UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1
SEGUNDO SEMESTRE 2020
ING. OTTO ESCOBAR
TUTOR ACADÉMICO SECCIÓN A: HERBERTH ARGUETA

MANUAL TECNICO: PRACTICA 4

NOMBRE: Juan José Ramos Campos
CARNET: 201801262
FECHA: 12/10/20

Contenido

Macros.inc

Este es el archivo principal de la aplicación, cuenta con una serie de macros especiales que le permiten realizar diferentes cosas entre ellas:

print

```
print macro cadena ; 
MOV ah,09h
MOV dx,@data
MOV ds,dx
MOV dx, offset cadena
int 21h
endm
```

Esta macro sirve para hacer un print en pantalla de una cadena que recibe como parámetro

getText

```
getText macro buffer
LOCAL CONTINUE, FIN
PUSH SI
PUSH AX

xor si,si
CONTINUE:
getChar
cmp al,0dh
je FIN
mov buffer[si],al
inc si
jmp CONTINUE

FIN:
mov al, '$'
mov buffer[si],al

POP AX
POP SI
endm
```

Esta macro sirve para poder escribir desde el teclado, recibe como parámetro un arreglo de caracteres donde se almacenara cada carácter pulsado en el teclado

getfecha y getHora

```
;OBTENER FECHA
getfecha macro
    MOV AH,2AH
    INT 21H
endm

;OBTENER HORA
gethora macro
    MOV AH,2CH
    INT 21H
endm
```

Esta macros sirven para obtener la fecha y hora de la terminal de Windows

obtenerFechaHora

```
;OBTENER FECHA
obtenerFechaHora macro dia, mes, ano,
    XOR AX,AX
    XOR BX,BX
    XOR CX,CX
    getfecha
    MOV BX,CX
    convertirString1 BX,ano
    XOR BX,BX
    getfecha
    MOV BL,DH
    convertirString1 BX,mes
    getfecha
    MOV BL,DL
    convertirString1 BX,dia
    XOR BX,BX
    gethora
    MOV BL,CH
    convertirString1 BX,hora
    gethora
    MOV BL,CL
    convertirString1 BX,minuto
    gethora
    MOV BL,DH
    convertirString1 BX,segundo
endm
```

esta macro coloca en los arreglos correspondientes, los datos de fecha y hora obtenidos con la macro anterior.

CovertirString

```
ConvertirString macro buffer
    LOCAL Dividir,Dividir2,FinCr3,NEGATIVO,FIN2,FIN
    xor si,si
    xor cx,cx
    xor bx,bx
    xor dx,dx
    mov dl,0ah
    test ax,1000000000000000
    jnz NEGATIVO
    jmp Dividir2

    NEGATIVO:···
    Dividir:
        xor ah,ah
    Dividir2:···
    FinCr3:···
    FIN:
endm
```

Esta macro convierte a string un numero

ConvertirAscii

```
CovertirAscii macro numero
    LOCAL INICIO,FIN
    xor ax,ax
    xor bx,bx
    xor cx,cx
    mov bx,10   ;multiplicador 10
    xor si,si
    INICIO:···
    FIN:
endm
```

Esta macro, al contrario que la anterior, convierte un numero a string

saveOperaciones

```
saveopreciones macro bufferOperacion, bufferResultado, buf
LOCAL INICIO, FIN, INSERTOPERATION,INSERTARRESULTADO1,INSE
MOV SI, 0
MOV DI, 0
INICIO:···
INSERTOPERATION:···
INSERTARRESULTADO1:···
INSERTARRESULTADO:···
FIN:
MOV bufferOperaciones[SI], ','

clear bufferOperacion
clear bufferResultado
endm
```

Esta macro se encarga de almacenar en un arreglo las operaciones realizadas en el archivo de entrada

SavePadre

```
saveopreciones macro bufferOperacion, bufferResultado, buf
LOCAL INICIO, FIN, INSERTOPERATION,INSERTARRESULTADO1,INSE
MOV SI, 0
MOV DI, 0
INICIO:···
INSERTOPERATION:···
INSERTARRESULTADO1:···
INSERTARRESULTADO:···
FIN:
MOV bufferOperaciones[SI], ','

clear bufferOperacion
clear bufferResultado
endm
```

