

UNIVERSIDAD SAN CARLOS DE GUATEMALA
ARQUITECTURA DE COMPUTADORAS Y ENSAMBLADORES 1
SECCIÓN B
PRIMER SEMESTRE 2021

PROYECTO 4

201801263 Audrie Annelisse del Cid Ochoa

Manual Técnico

Macros utilizados constantemente a lo largo del proyecto:

Macro	Función
<pre>Print macro Text mov ax,@data mov ds,ax mov ah,09h lea dx,Text int 21h endm</pre>	Permite imprimir cadenas, utilizando la interrupción 21h
<pre>getChar macro mov ah,01h int 21h endm</pre>	Permite obtener un carácter. Se utilizó para obtener los datos al escoger una opción. Su valor queda guardado en al.
<pre>clean macro buffer, numbytes, caracter LOCAL Repetir xor si,si ; colocamos en 0 el contador xor cx,cx ; colocamos en 0 el contador mov cx,numbytes ;le pasamos a cx el número de bytes Repetir: mov buffer[si], caracter ;le asignamos el carácter inc si ;incremento si Loop Repetir ;se va a repetir hasta que se complete el número de bytes endm</pre>	clean, fue utilizado para limpiar arreglos. Permitía llenarlos con el carácter que se escogiera, en la mayoría de los casos fue el signo de dollar.
<pre>ConvertirArreglo macro arreglo LOCAL Ini, Fin xor di,di xor bx,bx limpiar arrayTexto, SIZEOF arrayTexto, 24h Ini: cmp arreglo[di], 24h je Fin mov ah, 0 mov al, arreglo[di] mov cl, 10 div cl add al, 30h add ah, 30h mov dl, ah mov arrayTexto[bx],al inc bx mov arrayTexto[bx],dl inc bx mov al, 20h mov arrayTexto[bx],al inc di inc bx jmp Ini Fin: ;print arrayTexto endm</pre>	Convertir Arreglo, fue utilizado para convertir un arreglo de enteros a uno de texto.

<pre> imprimirArregloDesc macro arreglo LOCAL Ini, Fin xor di,di xor bx,bx mov al, lengthArreglo mov decontador,al limpiar arrayTexto, sizeof arrayTexto, 24h Ini: cmp decontador, 1 jb Fin ActualizarContadorDi decontador ;Divide mov ah, 0 mov al, arreglo[di] mov cl, 10 div cl ;Convierta a texto add al, 30h add ah, 30h mov dl, ah ;Guarda en arreglo mov arrayTexto[bx],al inc bx mov arrayTexto[bx],dl inc bx mov al, 20h mov arrayTexto[bx],al ; Cuenta inc bx sub decontador,1 jmp Ini Fin: print arrayTexto endm </pre>	<p>Imprimir arreglo, fue utilizado para imprimir un arreglo de enteros.</p>
<pre> PrintX macro numero ;Impri Local Inicio,Imprimir Pushs mov bx,4 xor ax,ax mov ax,numero mov cx,10 Inicio: xor dx,dx div cx push dx dec bx jnz Inicio xor bx,4 Imprimir: pop dx PrintNum dl dec bx jnz Imprimir Pops endm </pre>	<p>ImprimeDecimal, convierte los datos ascii en enteros y los imprime en pantalla.</p>

1. Cargar Archivo

```

;comparar cmdabrir, bufferEntrada
;je abrir
ComprobarArchivo bufferEntrada,NombreArchivo
je abrir
Comparar cmdmediana, bufferEntrada

```

Se utilizaron los siguientes Macros:

Macro	Función
<pre> ComprobarArchivo macro entrada,nameArchivo local Inicio, Nombre,Fin,Salir,Comando Pushs mov nameArchivo,0 xor di,di xor si,si Inicio: mov al,entrada[si] cmp al,"_" je Nombre cmp al,"\$" je Fin inc si jmp Inicio Nombre: inc si mov al,entrada[si] cmp al,"\$" je Comando mov nameArchivo[di],al inc di jmp Nombre Comando: mov nameArchivo[di+1],00h LeerArchivo nameArchivo Comparar cmdabrir,cmdabrir jmp Salir Fin: Comparar cmdsalir,cmdabrir jmp Salir Salir: Pops endm </pre>	<p>ComprobarArchivo, este macro analiza el texto ingresado y lo separa para comprobar que el comando es correcto, además de obtener el nombre del archivo para abrir.</p>

