

¡Error! Marcador no definido.FUNCION	DESCRIPCION	PAR. ENTRADA	PAR. SALIDA	NOMBRE MACRO	NOTAS
<b>LECTURA DE CARACTER</b>					
<b>01H</b>	Lee un carácter del dispositivo de entrada estándar (con eco).	AH = 1	AL = carácter	READ_KBD_AND_ECHO	
<b>03H</b>	Lee un carácter del dispositivo auxiliar (COM1).	AH = 3	AL = carácter	AUX_INPUT	
<b>07H</b>	Lee un carácter del dispositivo de entrada estándar (sin eco).	AH = 7	AL = carácter	DIR_CONSOLE_INPUT	
<b>08H</b>	Lee un carácter del teclado (sin eco)	AH = 8	AL = carácter	READ_KBD	
<b>ESCRITURA DE CARACTER</b>					
<b>02H</b>	Envía un carácter a la salida estándar	AH = 2 // DL = carácter	Ninguno	DISPLAY_CHAR CHARACTER	
<b>04H</b>	Envía un carácter al dispositivo auxiliar	AH = 4 // DL = carácter	Ninguno	AUX_OUTPUT CHARACTER	
<b>05H</b>	Envía un carácter a la impresora estándar	AH = 5 // DL = carácter	Ninguno	PRINT_CHAR CHARACTER	
<b>OTRAS DE PANTALLA/TECLADO</b>					
<b>06H</b>	Si DL = 0FFH, devuelve el código ASCII de la tecla pulsada y Fz = 0. Si no hay tecla pulsada Fz = 1. Si DL <> 0FFH, éste se envía a la salida estándar	AH = 6 ¡Error! Marcador no definido. DL -> (ver descripción)	Si DL = 0FFH antes de la llamada: Fz = 0 indica que AL tiene un carácter del teclado. Si DL <> 0FFH no hay parámetro de salida	DIR_CONSOLE_IO SWITCH	
<b>09H</b>	Envía un string terminado con '\$' a la consola	AH = 9 DS:DX = puntero a string	Ninguno	DISP_STRING STRING	
<b>0AH</b>	Lee un string del teclado	AH = 0AH DS:DX = puntero a buffer	Ninguno	GET_STRING LIMIT,STRING	(1)
<b>0BH</b>	Comprueba si hay caracteres disponibles del teclado	AH = 0BH	AL = 0FFH -> hay caracteres AL = 0 -> No hay caracteres	CHECK_KBD_STATUS	
<b>0CH</b>	Borra el buffer de teclado. Si AL contiene los valores 1,6,7,8 ó 0Ah ejecuta la system call respectiva.	AH = 0CH AL = 1,6,7,8 ò = 0AH si AL <> 1,6,7,8,0AH solo borra buffer	AL = 0 (solo se vació el buffer) Si se ejecutó la s.c. 1,6,7,8 ó 0AH, los propios de la función respectiva.	FLUSH_AND_READ_KBD SWITCH	

¡Error! Marcador no definido.FUNCION	DESCRIPCION	PAR. ENTRADA	PAR. SALIDA	NOMBRE MACRO	NOTAS
<b>MANEJO DE DISCOS</b>					
<b>0EH</b>	Selecciona la unidad de disco actual	AH = 0EH DL = código unidad (0=A, 1=B, etc.)	AI = nº de unidades lógicas	SELECT_DISK DISK	
<b>19H</b>	Obtiene el código de la unidad actual	AH = 19H	AL -> disco actual (0=A, 1=B, etc.)	CURRENT_DISK	
<b>1AH</b>	Pone la dirección del DTA	AH = 1AH DS:DX = dirección del DTA	Ninguno	SET_DTA BUFFER	
<b>2FH</b>	Obtiene la dirección del DTA	AH = 2FH	ES:BX = puntero al DTA	GET_DTA	
<b>36H</b>	Devuelve el nº de clusters disponibles	AH = 36H DL = unidad de disco (0=defecto,1=A,2=B...)	AX = 0FFFFH si nº de disco inválido. En caso contrario indica nº de sectores por cluster. BX = clusters disponibles CX = bytes/sector DX = nº de clusters totales	GET_DISK_SPACE DRIVE	
<b>MANEJO DE DIRECTORIOS</b>					
<b>39H</b>	Crea un directorio	AH = 39H DS:DX = puntero a pathname	FC = 0 -> no hubo error FC = 1 -> se produjo error	MAKE_DIR PATH	(2)
<b>3AH</b>	Borra un directorio	AH = 3AH DS:DX = puntero a pathname	FC = 0 -> no hubo error FC = 1 -> se produjo error	REM_DIR PATH	(2)
<b>3BH</b>	Cambia de directorio	AH = 3BH DS:DX = puntero a pathname	FC = 0 -> no hubo error FC = 1 -> se produjo error	CHANGE_DIR PATH	(2)
<b>47H</b>	Devuelve el pathname absoluto del directorio en curso de un disco	AH = 47H DS:SI -> puntero a buffer de 64 Bytes DL = unidad de disco (0=defecto,1=A,2=B,...)	Fc = 0 no hubo error Fc = 1 hubo error; Si AX = 15 código de unidad de disco incorrecto	GET_DIR DRIVE,BUFFER	

