

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

PRÁCTICA 4

201801263 Audrie Annelisse del Cid Ochoa

Manual Técnico

Macros utilizados constantemente a lo largo del proyecto:

Macro	Función
<pre>print macro p1 mov ax,@data mov ds,ax lea dx, p1 ; E mov ah,09h ;Nu ;mov dx, offse int 21h endm</pre>	<p>Permite imprimir cadenas, utilizando la interrupción 21h</p>
<pre>getChar macro mov ah,01h int 21h endm</pre>	<p>Permite obtener un carácter. Se utilizó para obtener los datos al escoger una opción. Su valor queda guardado en al.</p>
<pre>;en el macro limpiar vamos a limpiar el arr limpiar 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 tam Repetir: mov buffer[si], caracter ;le asigno inc si ;incremento si Loop Repetir ;se va a repetir hasta endm</pre>	<p>Limpiar, fue utilizado para limpiar arreglos. Permitía llenarlos con el carácter que se escogiera, en la mayoría de los casos fue el signo de dollar.</p>
<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>	<p>Convertir Arreglo, fue utilizado para convertir un arreglo de enteros a uno de texto.</p>

<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> imprimeDecimal macro dato mov ah, 0 mov al, dato mov cl, 10 div cl add al, 30h add ah, 30h mov bl, ah mov dl, al mov ah, 02h int 21h mov dl, bl mov ah, 02h int 21h mov cx, 2 endm </pre>	<p>ImprimeDecimal, convierte los datos ascii en enteros y los imprime en pantalla.</p>

<pre> longitudArreglo macro arreglo LOCAL Ini, Fin mov lengthArreglo, 0 mov der, 0 xor si, si Ini: cmp arreglo[si], 24h je Fin inc si add lengthArreglo, 1 add der, 1 jmp Ini Fin: endm </pre>	<p>longitudArreglo, permite obtener el tamaño de un arreglo y lo guarda en la variable lengthArreglo.</p>
<pre> copiarArreglo macro origen, destino LOCAL Ini, Fin xor bx, bx xor si, si limpiar destino, SIZEOF destino, 24h Ini: mov bl, lengthArreglo cmp si, bx je Fin mov al, origen[si] mov destino[si], al inc si jmp Ini Fin: endm </pre>	<p>CopiarArreglo, permite mover todos los datos de un arreglo a otro.</p>
<pre> Dividir macro numero1, numero2 mov ah, 0 mov al, numero1 mov cl, numero2 idiv cl ;add al, 30h ;add ah, 30h ;mov bl, ah mov cociente, al mov gap, al mov residuo, bl ;imprimir cociente endm </pre>	<p>Dividir, realiza una división de enteros</p>

<pre> ActualizarContadorSi macro num LOCAL Ini, Fin xor si,si mov auxContador,0 Ini: mov al, auxContador cmp al, num je Fin inc si add auxContador,1 jmp Ini Fin: endm </pre>	<p>ActualizarcontadorSi, permite obtener el registro índice si, en el número deseado mediante un parámetro entero.</p>
<pre> ActualizarContadorDi macro num LOCAL Ini, Fin xor di,di mov auxContador,0 Ini: mov al, auxContador cmp al, num je Fin inc di add auxContador,1 jmp Ini Fin: endm </pre>	<p>Realiza la misma función que el anterior, pero para el registro índice di.</p>

1. Cargar Archivo

```
CargarArchivo:
    Saltolinea
    print menu01
    call AbrirArchivo

    jmp MenuPrincipal
AbrirArchivo:
    print carga1
    Saltolinea
    limpiar numeros, SIZEOF numeros,24h
    limpiar bufferentrada, SIZEOF bufferentrada,24h
    obtenerRuta bufferentrada
    abrir bufferentrada, handlerentrada
    limpiar bufferInformacion, SIZEOF bufferInformacion,24h
    leer handlerentrada, bufferInformacion, SIZEOF bufferInformacion
    call cerrarArchivo
    obtenerNumeros bufferInformacion
    convertirNumero numeros ;Resultados en el arreglo
    longitudArreglo numeroReal
    print finCarga
```

Se utilizaron los siguientes Macros:

Macro	Función
<pre>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 hex je endTexto mov buffer[si],al ;mov destino, fuente inc si ; si = si + 1 jmp ObtenerChar endTexto: mov al,00h ; asci del caracter nulo mov buffer[si], al endm</pre>	<p>Obtener Ruta, este macro funciona similar a obtener texto, dado que permite ingresar una serie de caracteres y termina en cuando encuentre un salto de línea. Fue visto en clase.</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. Ordenar
 - a. Método Burbuja

```
Burbuja:
    print msjVelocidad
    getChar
    sub al, 48
    mov velocidadBubble, al
    SaltoLinea
    add velocidadBubble, 30h
    ;print velocidadBubble
    print menu5
    print msjAscendente
    print msjDescendente
    SaltoLinea
    getChar
    cmp al, 31h
    je BurbujaAscendente

    cmp al, 32h
    je BurbujaDescendente
```

Macro	Función
-------	---------

```

BurbujaAsc macro arreglo
    Local Ini, For2, Fin, Swap
    limpiar arregloOrdenado, SIZEOF arregloOrdenado, 24h
    mov aux, 0
    mov i, 1
    mov j, 0
    xor si, si
    Ini:
        add i, 1
        mov j, 0
        xor si, si

        imprimirArreglo arreglo
        SaltoLinea
        ;Graficar arreglo
        mov al, lengthArreglo
        cmp i, al
        jae Fin      ; si i >= lengthArreglo
        jb For2      ; si i < lengthArreglo

    For2:
        mov al, lengthArreglo
        sub al, 1    ; lengthArreglo-1

        cmp j, al
        jae Ini      ; j >= lengthArreglo-1

        mov bl, arreglo[si]

        cmp bl, arreglo[si+1]
        ja Swap      ; arreglo[si] > arreglo[si+1]

        add j, 1
        inc si
        jmp For2
    Swap:

        mov al, arreglo[si]      ; aux = arreglo[si]
        mov aux, al

        mov al, arreglo[si+1]    ; al = arreglo[si+1]

        mov arreglo[si], al      ; arreglo[si] = arreglo[si+1]

        mov al, aux              ; al = aux
        mov arreglo[si+1], al    ; arreglo[si+1] = aux

```

Este macro permite realizar el ordenamiento burbuja de forma ascendente, se planteo igual que una parte de código java del mismo método. También se realizó un procedimiento para ordenar los datos de forma descendente, en ese caso sólo cambió la condición de salto en la parte del swap. De ja pasó a jb para indicar menor.

```

QuickSortAscP proc

    mov cx,der
    cmp izq,cx
    jge fin
    partitionAsc arrayQuick, izq, der
    mov bx, der
    push bx
    mov ax, resultadoPartition
    push ax
    dec ax
    mov der,ax

    imprimirArreglo arrayQuick
    SaltoLinea
    ;tienen que graficar aqui

    call QuickSortAscP
    pop ax
    pop bx
    mov der,bx
    inc ax
    mov izq,ax
    call QuickSortAscP
fin:
    ret
QuickSortAscP endp

```

Para el procedimiento Quicksort visto en clase se utilizó procedimiento en vez de un macro, dado que era recursivo.

<pre> ShellSort macro arreglo LOCAL Ini,For2,For3,Swap,Fin,prueba limpiar arregloOrdenado, SIZEOF arregloOrdenado, 24h mov i, 0 mov j, 0 xor si,si xor di,di mov al, lengthArreglo mov gap,al ;salto = A.lenth mov aux1,0 mov aux2,0 mov cond,0 Ini: Dividir gap, 2 ;salto /=2 ;imprimeDecimal gap mov al, gap ;ActualizarContadorSi gap imprimirArreglo arreglo Saltolinea mov cond,1 ;cambios =true cmp gap, 0 jbe Fin ; si gap <= 0 ja For2 ; si gap > 0 prueba: imprimeDecimal gap jmp Ini For2: cmp cond,0 je Ini mov cond,0 ; cambios =false mov al, gap mov i, al ; i=salto call For3 jmp For2 For3: mov al, lengthArreglo cmp i, al ; i>=lengthArreglo jae For2 ActualizarContadorSi i ; si=i </pre>	<p>Para el macro shellSort se guió por un ejemplo visto en internet en código java.</p>
--	---

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>	<p>Permite crear un archivo.</p>
<pre> escribir macro handler, buffer, numbytes mov ah, 40h mov bx, handler mov cx, numbytes lea dx, buffer int 21h jc Error3 endm </pre>	<p>Permite escribir texto dentro del archivo creado y abierto.</p>

	<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>		Permite finalizar el mod video y cambia a modo texto
	<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>		Permite pintar un pixel en la pantalla, teniendo un lienzo de 320*200 pixeles. Se realiza mediante coordenadas y eligiendo un color.

	<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>		<p>Imprimir Modo Video, permite</p>

4. Salir