	<pre> leer macro handler,buffer, numbytes mov ah,3fh ;interrupción para leer mov bx,handler mov cx,numbytes lea dx,buffer ;le pasamos al buffer int 21h jc Error5 endm </pre>		<p>Leer, permite leer el arreglo de entrada los cuales contienen los datos que estaban dentro del documento que se ha abierto.</p>
	<pre> obtenerNumeros macro arreglo LOCAL Inicio, Guardar, InicioGuardado, FinalGuardado, Final xor si,si xor di, di Inicio: cmp arreglo[si],24h ;Signo dollar je Final cmp arreglo[si],3Eh ;Signo > je Guardar inc si jmp Inicio Guardar: inc si cmp arreglo[si],24h ;Signo dollar je Inicio cmp arreglo[si], 0dh ;Salto de línea je Inicio jmp InicioGuardado InicioGuardado: cmp arreglo[si], 3ch ;Signo < je FinalGuardado mov al, arreglo[si] mov numeros[di],al inc si inc di jmp InicioGuardado FinalGuardado: mov numeros[di],20h ;Espacio inc di jmp Inicio Final: inc di mov numeros[di],24h endm </pre>		<p>Este macro fue utilizado para ir leyendo el arreglo de entrada que contiene los datos del documento y conforme va avanzando, los analiza y guarda en un arreglo de números ascii.</p>

```

convertirNumero macro arreglo
    LOCAL Inicio, Guardar, Final
    mov contadorArreglo,0
    xor si,si
    xor di,di
    limpiar valorNumero, SIZEOF valorNumero, 24h

    Inicio:
        ;imprimir arreglo[si]
        cmp arreglo[si], 24h ;Signo dollar
        je Final

        cmp arreglo[si],20h ;Espacio
        je Guardar

        mov al, arreglo[si]
        mov valorNumero[di],al
        inc di
        inc si
        jmp Inicio

    Guardar:
        mov di, contadorArreglo
        mov auxsi, si
        mov auxdi, di
        obtenerSizeBuffer valorNumero
        filtrarNumero valorNumero
        ;StringToInt valorNumero

        mov al, numero2
        ;imprimeDecimal numero2

        mov numeroReal[di],al
        limpiar valorNumero, SIZEOF valorNumero, 24h
        mov si, auxsi
        mov di, auxdi
        inc si
        xor di,di
        inc contadorArreglo
        jmp Inicio

    Final:
        ;imprimeDecimal numeroReal[0]

```

Este macro permite convertir los números del arreglo utilizado para guardar los números obtenidos del análisis del archivo cargado y los convierte en números decimales.

2. Funciones

Macro	Función
<pre> MaxCalc macro array Local Inicio, Fin, Mayor, Check, Check2 Pushs xor si, si mov ax, array[0] mov maxFreq, ax ; mayor= array[0] xor bx, bx mov contador, 0 mov si, -2 Inicio: inc si inc si mov ax, array[si+2] ; Siguiente cmp ax, 36 je Fin mov ax, array[si+2] ; Siguiente cmp ax, maxFreq ja Mayor jmp Inicio Mayor: xor bx, bx xor ax, ax inc contador inc contador mov ax, array[si+2] mov maxFreq, ax jmp Inicio Fin: Pops endm </pre>	<p>MaxCalc, este macro permitía obtener el valor máximo de un arreglo. Se utilizó para obtener la moda, recorriendo el arreglo de frecuencias previamente obtenido</p>
<pre> CalcularMax macro arregloOrdenado, size Pushs xor ax, ax xor di, di mov di, size mov ax, arregloOrdenado[di] mov MayorNum, ax Pops endm </pre>	<p>CalcularMax, este macro permitía obtener el valor máximo de los datos guardados en un arreglo previamente ordenado.</p>
<pre> CalcularMin macro arregloOrdenado Pushs xor ax, ax mov ax, arregloOrdenado[0] mov MinimoNum, ax Pops endm </pre>	<p>CalcularMin, , este macro permitía obtener el valor mínimo de los datos guardados en un arreglo previamente ordenado.</p>

```

PromedioCalc macro arreglo
    Local Inicio, Dividir
    Pushs
    xor si, si
    Inicio:
        cmp si, lengthArray
        ja Dividir

        mov ax, arreglo[si]
        add sumaDatos, ax

        inc si
        inc si
        jmp Inicio
    Dividir:
        ;mov ax, sumaDatos
        ;cld
        ;mov bx, TamaArreglo
        ;idiv bx
        ;mov promedio, ax
        ;mov residuo, dx
        ;jmp Fin
        ManejaDecimal sumaDatos, TamaArreglo, residuo, promedio
        ;Decimal_Text promedio, promedioTxt
        ;Decimal_Text residuo, residuoTxt
    Pops
endm