;Error! Marcador no definido.FUNCION	DESCRIPCION	PAR. ENTRADA	PAR. SALIDA	NOMBRE MACRO	NOTAS
<b>MANEJO DE FICHEROS</b>					
<b>3CH</b>	Crea un fichero y le asigna el primer <b>handle</b> disponible	AH = 3CH DS:DX = puntero a string donde reside el pathname. CX = atributo	Fc = 1 -> error: AX=3 pathname inválido,AX=4 Demasiados ficheros abiertos,AX=5 Acceso denegado. Fc = 0 -> AX = nº de <b>handle</b>	CREATE_HANDLE PATH,ATTRIB	(2)
<b>41H</b>	Borra un fichero	AH = 41H DS:DX = puntero a string con pathname	Fc = 1 -> error. AX = 2 no se encuentra fichero. AX = 5 acceso denegado Fc = 0 -> no hubo error	DELETE_ENTRY PATH	
<b>43H</b>	Lee o pone los atributos de un fichero	AH = 43H // AL = 0 leer atributos // AL = 1 poner atributos. CX <- atributos (si AL = 1) DS:DX = puntero a string	Si Fc = 1 -> error: AX=1 función inválida // AX=2 fichero no encontrado // AX=3 path inválido // AX=5 acceso denegado. Si Fc = 0 -> CL = atributos	CHANGE_ATTR PATH,ACTION,ATTRIB	(2)
<b>4EH</b>	Busca la primera entrada de fichero que coincida con las especificaciones dadas	AH = 4EH DS:DX = puntero a string ASCIIZ con pathname (se admiten metacaracteres) CX = atributos	Si Fc = 1 -> error: AX=2 pathname inválido // AX = 12H No se encontró el fichero coincidente. Si Fc = 0 -> fichero encontrado (datos en el DTA).	FIND_FIRST_FILE PATH,ATTRIB	(3)
<b>4FH</b>	Busca las siguientes entradas si la función anterior (4EH) encontró una entrada coincidente (Fc = 0).	AH = 4FH	Si Fc = 1 y AX = 12H no hay más ficheros coincidentes con el patrón especificado en la función anterior Si Fc = 0 -> ficheros encontrado (datos en el DTA).	FIND_NEXT_FILE	(3)
<b>56H</b>	Renombra un fichero	AH = 56H DS:DX = puntero a pathname original ES:DI = puntero a pathname nuevo	FC = 0 -> no hubo error FC = 1 -> se produjo error // AX=2 -> fichero no encontrado // AX=5 -> Acceso denegado // AX = 17 -> los dos pathname son de otro drive	RENAME_FILE OLD_PATH,NEW_PATH	

