
MANUAL TÉCNICO - CALCULADORA

Lab Arquitectura de compiladores y ensambladores 1 - practica 5 - MASM

Desarrollador

José Daniel López Gonzalez - 201503836

Manual técnico

Microsoft Macro Assembler (MASM)

Es un ensamblador para la familia x86 de microprocesadores. Fue producido originalmente por Microsoft para el trabajo de desarrollo en su sistema operativo MS-DOS, y fue durante cierto tiempo el ensamblador más popular disponible para ese sistema operativo. El MASM soportó una amplia variedad de facilidades para macros y programación estructurada, incluyendo construcciones de alto nivel para bucles, llamadas a procedimientos y alternación.

Partes del código

1. Include
2. .model small
3. .stack 64
4. .data
5. .code
6. end

Include

Segmento de código donde se enlazan a otros archivos

```
include macros.asm
```

.model small

Se declara el tipo de modelo

.stack 64

Segmento de código donde se almacena la pila, no se utilizó

.data

Segmento de código donde se almacenan las variables, constantes y arreglos

```
v_f_mes      db 2 dup (48), '$'
v_f_dia      db 2 dup (48), '$'
v_f_hr       db 2 dup (48), '$'
v_f_min      db 2 dup (48), '$'
v_f_seg      db 2 dup (48), '$'
v_num        db 0, '$'
;-----COEFICIENTES DE LA FUNCION ORIGINAL-----
coeficiente0 db 43,48,'$'
coeficiente1 db 43,48,'$' ;45 negativo
coeficiente2 db 43,48,'$'
coeficiente3 db 43,48,'$'
coeficiente4 db 43,48,'$'
num0         dd 0
num1         dd 0
num2         dd 0
num3         dd 0
num4         dd 0
numGrafica4  dd 0
numGrafica3  dd 0
numGrafica2  dd 0
numGrafica1  dd 0
numGrafica0  dd 0
signoGrafica4 db 43
signoGrafica3 db 43
signoGrafica2 db 43
signoGrafica1 db 43
signoGrafica0 db 43
f_original   db 50 dup('$')
;-----COEFICIENTES DE LA DERIVADA-----
coefDerivada1 db 43,48,'$'
coefDerivada2 db 43,48,'$'
coefDerivada3 db 43,48,'$'
coefDerivada0 db 43,48,'$'
numDerivada1 dd 0
numDerivada2 dd 0
numDerivada3 dd 0
numDerivada0 dd 0
;-----COEFICIENTES DE LA NTEGRAL-----
coefIntegral0 db 43,48,'/',48,'$'
coefIntegral1 db 43,48,'/',48,'$'
coefIntegral2 db 43,48,'/',48,'$'
coefIntegral3 db 43,48,'/',48,'$'
coefIntegral4 db 43,48,'/',48,'$'
```

```

coefIntegral5    db 43,48,'/',48,'$'
numIntegral0     dd 0
numIntegral1     dd 0
numIntegral2     dd 0
numIntegral3     dd 0
numIntegral4     dd 0
numIntegral5     dd 0

```

.code

Segmento de código donde se almacena el código el programa

Etiquetas principales

```

;=====
; Cargar archivo con funciones aritmeticas
;=====

ImprimirCadena opciones2
M_LIMPIAR_VAR v_f_in
M_LIMPIAR_VAR v_ruta
M_OBTENER_RUTA v_f_in
M_PASS_RUTA v_f_in, v_ruta
;ImprimirCadena v_ruta; muestra la ruta
M_ABRIR_ARCHIVO v_ruta, maneja
jc et_error_abrir_archivo
mov maneja, ax
M_LEER_ARCHIVO maneja, v_buffer, 100
jc et_error_leer_archivo
ImprimirCadena v_buffer
M_CERRAR_ARCHIVO maneja