Esta macro, guarda el identificador padre del archivo de entrada

Estad.inc

CompararOperacion

```
COMPARAROPERACION MACRO buffer
LOCAL INICIO, GETPADRE, SALIR2, SALIR1, VE
INICIO:
    CMP buffer[SI], '$'
    JE SALIR1
    MOV AL, buffer[SI]
    MOV padre[DI], AL
    INC SI
    INC DI
    JMP INICIO
SALIR1:
    MOV SI,0

GETPADRE: ···
SALIR2:

VERIFICAR:  ···
SIEXISTE:
    print exitobusqueda
    INC SI
NUMERO: ···
REGRESAR: ···
ENDF:
    print errorbusqueda
    print padre

TERMINAR:
    clear padre
ENDM
```

Esta macro sirve para mostrar el resultado de una operación que venia en el archivo de entrada

ComparарExit

```
COMPARAREXIT MACRO buffer
LOCAL IGUAL, DIFERENTE

    xor CX, CX
    mov cx,4   ;Determinamos la cantidad de
    mov AX,DS  ;mueve el segmento datos a A
    mov ES,AX  ;Mueve los datos al segmento
    lea si, buffer                ; Reset p
    lea di, exit

    repe cmpsb  ;compara las dos cadenas
    je IGUAL ;Si fueron iguales
    jne DIFERENTE

    IGUAL:
        jmp MENUPRINCIPAL
    DIFERENTE:

ENDM
```

Esta macro sirve para comparar el comando EXIT y salir de consola

Ctrl.inc

getSize

```
getSize macro buffer
LOCAL INICIO, LIMPIAR
XOR SI,SI   ;Seteamos los regi
XOR CX,CX
XOR AX,AX
LEA SI,[buffer] ;Movemos la pa
INICIO:
    LODSB ;Extraemos el ultimo
    CMP AL,'$' ;SI ES IGUAL A
    JE LIMPIAR
    INC CX    ;INCREMENTAMOS E
    JMP INICIO  ;volvemos al b
LIMPIAR:;CONTADOR TENDRA LA LO
    xor si, si
    xor di, di

endm
```

Esta macro retorna el size del archivo de entrada, es decir, la cantidad de caracteres encontrados se utiliza para hacer loops

limpiar

```
limpiar macro buffer
LOCAL EX, SALIR, CORREGIR, CAD, NUM, CAD1, CAD2

getSize buffer
EX: ···
CAD1: ···
NUM: ···
CAD: ···
CAD2: ···
CORREGIR: ···
SALIR:

endm
```

Esta macro recibe como parámetro, el texto que contiene el archivo de entrada y lo limpia, es decir, le quita todos los caracteres no necesarios para su ejecucion

prueba

```
prueba macro buffer, indice
LOCAL INICIO, FIN,CADENA, GETNUM
INICIO: ···
CADENA:
    INC indice

    XOR DI, DI
    getIdentificador buffer, indice, DI
    INC indice
    JMP INICIO
GETNUM: ···
GETENDTEMP: ···
GETENDTEMP2: ···
SALIR:
endm
```

Esta macro se encarga de obtener los identificadores y los números a operar del archivo de entrada, tanto los identificadores como los números los toma por separado, no interfiere ente ambos, cuando encuentra el patron siguiente "} , { " es decir, llave cierre, coma, llave apertura, entonces significa que puede venir otro identificador, entonces guarda el identificador anterior y el resultado de las operaciones anteriores.