¡Error! Marcador no definido.FUNCION	DESCRIPCION	PAR. ENTRADA	PAR. SALIDA	NOMBRE MACRO	NOTAS
<b>MANEJO DE CONTENIDO DE FICHEROS</b>					
<b>3DH</b>	Abre un fichero o dispositivo existente	AH = 3DH DS:DX = puntero a pathname. AL = byte de acceso	Fc = 0 -> no hubo error: AX = <b>handle</b> Fc = 1 -> error: AX = 1 función inválida // AX=2 fichero no encontrado //AX=3 pathname inválido //AX=4 Demasiados ficheros abiertos // AX=5 Acceso denegado // AX=12 acceso inválido.	OPEN_HANDLE PATH,ACCESS	(4)
<b>3EH</b>	Cierra un fichero que se abrió con las funciones 3DH o 3CH.	AH = 3EH BX = <b>handle</b>	Si Fc = 1 -> error: AX = 6 si handle inválido Si Fc = 0 -> func. ejecutada normalmente.	CLOSE_HANDLE HANDLE	
<b>3FH</b>	Lee de un fichero o dispositivo un nº de bytes determinado hacia un buffer	AH = 3FH BX = <b>handle</b> CX = nº de bytes DS:DX = puntero a buffer	Si Fc = 1 -> error: AX=5 acceso denegado // Ax=6 handle inválido. Si Fc = 0 -> AX = nº de bytes leídos	READ_HANDLE HANDLE,BUFFER,BYTES	(5)
<b>40H</b>	Escribe a un fichero o dispositivo un nº de bytes determinado desde un buffer	AH = 40H BX = <b>handle</b> CX = nº de bytes DS:DX = puntero a buffer	Si Fc = 1 -> error: AX=5 acceso denegado // Ax=6 handle inválido. Si Fc = 0 -> AX = nº de bytes escritos	WRITE_HANDLE HANDLE,DATA,BYTES	(5)
<b>42H</b>	Mueve la posición del puntero asociado al <b>handle</b> de un fichero para indicar la posición a la que accederá la siguiente operación	AH = 42H AL = indicador de posición: 0 -> BOF 1 -> POSICION ACTUAL 2 -> EOF BX = <b>handle</b> CX:DX = offset (ver nota 6)	Si Fc = 0: DX:AX = Ubicación del puntero (contando desde el inicio) Si Fc = 1 -> error: AX = 1 -> función inválida AX = 6 -> handle inválido	MOVE_PTR HANDLE,HIGH,LOW,METHOD	(6)
<b>MANEJO DE FECHA Y HORA DEL SISTEMA</b>					
<b>2AH</b>	Devuelve la fecha del sistema	AH = 2AH	CX = Año (de 1980 a 2099) DH = mes (1-12) // DL = Día(1-31) // AL = Día de la semana (0 = Domingo)	GET_DATE	
<b>2BH</b>	Pone la fecha del sistema	AH = 2BH CX = Año DH = Mes DL = Día	AL = 0FFH -> Error, fecha inválida AL = 00 -> No hubo error	SET_DATE YEAR,MONTH,DAY	

¡Error! Marcador no definido.FUNCION	DESCRIPCION	PAR. ENTRADA	PAR. SALIDA	NOMBRE MACRO	NOTAS
<b>2CH</b>	Devuelve la hora del sistema	AH = 2CH	CH = Hora CL = Minutos (0-59) DH = Segundos (0-59) DL = Centésimas (0-99)	GET_TIME	
<b>2DH</b>	Pone la hora del sistema	AH = 2DH CH = Hora (0-23) CL = Minutos (0-59) DH = Segundos (0-59) DL = Centésimas (0-99)	AL = 0FFH -> Error, hora no válida AL = 00 -> No hubo error	SET_TIME HOUR,MINUTES,SECON DS,HUNDRETHS	
<b>MANEJO DINAMICO DE MEMORIA</b>					
<b>48H</b>	Pide memoria al sistema para asignarla al proceso en curso.	AH = 48H BX = nº de párrafos a pedir	Si Fc = 0: AX contiene la dirección del segmento del bloque asignado. Si Fc = 1 -> Error: AX = 7 bloque de control de memoria mal. // AX = 8 memoria insuficiente (BX indica el nº máximo de párrafos disponibles.	ALLOCATE_MEMORY BYTES	
<b>49H</b>	Libera la memoria obtenida con la función 48H	AH = 49H ES = Dirección inicial del segmento	Si Fc = 0 -> No hubo error Si Fc = 1 -> Error: AX=7 Bloque malo // AX=9 Valor en ES inválido	FREE_MEMORY SEG_ADDR	
<b>4AH</b>	Cambia el tamaño de un bloque de memoria	AH = 4AH ES = dirección del segmento del bloque a cambiar BX = Nuevo tamaño del bloque en párrafos	Si Fc = 0 -> No hubo error Si Fc = 1 -> Error: AX = 7 Bloque mal // AX = 9 Valor en ES inválido // AX = 8 No hay suficiente memoria	LIB_MEM_COM LAST_BYTE,STACK_SIZE Y LIB_MEM_EXE LAST_SEGM	(7)