```

Derivar función

```

;=====
;  DERIVAR FUNCION ORIGINAL
;=====

HacerDerivada:
    mov coefDerivada3[ 0 ],43
    mov coefDerivada2[ 0 ],43
    mov coefDerivada1[ 0 ],43
    mov coefDerivada0[ 0 ],43
    mov coefDerivada3[ 1 ],0
    mov coefDerivada2[ 1 ],0
    mov coefDerivada1[ 1 ],0
    mov coefDerivada0[ 1 ],0
    mov numDerivada3, 0
    mov numDerivada2, 0
    mov numDerivada1, 0
    mov numDerivada0, 0

Derivar4:
    cmp num4, 0
    je Derivar3

```

```

    mov EAX, num4
    mov EBX, 4
    mul EBX
    mov numDerivada3, EAX
    add EAX, 48
    mov coefDerivada3[ 1 ], al
    mov al, coeficiente4[ 0 ]
    mov coefDerivada3[ 0 ], al
Derivar3:
    cmp num3, 0
    je Derivar2
    mov EAX, num3
    mov EBX, 3
    mul EBX
    mov numDerivada2, EAX
    add EAX, 48
    mov coefDerivada2[ 1 ], al
    mov al, coeficiente3[ 0 ]
    mov coefDerivada2[ 0 ], al
Derivar2:
    cmp num2, 0
    je Derivar1
    mov EAX, num2
    mov EBX, 2
    mul EBX
    mov numDerivada1, EAX
    add EAX, 48
    mov coefDerivada1[ 1 ], al
    mov al, coeficiente2[ 0 ]
    mov coefDerivada1[ 0 ], al
Derivar1:
    cmp num1, 0
    je FinDrivacion
    mov EAX, num1
    mov EBX, 1
    mul EBX
    mov numDerivada0, EAX
    add EAX, 48
    mov coefDerivada0[ 1 ], al
    mov al, coeficiente1[ 0 ]
    mov coefDerivada0[ 0 ], al
FinDrivacion:
    ret

```

Integrar función

```

;=====
;  INTEGRAR FUNCION ORIGINAL
;=====
HacerIntegral:
    mov coefIntegral5[ 0 ], 43
    mov coefIntegral4[ 0 ], 43
    mov coefIntegral3[ 0 ], 43
    mov coefIntegral2[ 0 ], 43
    mov coefIntegral1[ 0 ], 43
    mov coefIntegral0[ 0 ], 43
    mov coefIntegral5[ 1 ], 0
    mov coefIntegral4[ 1 ], 0
    mov coefIntegral3[ 1 ], 0

```

```

mov coefIntegral2[ 1 ], 0
mov coefIntegral1[ 1 ], 0
mov coefIntegral0[ 1 ], 0
mov coefIntegral5[ 3 ], 0
mov coefIntegral4[ 3 ], 0
mov coefIntegral3[ 3 ], 0
mov coefIntegral2[ 3 ], 0
mov coefIntegral1[ 3 ], 0
mov coefIntegral0[ 3 ], 0
mov numIntegral5, 0
mov numIntegral4, 0
mov numIntegral3, 0
mov numIntegral2, 0
mov numIntegral1, 0
mov numIntegral0, 0
Integrar4:
    cmp num4,0
    je Integrar3
    mov al,coeficiente4[ 0 ]
    mov coefIntegral5[ 0 ], al
    mov al,coeficiente4[ 1 ]
    mov coefIntegral5[ 1 ],al
    mov coefIntegral5[ 3 ],53
Integrar3:
    cmp num3,0
    je Integrar2
    mov al,coeficiente3[ 0 ]
    mov coefIntegral4[ 0 ], al
    mov al,coeficiente3[ 1 ]
    mov coefIntegral4[ 1 ],al
    mov coefIntegral4[ 3 ],52
Integrar2:
    cmp num2,0
    je Integrar1
    mov al,coeficiente2[ 0 ]
    mov coefIntegral3[ 0 ], al
    mov al,coeficiente2[ 1 ]
    mov coefIntegral3[ 1 ],al
    mov coefIntegral3[ 3 ],51
Integrar1:
    cmp num1,0
    je Integrar0
    mov al,coeficiente1[ 0 ]
    mov coefIntegral2[ 0 ], al
    mov al,coeficiente1[ 1 ]
    mov coefIntegral2[ 1 ],al
    mov coefIntegral2[ 3 ],50
Integrar0:
    cmp num0,0
    je FinIntegracion
    mov al,coeficiente0[ 0 ]
    mov coefIntegral1[ 0 ], al
    mov al,coeficiente0[ 1 ]
    mov coefIntegral1[ 1 ],al
    mov coefIntegral1[ 3 ],49
FinIntegracion:
Ret

