
MANUAL TÉCNICO - GO

Lab Arquitectura de compiladores y ensambladores 1 - practica 4 - MASM

Desarrollador

José Daniel López Gonzalez - 201503836

Manual técnico

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

```
;--- Arreglos
a_matriz    db  40,42,44,46,48,50,52,54
             db  59,61,63,65,67,69,71,73
             db  78,80,82,84,86,88,90,92
             db  97,99,101,103,105,107,109,111
             db  116,118,120,122,124,126,128,130
             db  135,137,139,141,143,145,147,149
             db  154,156,158,160,162,164,166,168
```

```

a_tablero    db 173,175,177,179,181,183,185,187,'$'
a_comando    db 10 dup( 17 dup('-',10,13),10,13, '$'
a_temporal   db 5 dup('0'), '$'

txt_temp1    db 'igual','$'
txt_temp2    db 'diferente','$'
;----- D - variables -----
v_ficha      db 'n','$'
v_pts_b      db '0', '$'
v_pts_n      db '0', '$'
v_mov        db '0', '$'
v_tmp        dw '0', '$'
v_tmp2       dw '0', '$'
v_tmp3       dw '0', '$'
v_tmp4       dw '0', '$'
v_pass_b     db 00h, '$'
v_pass_n     db 00h, '$'
v_f_ano      db '2020', '$'
v_f_mes      db 2 dup (48), '$'
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, '$'
;----- M - ARCHIVOS -----
v_f_html     db '/repl.html',0
v_f_in       db 50 dup('$')
v_handler    dw ?
v_buffer     db 7000 dup('$')

```

.code

Segmento de código donde se almacena el código el programa

Etiquetas principales

- Inicio:
- Et_menu:
- Et_juego:
- Et_cargar_juego:
- salir

Código para mostrar y obtener tecla para menú

```

M_MENU txt_menu      ;Imprime menu y espera respuesta
;Comparación de ASCII
cmp al,49             ;ascii 49 = numero 1 comparar registro AL
je et_juego
cmp al,50             ;ascii 50 = numero 2 comparar registro AL
je et_cargar_juego
cmp al,51             ;ascii 51 = numero 3 comparar registro AL
je Salir
jne et_menu

```

Lectura y ejecución de comandos

```

M_READ a_comando      ;leer comando
M_COMPARAR_COMANDO a_comando;comparar comando

mov v_mov[0], cl      ;Se guarda valor de comando
;M_PRINT v_mov
;Ejecutar comandos
xor al, al            ;Limpiar registro
mov al, v_mov[0]
cmp al, 190           ;Error
je et_error_b
cmp al, 33             ;Comando PASS
je et_comando_n_pass
cmp al, 34             ;Comando EXIT
je et_comando_exit
cmp al, 35             ;Comando SAVE
je et_comando_save
cmp al, 36             ;Comando SHOW
je et_comando_show

```

Comando SAVE (guardar)

```

et_comando_save:
    ;Guardar juego
    M_PRINT txt_rutaIn
    M_OBTENER_RUTA v_f_in
    M_PRINT txt_salto
    M_PRINT ASCII 124
    M_PRINT_ASCII 32
    M_PRINT_ASCII 45
    M_PRINT_ASCII 32
    M_PRINT v_f_in
    ;Crear archivo
    M_CREAR_ARCHIVO v_f_in
    mov v_handler, ax
    ;Escribir archivo
    M_ESCRIBIR_ARCHIVO a_tablero, v_handler, SIZEOF a_tablero -1
    ;Cerrar archivo
    M_CERRAR_ARCHIVO v_handler
    jmp et_inicio_turnos

```

Comando SHOW (mostrar reporte)

```

et_comando_show:
    ;Generar reporte
    M_PRINT txt_show1
    ;Abrir archivo
    M_CREAR_ARCHIVO v_f_html
    mov v_handler, ax
    ;Obtener fecha
    mov ah, 2ah
    int 21h
    mov v_num, dh
    M_OBTENER_NUMERO v_f_mes, v_num
    mov v_num, dl
    M_OBTENER_NUMERO v_f_dia, v_num
    ;Obtener hora
    mov ah, 2ch
    int 21h