¡Error! Marcador no definido.FUNCION	DESCRIPCION	PAR. ENTRADA	PAR. SALIDA	NOMBRE MACRO	NOTAS
<b>MANEJO DE VECTORES DE INTERRUPCION</b>					
<b>35H</b>	Obtiene la dirección de la rutina de servicio de una interrupción	AH = 35H AL = nº de interrupción	ES:BX = Dirección Rutina	GET_VECTOR INTERRUPT	
<b>25H</b>	Modifica la dirección de la rutina de servicio de una interrupción	AH = 25H AL = nº interrupción DS:DX = Dirección	Ninguno	CAPTURE_INT INTERRUPT,HANDLER_OFF, HANDLER_SEG  Y  RESTORE_INT	
<b>SALIDA DE PROGRAMA</b>					
<b>4CH</b>	Termina el programa y devuelve control al sistema operativo.	AH = 4CH AL = código de retorno	Ninguno	END_PROCESS RETURN_CODE	
<b>31H</b>	Idéntica a la 4CH excepto que deja al programa residente (No libera memoria).	AH = 31H AL = código de retorno DX = N° de párrafos de memoria que deben quedar sin liberar	Ninguno	KEEP_PROCESS_COM RETURN_CODE, LAST_BYTE  Y  KEEP_PROCESS_EXE RETURN_CODE, LAST_BYTE, FIRST_SEGM	(8)

**NOTAS:**

1.- La composición del buffer para la función 0AH debe tener la siguiente estructura:

**BYTE 1:** N° máximo de caracteres en el buffer incluyendo el retorno de carro

**BYTE 2:** N° real de caracteres tecleados sin contar el retorno de carro, este valor lo pone la función pero hay que dejar reservado el byte.

**BYTE 3 a N:** Es el buffer donde se depositarán los caracteres tecleados, debe tener al menos la longitud especificada en el byte 1.

2.- El pathname debe ser un string de caracteres terminado por un NULL y debe especificar el camino absoluto o relativo.

Atributos de fichero:

<b>00</b>	<b>Normal</b>
<b>01</b>	<b>Solo Lectura</b>
<b>02</b>	<b>Oculto</b>
<b>04</b>	<b>Fichero Sistema</b>

3.- Estas funciones (4EH y 4FH) devuelven las características del fichero encontrado en los 43 primeros bytes del DTA, que se supone está en un buffer declarado con anterioridad con la función 1AH.

La estructura del DTA es la siguiente:

**Bytes 0 a 14h:** Reservados

**Byte 15h:** Atributo

**Bytes 16h a 19h:** Hora y fecha última escritura (\*)

**Bytes 1Ah a 1Dh:** Tamaño en bytes

**Bytes 1Eh a 2Ah:** Nombre y extensión del fichero ajustado a la izda. y terminado en NULL. Si tiene extensión hay un punto entre el nombre y la extensión. No hay caracteres blancos intermedios.

(\*) **Formato de Hora y Fecha:**

Offsets 16H y 17H (Hora), forman una Word (offset 17H parte alta de la Word):

bits 0 a 4 → Segundos

bits 5 a 10 → Minutos

bits 11 a 15 → Horas

Offsets 18H y 19H (Fecha), forman una Word (offset 19H parte alta de la Word):

bits 0 a 4 → Día

bits 5 a 8 → Mes

bits 9 a 15 → Año

4.- Byte de acceso:

0 = Lectura

1 = Escritura

2 = Lectura / escritura

5.- Si Fc = 0 a la salida de la función, AX indica el número de bytes leídos o escritos. Si AX es menor que el número de bytes pedidos, es porque se ha producido una lectura o escritura parcial causada probablemente porque se alcanzó el EOF en lectura o porque el disco está lleno en escritura.

- 6.- A la función se le pasa un parámetro de 32 bits (entero con signo) que se interpreta como un desplazamiento relativo a una posición inicial que se especifica en AL (BOF,EOF o posición actual). Si el desplazamiento es negativo, el puntero se mueve hacia el comienzo del fichero. Si es positivo se mueve hacia el final del fichero. Si se intenta mover el puntero por debajo del BOF o por encima del EOF no da error. El puntero se queda en BOF o EOF (Se puede comprobar por el valor en DX:AX).
- 7.- Para reducir la cantidad de memoria inicialmente asignada a un programa COM utilizar la macro LIB\_MEM\_COM, y para un programa EXE utilizar la macro LIB\_MEM\_EXE.
- 8.- El valor en DX debe ser el tamaño total a permanecer en memoria (no solo el tamaño del segmento de código). Por ejemplo, no se debe olvidar incluir el tamaño del PSP.  
KEEP\_PROCESS\_COM se sale dejando residente un programa COM.  
KEEP\_PROCESS\_EXE se sale dejando residente un programa EXE.