```

Dibujar ejes

```
;=====
;  DIBUJAR EJES
;=====
DibujarEjes:
    mov ecx, 0
    mov edx, 100
    ejex:
        ;pixel x, y, color
        pixel ecx, edx, 48h ;color para los ejes
        inc ecx
        cmp ecx, 320 ; posicion inicio
        jne ejex
    mov ecx, 160
    mov edx, 0
    ejey:
        pixel ecx, edx, 48h ; color para los ejes
        inc edx
        cmp edx, 200 ; posicion inicio
        jne ejey
    ret
```

End

Sentencia para finalizar el programa

```
Salir:
    mov ah, 04ch
    int 21h
```

MACROS

Imprimir en pantalla

```
M_PRINT macro buffer
    mov ax, @data          ;Registro acumulador con la dirección de dato
    mov ds, ax             ;Apuntar al segmento de dato - donde esta la cadena
    mov ah, 09h            ;F - Visualización de cadena
    mov dx, offset buffer;Desplazamiento de la cadena dentro de segmento
    int 21h                ;Ejecutar instrucción 09h
    ;Mostrar cadena de texto en consola
endm
```

Imprimir con ASCII en pantalla

```
M_PRINT_ASCII macro buffer
    mov ah, 0              ;Limpiar registro AH
    mov ah, 06h
    mov dl, buffer
```

```
int 21h
endm
```

Mostrar texto y esperar ingreso de teclado

```
M_READ macro buffer
    local et_inicio_ciclo, et_salir_ciclo
    mov dx, offset buffer    ;Obtener posición del vector
    mov bp, dx               ;Inicializar indice en la posición del vector
    et_inicio_ciclo:
        mov ah, 01h          ;Instrucción para leer un caracter
        int 21h
        cmp al, 13           ;Comparamos si hay retorno de carro CR
        je et_salir_ciclo    ;Si flag Zero está activa saltar
        mov ds:[bp], al      ;Guardar caracter en el vector
        inc bp               ;Aumentar indice
        loop et_inicio_ciclo
    et_salir_ciclo:
endm
```

Obtener ruta para guardar archivo

```
M_OBTENER_RUTA macro buffer
    local et_obtener_char, et_salir
    xor si, si               ;Limpiar Registro SI
    et_obtener_char:
        mov ah, 01h          ;Funcion de lectura
        int 21h
        cmp al, 13           ;Comparamos CR = 13
        je et_salir
        mov buffer[si], al   ;Insertar caracter en vector
        inc si               ;SI + 1
        jmp et_obtener_char
    et_salir:
        mov al, 0            ;Insertar NULL = 0
        mov buffer[si], al   ;Insertar fin de lectura
endm
```

Abrir archivo

```
M_ABRIR_ARCHIVO macro nombre, handle
    xor ax, ax
    mov ah, 3dh              ;Funcion Abrir Archivo
    mov al, 00h              ;Modo de Escritura
    lea dx, nombre
    ;mov dx, offset nombre    ;Nombre del archivo a abrir
    int 21h
    jc et_error_abrir_archivo
endm
```

Cerrar archivo

```
M_CERRAR_ARCHIVO macro handle
    xor ax, ax
    mov ah, 3eh
    mov bx, handle
    int 21h
    jc et_error_cerrar_archivo
endm
```

Escribir archivo

```
M_ESCRIBIR_ARCHIVO macro buffer, handle, num
    xor ax, ax
    mov ah, 40h
    mov bx, handle
    mov cx, num
    mov dx, offset buffer
    int 21h
    jc et_error_generar_archivo
endm
```

Leer archivo

```
M_LEER_ARCHIVO macro handle, buffer, num
    xor ax, ax
    mov ah, 3fh
    mov bx, handle
    mov cx, num
    mov dx, offset buffer
    int 21h
    jc et_error_leer_archivo
endm
```

