references
modoTexto macro
    mov ax,0003h
    int 10h
    mov ax, @data
    mov ds,ax
72 references
endm
```

- Macros “PunteroDSaDatos” y “PunteroDSaVideo”
  Estas macros se encargan de mover como su nombre lo indica al datasegmente o al puntero de video una vez que se halla iniciado el modo de video para no tener que estar usando la interrupción 10h a cada momento, ya que esto no es nada óptimo.

```
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;%%%%%%%%%%%%%%%%%%%%%%%%%% PUNTERO A DS DATOS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

;Cambia DS a donde tenemos las variables
40 references
punteroDSaDatos macro
        push ax
        mov ax,@data
        mov ds,ax
        pop ax
71 references
endm

;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;%%%%%%%%%%%%%%%%%%%%%%%%%% PUNTERO A DS VIDEO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

;Cambia DS a donde tenemos las variables
10 references
punteroDSaVideo macro
        push ax
        mov ax,0A000h
        mov ds,ax
        pop ax
72 references
endm
```

- Macro “pintar_pixel”
  Esta macro es el corazón del modo video ya que se encarga de dibujar un punto o una serie de puntos indicando fila, columna y color que se desea.

```
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% DIBUJAR UN PIXEL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

;dibujar un pixel(un punto)
;params: fil,col,color
5 references
pintar_pixel macro x_,y_,color
    push ax
    push bx
    push dx
    push di
    xor ax,ax
    xor bx,bx
    xor dx,dx
    xor di,di
    mov ax,320d ;320
    mov bx,x_
    mul bx ;320*x
    add ax,y_;320*x+y
    mov di,ax;se guarda la posicion que se obtiene atraves de la ecuacion para posicionar en la memoria
    punteroDSaDatos
    mov dl,color
    punteroDSaVideo
    mov [di],dl;en esa poscion de memoria colocar el color deseado
    pop di
    pop dx
    pop bx
    pop ax
71 references
endm
```

MACROS EN REPORT.ASM:

En este archivo se encuentran las macros que ayudan a crear el reporte tales como convertir de hexadecimal a decimal

- Macro “convertirHexADec”
  Esta macro ayuda a convertir la información almacenada en hexadecimal a decimal para que el usuario al crear el reporte lo pueda almacenar en la notación correcta.

```
convertirHexADec macro vector,vectorORden
    LOCAL WHILLE,CICLOVALS,SALLLIDA
    punteroDSaDatos
    xor di,di
    mov di,0
    limpiarResuVec vectorORden, SIZEOF vectorORden, 24h
    2 references
    WHILLE:
        xor cx,cx
        mov cx,di
        cmp cl,TotalNumeros;compara si bx ya es igual a el total de numeros, si lo es se sale
    je SALLLIDA
        xor cx,cx
        mov cl,vector[di]
        convertirHexADecReport cx;mando la pos actual a convertir
        agregarNumeroVectorReporte vectorORden,valorDec
        inc di;incremento bx para la siguiente posicion
    jmp WHILLE
    2 references
    SALLLIDA:
        ;ADIOS
72 references
endm
```

```
agregarNumeroVectorReporte macro vectorCopia,VectorOriginal
    LOCAL buscar,agregar,prefinalles,finalles
    push ax
    push si
    push di
    xor si,si
    xor di,di

    xor ax,ax
    2 references
    buscar:
        cmp vectorCopia[si],'$'
        je agregar
        inc si
    jmp buscar
    3 references
    agregar:
        cmp VectorOriginal[di],'$'
    je prefinalles
        mov al,VectorOriginal[di]
        mov vectorCopia[si],al
        inc di
        inc si
    jmp agregar

    2 references
    prefinalles:
        pop di
        xor cx,cx
        mov cx,di
        inc cx
        cmp cl,TotalNumeros;compara si bx ya es igual a el total de numeros, si lo es se sale
    je finalles
        mov vectorCopia[si],','
```

EDIN MONTENEGRO-201709311

INTERRUPCIONES

Interrupciones Hardware

Las interrupciones internas son generadas por ciertos eventos que surgen durante la ejecución de un programa.

Este tipo de interrupciones son manejadas en su totalidad por el hardware y no es posible modificarlas.

Un ejemplo claro de este tipo de interrupciones es la que actualiza el contador del reloj interno de la computadora, el hardware hace el llamado a esta interrupción varias veces durante un segundo para mantener la hora actualizada.

Aunque no podemos manejar directamente esta interrupción (no podemos controlar por software las actualizaciones del reloj), es posible utilizar sus efectos en la computadora para nuestro beneficio, por ejemplo, para crear un "reloj virtual" actualizado continuamente gracias al contador del reloj interno. únicamente debemos escribir un programa que lea el valor actual del contador y lo traduzca a un formato entendible para el usuario.

Interrupciones Software

Las interrupciones de software pueden ser activadas directamente por el ensamblador invocando al número de interrupción deseada con la instrucción INT.

El uso de las interrupciones nos ayuda en la creación de programas, utilizándolas nuestros programas son más cortos, es más fácil entenderlos y usualmente tienen un mejor desempeño debido en gran parte a su menor tamaño.

Este tipo de interrupciones podemos separarlas en dos categorías: las interrupciones del sistema operativo DOS y las interrupciones del BIOS.

La diferencia entre ambas es que las interrupciones del sistema operativo son más fáciles de usar, pero también son más lentas ya que estas interrupciones hacen uso del BIOS para lograr su cometido, en cambio las interrupciones del BIOS son mucho más rápidas, pero tienen la desventaja que, como son parte del hardware son muy específicas y pueden variar dependiendo incluso de la marca del fabricante del circuito.

La elección del tipo de interrupción a utilizar dependerá únicamente de las características que le quiera dar a su programa: velocidad (utilizando las del BIOS) o portabilidad (utilizando las del DOS).

Interrupciones utilizadas:

21H: Propósito Llamar a diversas funciones del DOS.

10H: Llamar a diversas funciones de video del BIOS.