**CONTENIDO DEL FICHERO MACROS.MAC*****LECTURA DE CARACTER***

<b>READ_KBD_AND_ECHO</b>	<b>MACRO</b>	
	MOV	AH,01
	INT	21H
	<b>ENDM</b>	
<b>AUX_INPUT</b>	<b>MACRO</b>	
	MOV	AH,03
	INT	21H
	<b>ENDM</b>	
<b>DIR_CONSOLE_INPUT</b>	<b>MACRO</b>	
	MOV	AH,7
	INT	21H
	<b>ENDM</b>	
<b>READ_KBD</b>	<b>MACRO</b>	
	MOV	AH,8
	INT	21H
	<b>ENDM</b>	

***ESCRITURA DE CARACTER*****DISPLAY\_CHAR****MACRO**  
MOV DL,CARACTER  
MOV AH,02  
INT 21H  
**ENDM****AUX\_OUTPUT****MACRO**  
MOV DL,CARACTER  
MOV AH,04  
INT 21H  
**ENDM****PRINT\_CHAR****MACRO**  
MOV DL,CARACTER  
MOV AH,05  
INT 21H  
**ENDM**

### ***OTRAS DE PANTALLA/TECLADO***

<b>DIR_CONSOLE_IO</b>	<b>MACRO</b>	<b>SWITCH</b>
	MOV	DL,SWITCH
	MOV	AH,06
	INT	21H
	<b>ENDM</b>	

<b>DISP_STRING</b>	<b>MACRO</b>	<b>STRING</b>
	LEA	DX,STRING
	MOV	AH,9
	INT	21H
	<b>ENDM</b>	

; Macro DISP\_STRINGZ, visualiza un string ASCIIZ. Puede contener el carácter "\$". Modifica AX,BX,DL y Fc

<b>DISP_STRINGZ</b>	<b>MACRO</b>	<b>STRINGZ</b>
LOCAL	BUCLE,FIN	
	LEA	BX,STRINGZ
BUCLE:	MOV	DL,[BX]
	TEST	DL,DL
	JZ	FIN
	MOV	AH,2
	INT	21H
	INC	BX
	JMP	SHORT BUCLE
FIN:	<b>ENDM</b>	

<b>GET_STRING</b>	<b>MACRO</b>	<b>LIMIT,STRING</b>
	LEA	DX,STRING
	MOV	STRING, LIMIT
	MOV	AH,0AH
	INT	21H
	<b>ENDM</b>	

<b>CHECK_KBD_STATUS</b>	<b>MACRO</b>	
	MOV	AH,0BH
	INT	21H
	<b>ENDM</b>	

<b>FLUSH_AND_READ_KBD</b>	<b>MACRO</b>	<b>SWITCH</b>
	MOV	AL,SWITCH
	MOV	AH,0CH
	INT	21H
	<b>ENDM</b>	

***MANEJO DE DISCOS***

<b>SELECT_DISK</b>	<b>MACRO</b>	<b>DISK</b>
	MOV	DL,DISK - 'A'
	MOV	AH,0EH
	INT	21H
	<b>ENDM</b>	
<b>CURRENT_DISK</b>	<b>MACRO</b>	
	MOV	AH,19H
	INT	21H
	<b>ENDM</b>	
<b>SET_DTA</b>	<b>MACRO</b>	<b>BUFFER</b>
	LEA	DX,BUFFER
	MOV	AH,1AH
	INT	21H
	<b>ENDM</b>	
<b>GET_DTA</b>	<b>MACRO</b>	
	MOV	AH,2FH
	INT	21H
	<b>ENDM</b>	
<b>GET_DISK_SPACE</b>	<b>MACRO</b>	<b>DRIVE</b>
	MOV	DL,DRIVE
	MOV	AH,36H
	INT	21H
	<b>ENDM</b>	