```

PromedioCalc, este macro permite calcular el promedio con los datos del archivo cargado al programa.

```

MedianCalc macro arreglo, size
    Local Par, Impar, Fin
    mov aux2, 0
    mov aux, 0
    Pushs
    xor ax, ax
    xor bx, bx
    mov bx, 2
    mov ax, size
    add ax, 2
    cld
    idiv bx
    mov Pivote, ax
    cld
    idiv bx
    cmp dx, 0
    je Inicio
    jmp Impar
    Par:
        mov ax, size
        add ax, 2
        mov bx, 2
        cld
        idiv bx
        mov si, ax
        mov bx, arreglo[si]
        mov aux, bx
        dec si
        dec si
        mov bx, arreglo[si]
        mov aux2, bx
        mov ax, aux
        mov bx, aux2
        add ax, bx
        mov bx, 2
        mov aux, ax
        mov aux2, bx

        ManejaDecimal aux, aux2, MedDecimal, MedEntero
        jmp Fin
    Impar:
        mov ax, size
        mov bx, 2
        cld
        idiv bx
        mov si, ax
        mov bx, arreglo[si]
        mov MedEntero, bx
        jmp Fin
    Fin:
        Pops
endm

```

MedianaCalc, este macro permite calcular la mediana de los datos cargados al programa.

3. Generar Reporte

Macro	Función
<pre>crear macro buffer, handler mov ah,3ch mov cx,00h lea dx,buffer int 21h jc Error4 mov handler, ax endm</pre>	Permite crear un archivo.
<pre>escribir macro handler, buffer, numbytes mov ah, 40h mov bx, handler mov cx, numbytes lea dx, buffer int 21h jc Error3 endm</pre>	Permite escribir texto dentro del archivo creado y abierto.
<pre>cerrar macro handler mov ah,3eh mov bx, handler int 21h jc Error2 mov handler,ax endm</pre>	Permite cerrar el archivo abierto.
<pre>INI_VIDEO macro mov ax, 0013h int 10h mov ax, 0A000h mov ds, ax endm</pre>	Permite iniciar el modo video. Cambia de modo texto a modo video.

<pre> FIN_VIDEO macro mov ax, 0003h int 10h mov ax, @data mov ds, ax endm </pre>	<p>Permite finalizar el mod video y cambia a modo texto</p>
<pre> pintar_pixel macro a, b, color push ax push bx push di xor ax, ax xor bx, bx xor di, di mov ax, 320d mov bx, a mul bx add ax, b mov di, ax mov al, color mov [di], al pop di pop bx pop ax endm </pre>	<p>Permite pintar un pixel en la pantalla, teniendo un lienzo de 320*200 pixeles. Se realiza mediante coordenadas y eligiendo un color.</p>
<pre> delay macro param LOCAL ret2, ret1, finRet push ax push bx xor ax, ax xor bx, bx mov ax, param ret2: dec ax jz finRet mov bx, param ret1: dec bx jnz ret1 jmp ret2 finRet: pop bx pop ax endm </pre>	<p>Delay, permite generar un retardo mediante la simulación de un bucle.</p>

<pre> pintar_marco macro izq, der, arr, aba, color LOCAL ciclo1,ciclo2 push si xor si,si mov si, izq ciclo1: pintar_pixel arr, si, color pintar_pixel aba, si, color inc si cmp si, der jne ciclo1 xor si, si mov si, arr ciclo2: pintar_pixel si, der, color pintar_pixel si, izq, color inc si cmp si, aba jne ciclo2 pop si endm </pre>		<p>Pintar_macro, permite pintar marcos mediante medidas de inicio en el eje x, fin en el eje x, inicio en el eje y , fin en el eje y. Así como el color.</p>
<pre> Datos_Video macro push ax mov ax, 0A000h mov ds, ax pop ax endm </pre>		<p>Datos_video, permite pasar de modo video a modo Texto, sin necesidad de sacarte de la aplicación.</p>
<pre> Video_Datos macro push ax mov ax, @data mov ds, ax pop ax endm </pre>		<p>Video_Datos, permite pasar de modo video a modo texto sin sacarte de la aplicación.</p>
<pre> PintarBarra macro xo,yo,yf,xf, color LOCAL ciclo1, ciclo2 xor cx, cx xor si,si mov dx, xo mov si, dx ciclo1: xor cx, cx mov dx, yo mov cx, dx ciclo2: mov al, color pintar_pixel cx,si,9d inc cx mov dx, yf cmp cx,dx jnz ciclo2 inc si cmp si, xf jne ciclo1 endm </pre>		<p>Pintar Barra, permite generar barras mediante coordenadas en diferentes posiciones del lienzo.</p>

<pre>ImprimirModoVideo macro fila,columna, texto xor ax,ax mov ah, 02h mov bh, 00h mov dh, fila mov dl, columna int 10h Video_Datos print texto Datos_Video endm</pre>	Imprimir Modo Video, permite
--	------------------------------

4. Salir