Crear archivo

```
M_CREAR_ARCHIVO macro nombre
    xor ax, ax
    mov ah, 3ch
    mov cx, 00h
    mov dx, offset nombre
    int 21h
    jc et_error_crear_archivo
endm
```

Convertir numero a ASCII

```
M_OBTENER_NUMERO macro numero, x
```



```

    mov al, x
    aam
    add al, 48
    mov numero[1], al

    mov al, ah
    aam
    add al, 48
    mov numero[0], al
endm

```

Dibujar Pixel

```

pixel macro x, y, color
    mov ecx, x ; movemos el registro
    mov edx, y ; movemos el registro
    mov ah, 0ch ; mandamos el exadecimal 0ch al registro ah para lanzar la interrupcion
    mov al, color
    int 10h
endm

```

Obtener ruta ##ej.e.arq##

```

M_PASS_RUTA macro txt, ruta
local ciclo, fin_lectura, salir
    xor si, si
    xor bx, bx
    mov al, txt[si]
    cmp al, 35 ;Comparamos # = 35
    jne err_syntaxis
    inc si ;SI ++
    mov al, txt[si] ;Compramos # = 35
    cmp al, 35
    jne err_syntaxis
    inc si ;SI ++
    ciclo:
        mov al, txt[si]
        cmp al, 35 ;Comparamos # = 35
        je fin_lectura
        cmp al, 0 ;Comparamos NULL = 0
        je err_caracter_invalido
        cmp al, 36 ;Comparamos $ = 36
        je err_caracter_invalido
        mov ruta[bx], al;Mover el caracter a ruta[]
        inc bx
        inc si
        jmp ciclo
    fin_lectura:
        mov al, 0 ;Insertar NULL = 0
        mov ruta[bx], al;Insertar fin de lectura
        inc si ;SI ++
        mov al, txt[si]
        cmp al, 35 ;Comparamos # = 35
        jne err_syntaxis
        inc si ;SI++
        mov al, txt[si]
        cmp al, 0 ;Comparamos NULL = 0

```

```

        je salir
        jne err_syntaxis
        cmp al, 36      ;Comparamos $ = 36
        je salir
        jne err_syntaxis
    salir:
endm

```

Limpiar Variables

```

M_LIMPIAR_VAR macro var
local ciclo, salir
    xor si, si
    ciclo:
        mov var[si], 36
        cmp si, 48
        je salir
        inc si
        jmp ciclo
    salir:
Endm

```

Interrupciones


Int 21h

La mayoría de servicios ó funciones del sistema operativo MS-DOS se obtienen a través de la interrupción software 21H. Es por esto que se le denomina DOS-API: DOS-APPLICATION-PROGRAM-INTERFACE La INT 21H está compuesta por un grupo de funciones. Cuando se accede a la INT 21H, hay que indicar el número de función que queremos ejecutar. La llamada a la INT 21H se realizará como sigue:

- Introducimos en (AH) el número de función a la que deseamos acceder.
- En caso de que deseemos acceder a una sub-función dentro de una función, debemos indicarlo introduciendo en (AL) el número de esa sub-función.
- Llamar a la INT 21H.

Funciones:

- INT 21H Función **01H** - Entrada de Carácter con Eco (ó salida)
- INT 21H Función **02H** - Salida de Carácter
- INT 21H Función **09H** - Visualización de una cadena de caracteres
- INT 21H Función **0AH** - Leer cadena de símbolos desde teclado y su código se almacena al buffer
- INT 21H Función **3CH** - Crear Fichero
- INT 21H Función **3DH** - Abrir Fichero
- INT 21H Función **3EH** - Cerrar Fichero

- 
- INT 21H Función 3FH - Lectura de Fichero o dispositivo
 - INT 21H Función 40H - Escritura en Fichero o dispositivo.

Int 10h

- INT 10H Función 00H - Establecer modo de Vídeo
 - ◆ 03h - 80 por 25 - 16 - Modo Texto
 - ◆ 13h - 320 por 200 -256 - Modo Gráfico