**MANEJO DE DIRECTORIOS**

<b>MAKE_DIR</b>	<b>MACRO</b>	<b>PATH</b>
	LEA	DX, PATH
	MOV	AH,39H
	INT	21H
	<b>ENDM</b>	
<b>REM_DIR</b>	<b>MACRO</b>	<b>PATH</b>
	LEA	DX,PATH
	MOV	AH,3AH
	INT	21H
	<b>ENDM</b>	
<b>CHANGE_DIR</b>	<b>MACRO</b>	<b>PATH</b>
	LEA	DX,PATH
	MOV	AH,3BH
	INT	21H
	<b>ENDM</b>	
<b>GET_DIR</b>	<b>MACRO</b>	<b>DRIVE,BUFFER</b>
	MOV	DL,DRIVE
	LEA	SI,BUFFER
	MOV	AH,47H
	INT	21H
	<b>ENDM</b>	

### *MANEJO DE FICHEROS*

<b>CREATE_HANDLE</b>	<b>MACRO</b>	<b>PATH,ATTRIB</b>
	LEA	DX,PATH
	MOV	CX,ATTRIB
	MOV	AH,3CH
	INT	21H
	<b>ENDM</b>	
<b>DELETE_ENTRY</b>	<b>MACRO</b>	<b>PATH</b>
	LEA	DX,PATH
	MOV	AH,41H
	INT	21H
	<b>ENDM</b>	
<b>CHANGE_ATTR</b>	<b>MACRO</b>	<b>PATH,ACTION,ATTRIB</b>
	LEA	DX,PATH
	MOV	AL,ACTION
	MOV	CX,ATTRIB
	MOV	AH,43H
	INT	21H
	<b>ENDM</b>	
<b>FIND_FIRST_FILE</b>	<b>MACRO</b>	<b>PATH, ATRIB</b>
	LEA	DX,PATH
	MOV	CX,ATRIB
	MOV	AH,4EH
	INT	21H
	<b>ENDM</b>	
<b>FIND_NEXT_FILE</b>	<b>MACRO</b>	
	MOV	AH,4FH
	INT	21H
	<b>ENDM</b>	
<b>RENAME_FILE</b>	<b>MACRO</b>	<b>OLD_PATH,NEW_PATH</b>
	LEA	DX,OLD_PATH
	PUSH	DS
	POP	ES
	LEA	DI,NEW_PATH
	MOV	AH,56H
	INT	21H
	<b>ENDM</b>	

**MANEJO DE CONTENIDO DE FICHEROS**

<b>OPEN_HANDLE</b>	<b>MACRO</b>	<b>PATH,ACCESS</b>
	LEA	DX, PATH
	MOV	AL, ACCESS
	MOV	AH,3DH
	INT	21H
	<b>ENDM</b>	
<b>CLOSE_HANDLE</b>	<b>MACRO</b>	<b>HANDLE</b>
	MOV	BX,HANDLE
	MOV	AH,3EH
	INT	21H
	<b>ENDM</b>	
<b>READ_HANDLE</b>	<b>MACRO</b>	<b>HANDLE,BUFFER,BYTES</b>
	MOV	BX,HANDLE
	LEA	DX, BUFFER
	MOV	CX,BYTES
	MOV	AH,3FH
	INT	21H
	<b>ENDM</b>	
<b>WRITE_HANDLE</b>	<b>MACRO</b>	<b>HANDLE,DATA,BYTES</b>
	MOV	BX,HANDLE
	LEA	DX, DATA
	MOV	CX,BYTES
	MOV	AH,40H
	INT	21H
	<b>ENDM</b>	
<b>MOVE_PTR</b>	<b>MACRO</b>	<b>HANDLE,HIGH,LOW,METHOD</b>
	MOV	BX,HANDLE
	MOV	CX,HIGH
	MOV	DX,LOW
	MOV	AL,METHOD
	MOV	AH,42H
	INT	21H
	<b>ENDM</b>	

***MANEJO DE FECHA Y HORA DEL SISTEMA***

<b>GET_DATE</b>	<b>MACRO</b>	
	MOV	AH,2AH
	INT	21H
	<b>ENDM</b>	
<b>SET_DATE</b>	<b>MACRO</b>	<b>YEAR,MONTH,DAY</b>
	MOV	CX,YEAR
	MOV	DH,MONTH
	MOV	DL,DAY
	MOV	AH,2BH
	INT	21H
	<b>ENDM</b>	
<b>GET_TIME</b>	<b>MACRO</b>	
	MOV	AH,2CH
	INT	21H
	<b>ENDM</b>	
<b>SET_TIME</b>	<b>MACRO</b>	<b>HOUR,MINUTES,SECONDS,HUNDRETHS</b>
	MOV	CH,HOUR
	MOV	CL,MINUTES
	MOV	DH,SECONDS
	MOV	DL,HUNDRETHS
	MOV	AH,2DH
	INT	21H
	<b>ENDM</b>	