```

```

mov v_num, ch
M_OBTENER_NUMERO v_f_hr, v_num
mov v_num, cl
M_OBTENER_NUMERO v_f_min, v_num
mov v_num, dh
M_OBTENER_NUMERO v_f_seg, v_num
;Colocar fecha en h1
mov al, v_f_dia[0]
mov html_h1[4], al
mov al, v_f_dia[1]
mov html_h1[5], al
mov al, v_f_mes[0]
mov html_h1[7], al
mov al, v_f_mes[1]
mov html_h1[8], al
mov al, v_f_hr[0]
mov html_h1[15], al
mov al, v_f_hr[1]
mov html_h1[16], al
mov al, v_f_min[0]
mov html_h1[18], al
mov al, v_f_min[1]
mov html_h1[19], al
mov al, v_f_seg[0]
mov html_h1[21], al
mov al, v_f_seg[1]
mov html_h1[22], al
;Escribir HTML
M_ESCRIBIR_ARCHIVO html_1, v_handler, SIZEOF html_1
M_ESCRIBIR_ARCHIVO html_2, v_handler, SIZEOF html_2
M_ESCRIBIR_ARCHIVO html_3, v_handler, SIZEOF html_3
M_ESCRIBIR_ARCHIVO html_4, v_handler, SIZEOF html_4
M_ESCRIBIR_ARCHIVO html_5, v_handler, SIZEOF html_5
M_ESCRIBIR_ARCHIVO html_6, v_handler, SIZEOF html_6
M_ESCRIBIR_ARCHIVO html_7, v_handler, SIZEOF html_7
M_ESCRIBIR_ARCHIVO html_8, v_handler, SIZEOF html_8
M_ESCRIBIR_ARCHIVO html_9, v_handler, SIZEOF html_9
M_ESCRIBIR_ARCHIVO html_h1, v_handler, SIZEOF html_h1 - 1
M_ESCRIBIR_ARCHIVO html_10, v_handler, SIZEOF html_10
M_ESCRIBIR_ARCHIVO html_tr_in, v_handler, SIZEOF html_tr_in
;Inicio del ciclo
mov v_tmp, 64 ;Limite de lectura a_matriz
mov v_tmp2, 0 ;Indice de a_matriz
mov v_tmp3, 50 ;Numeros
mov v_tmp4, 0 ;Indice de posición
mov cx, 64
mov bx, 0
repl_inicio:
    mov cx, v_tmp
    mov bx, v_tmp2
    mov al, a_matriz[bx]
    mov v_tmp4, ax
    ;Comparar
    mov si, v_tmp4
    mov al, a_tablero[si]
    cmp al, 78
    je repl_ficha_negra
    cmp al, 66

```

```

        je repl_ficha_blanca
        jne repl_ficha_null
    repl_seguir:
        mov ax, v_tmp4
        cmp al, 54
        je repl_tr_fin
        cmp al, 73
        je repl_tr_fin
        cmp al, 92
        je repl_tr_fin
        cmp al, 111
        je repl_tr_fin
        cmp al, 130
        je repl_tr_fin
        cmp al, 149
        je repl_tr_fin
        cmp al, 168
        je repl_tr_fin
        jmp repl_loop
    repl_ficha_negra:
        M_ESCRIBIR_ARCHIVO html_div_n, v_handler, SIZEOF html_div_n
        jmp repl_seguir
    repl_ficha_blanca:
        M_ESCRIBIR_ARCHIVO html_div_b, v_handler, SIZEOF html_div_b
        jmp repl_seguir
    repl_ficha_null:
        M_ESCRIBIR_ARCHIVO html_div, v_handler, SIZEOF html_div
        jmp repl_seguir
    repl_tr_fin:
        mov ax, v_tmp3
        mov html_tr_in[8], al
        M_ESCRIBIR_ARCHIVO html_tr_out, v_handler, SIZEOF html_tr_out
        M_ESCRIBIR_ARCHIVO html_tr_in, v_handler, SIZEOF html_tr_in
        add v_tmp3, 1
        jmp repl_loop
    repl_loop:
        add v_tmp2, 1
        sub v_tmp, 1
        cmp v_tmp, 0
        je repl_fin
        jmp repl_inicio
    repl_fin:
        ;Terminar con el ciclo
        M_ESCRIBIR_ARCHIVO html_tr_out, v_handler, SIZEOF html_tr_out
        M_ESCRIBIR_ARCHIVO html_11, v_handler, SIZEOF html_11 - 1
        M_PRINT txt_show2
        M_CERRAR_ARCHIVO v_handler
        jmp et_inicio_turnos

```

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
```

Llenar tablero y validar posición

```
M_Llenar_tablero macro buffer, coordenada, ficha, var
    local et_colocar_ficha, et_error_ficha, et_salir
    ;buffer es la matriz que contiene los datos del juego
    ;Ejecutar coordenada
    mov ax, 0
    mov cx, 0
    mov bh, 0          ;Limpiar registro BH
```

```

mov bl, coordenada[0]    ;Asignar coordenada
mov al, buffer[bx]       ;Obtenemos valor de la casilla
cmp al, 45               ;Comparamos - = 45
je et_colocar_ficha      ;Es igual - Esta disponible
jne et_error_ficha       ;NO esta disponible
et_colocar_ficha:
    mov buffer[bx], ficha;Poner una N en coordenada
    add var, 1
    jmp et_salir
et_error_ficha:
    mov ch, 190          ;Mandar error por registro CH
et_salir:
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
```