ObtenerNumero

```
obtenerNumero macro buffer, indice, start
LOCAL INICIO, FIN, SALIR, SUMA, RESTA, MULTI, DIVI, SALIR2, INIT

INIT: ···
INICIO: ···
FIN1: ···
FIN: ···
SUMA:
    PUSH CX
    POP AX
    add ax, bx
    JMP SALIR2

RESTA: ···
MULTI: ···
DIVI: ···
SALIR: ···
SALIR2: ···
ENDF:
endm
```

Talvez la macro mas importante, puesto que esta se encarga de convertir los "numeros" que vienen en la cadena a números reales y operar con ellos, la lógica es la siguiente, convierto un numero, positivo o negativo y lo inserto a la pila, pero antes de insertar verifico si no hay otro numero, si lo hay, se saca el numero, con el signo, se operan y se inserta en la pila el resultado.

evaluarNumero

```
evaluarNumero macro buffer
LOCAL NEGAR, SALIR, NEGAR2, NEGAR3

    CMP buffer[0], '-'
    JE NEGAR
    CovertirAscii buffer
    JMP SALIR
NEGAR:
    MOV DI, 1
    JMP NEGAR2
NEGAR2:
    MOV AL, buffer[DI]
    CMP AL, '$'
    JE NEGAR3
    mov temporal2[DI-1], AL
    INC DI
    JMP NEGAR2
NEGAR3:
    CovertirAscii temporal2
    not AX
    ADD AX, 1
    JMP SALIR
SALIR:
    clear temporal2

endm
```

esta macro sirve para evaluar si un numero es positivo o negativo y convertirlo mediante el primer carácter del string, si es un “-” significa que es negativo, se hacen los arreglos necesarios y se inserta el valor. Si es positivo solo se convierte y se inserta a la pila.

isnertarSimbolo

```
insertarSimbolo macro char
    MOV AX, char
    PUSH AX
endm
```

inserta a la pila el símbolo que se detecta en la macro **getIdentificador**

getIdentificador

esta macro es llamada en la macro **prueba** cuando se detectan las comillas (“), recibe el buffer y el índice desde donde buscar, concatena caracteres hasta encontrar las comillas de cierre (“) y los guarda en un arreglo especial, también busca los símbolos de suma, resta, multiplicación y división y los manda a guardar, así como también las palabras reservadas add, sub, mul, div que guarda los símbolos también

```
getIdentificador macro buffer, INICIO, posicion
LOCAL SALIR, OBTENER, VERSUMA, VERRESTA,  VERMULTI, VERDIVI, INSE
OBTENER:

    mov AL, buffer[INICIO]
    CMP AL, '"'
    JE SALIR
    CMP AL, 'a'
    JE VERSUMA
    CMP AL, 'A'
    JE VERSUMA
    CMP AL, 's'
    JE VERRESTA
    CMP AL, 'm'
    JE VERMULTI
    CMP AL, 'd'
    JE VERDIVI
    cmp AL, '+'
    JE GETMAS
    cmp AL, '-'
    JE GETMENOS
    cmp AL, '*'
    JE GETPOR
    cmp AL, '/'
    JE GETDIV
```

clear

```
clear macro buffer
LOCAL INICIO, SALIR
    PUSH DI
    xor DI,DI
INICIO:
    CMP DI,300
    JE SALIR
    CMP buffer[DI], '$'
    JE SALIR

    MOV buffer[DI], '$'
    INC DI
    JMP INICIO
SALIR:
    XOR DI, DI
    POP DI
endm
```

esta macro se encarga de limpiar los arreglos que sirven como variables.

Archivo.inc

Openfile

```
openFile macro ruta,handle
mov ah,3dh
mov al,10b
lea dx,ruta
int 21h
mov handle,ax
jc ErrorAbrir
endm
```

Esta macro se encarga de abrir un archivo según la path o nombre

CloseFile

```
closeFile macro handle
mov ah,3eh
mov handle,bx
int 21h
endm
```

Se encarga de cerrar el archivo abierto

CréateFile

```
createFile macro buffer,handle
mov ah,3ch
mov cx,00h
lea dx,buffer
int 21h
mov handle,ax
jc ErrorCrear
endm
```

Se encarga de crear un archivo según el tipo de extensión

writeFile

```
writeFile macro numbytes,buffer,handle
    mov ah, 40h
    mov bx,handle
    mov cx,numbytes
    lea dx,buffer
    int 21h
    jc ErrorEscribir
endm
```

sirve para escribir dentro del archivo