## *MANEJO DE MEMORIA DINAMICA*

```

ALLOCATE_MEMORY  MACRO BYTES
                    MOV      BX,BYTES
                    MOV      CL,4
                    SHR      BX,CL
                    INC      BX
                    MOV      AH,48H
                    INT      21H
                    ENDM

FREE_MEMORY      MACRO      SEG_ADDR
                    MOV      AX,SEG_ADDR
                    MOV      ES,AX
                    MOV      AH,49H
                    INT      21H
                    ENDM

LIB_MEM_COM      MACRO      LAST_BYTE,STACK_SIZE
                    LOCAL    FIN
                    MOV      AX,8          ;Código de retorno mem. insuficiente
                    LEA     BX,LAST_BYTE-1 ;Offset último byte del prog.
                    ADD     BX,STACK_SIZE
                    JC      FIN          ;SI Carry sale por error FC=1,AX=8
                    ADD     BX,2        ;Espacio para meter word a ceros
                    JC      FIN          ;Si carry, sale por error
                    MOV     CL,4
                    SHR     BX,CL
                    INC     BX          ;Nº párrafos necesarios por exceso
                    MOV     AH,4AH      ;Modifica tamaño bloque
                    INT     21H
                    JC      FIN          ;Si error,sale con Fc=1
                    SHL     BX,CL      ;Multiplica párrafos * 16 y
                    MOV     SP,BX      ;pone al final el SP
                    XOR     AX,AX      ;Coloca una word a ceros en pila
                    PUSH    AX
                    FIN:
                    ENDM

LIB_MEM_EXE      MACRO LAST_SEGM
                    LOCAL    SIGUE
                    MOV      BX,SP
                    DEC     BX          ;Ultimo offset de la pila
                    MOV     CL,4
                    SHR     BX,CL      ;Calcula nº de párrafos por exceso
                    INC     BX
                    MOV     AX,SS
                    ADD     AX,BX      ;AX=SS+(SP/16) por exceso
                    MOV     BX,LAST_SEGM ;BX = Last Segment
                    CMP     AX,BX      ;Compara ambos valores
                    JBE     SIGUE      ;y coge el valor mayor en BX
                    MOV     BX,AX
                    SIGUE: MOV     AX,ES ;Le resta el segmento del PSP
                    SUB     BX,AX
                    MOV     AH,4AH
                    INT     21H
                    ENDM

```

## *MANEJO DE VECTORES DE INTERRUPCIÓN*

<b>GET_VECTOR</b>	<b>MACRO</b>	<b>INTERRUPT</b>
	MOV	AL,INTERRUPT
	MOV	AH,35H
	INT	21H
	<b>ENDM</b>	
<b>CAPTURE_INT</b>	<b>MACRO</b>	<b>INTERRUPT,HANDLER_OFF,HANDLER_SEG</b>
	PUSH	DS
	MOV	AX,HANDLER_SEG
	MOV	DS,AX
	MOV	AL,INTERRUPT
	LEA	DX,HANDLER_OFF
	MOV	AH,25H
	INT	21H
	POP	DS
	<b>ENDM</b>	
<b>RESTORE_INT</b>	<b>MACRO</b>	<b>INTERRUPT,FAR_OLD_ADDRESS</b>
	PUSH	DS
	LDS	DX,FAR_OLD_ADDRESS
	MOV	AL,INTERRUPT
	MOV	AH,25H
	INT	21H
	POP	DS
	<b>ENDM</b>	

***FINALIZACION DE PROGRAMA***

<b>END_PROCESS</b>	<b>MACRO</b>	<b>RETURN_CODE</b>
	MOV	AL,RETURN_CODE
	MOV	AH,4CH
	INT	21H
	<b>ENDM</b>	
<b>KEEP_PROCESS_COM</b>	<b>MACRO</b>	<b>RETURN_CODE, LAST_BYTE</b>
	MOV	AL,RETURN_CODE
	LEA	DX, LAST_BYTE - 1
	MOV	CL,4
	SHR	DX,CL
	INC	DX
	MOV	AH,31H
	INT	21H
	<b>ENDM</b>	
<b>KEEP_PROCESS_EXE</b>	<b>MACRO</b>	<b>RETURN_CODE, LAST_BYTE, FIRST_SEGM</b>
	LOCAL	FIN
	LEA	DX, LAST_BYTE - 1
	MOV	CL,4
	SHR	DX,CL
	ADD	DX,11H
	ADD	DX,SEG LAST_BYTE
	JC	FIN
	MOV	AX, FIRST_SEGM
	SUB	DX, AX
	JC	FIN
	MOV	AL, RETURN_CODE
	MOV	AH, 31H
	INT	21H
<b>FIN:</b>	<b>ENDM</b>	

## ***COPROCESADOR***

; Esta macro transfiere el byte más significativo de la palabra de estado del coprocesador, al byte menos significativo de PSW.  
; Requiere como parámetro un nombre de variable tipo word que se usa como almacenamiento intermedio. Solo modifica AX y PSW<sub>L</sub>.

```
MOVE_STSWH_TO_PSWL    MACRO      VAR_W  
                        FSTSW      VAR_W  
                        MOV        AX,VAR_W  
                        SAHF  
                        ENDM
```

## *RUTINAS DE LIBRERÍA*

### **UNSIGN DOUB TO ASCII**

```

; Rutina de utilidad general para convertir una doble word sin signo en un string de caracteres que
; representa el valor de la doble word en la base de numeración BASE.
;
; Parámetros de entrada:
;
; ALTO: Parte alta de la doble word
; BAJO: Parte baja de la doble word
; BASE: Base de numeración (de 2 a 16)
; MIN: Mínimo n° de caracteres en el string resultante
; OFF_TABLA: Offset de una tabla en el segmento de datos que debe contener los
; caracteres ASCII de las cifras válidas en la base de numeración
; elegida.
; OFF_STRING: Es el offset de un string en el segmento de datos donde se deposita
; el resultado. (Debe tener el tamaño suficiente)
;
; Parámetros de salida:
;
; Devuelve en AX la longitud del string resultante.
;
; Registros afectados:
; Solamente AX

```

```

ALTO      EQU [BP+14]
BAJO      EQU [BP+12]
BASE      EQU [BP+10]
MIN       EQU [BP+8]
OFF_TABLA EQU [BP+6]
OFF_STRING EQU [BP+4]

```

```

CODE          SEGMENT          PUBLIC
               ASSUME CS:CODE
UNSIGN DOUB TO ASCII PROC NEAR
               PUBLIC UNSIGN DOUB TO ASCII
               PUSH BP
               MOV BP,SP
               PUSH BX
               PUSH CX
               PUSH DX
               PUSH SI
               PUSH DI
               PUSHF
               MOV SI,ALTO
               MOV DI,BAJO
               MOV BX,BASE
               XOR CX,CX

BUCLE1:
               PUSH SI
               OR SI,DI
               POP SI
               JZ BUCLE2
               MOV AX,SI
               XOR DX,DX
               DIV BX
               MOV SI,AX
               MOV AX,DI
               DIV BX
               PUSH DX
               INC CX
               MOV DI,AX
               JMP SHORT BUCLE1

BUCLE2:
               CMP CX,MIN
               JAE SIGUE
               PUSH SI
               INC CX

```

```
JMP    SHORT BUCLE2
SIGUE:
      MOV    SI,CX
      MOV    BX,OFF_TABLA
      MOV    DI,OFF_STRING
      JCXZ   FIN
BUCLE3:
      POP    AX
      XLATB
      MOV    [DI],AL
      INC    DI
      LOOP  BUCLE3
FIN:
      MOV    BYTE PTR [DI],0
      MOV    AX,SI
      POPF
      POP    DI
      POP    SI
      POP    DX
      POP    CX
      POP    BX
      POP    BP
      RET    12
UNSIGN_DOUB_TO_ASCII ENDP
CODE                 ENDS
END
```

## ; ASCII\_TO\_UNSIGN\_DOUB

```
; Rutina de utilidad general que convierte el string ASCIIZ que está en la
; variable OFF_STRING en su valor binario.
;
; Parámetros de entrada (por pila)
;
; OFF_STRING: Dirección de un string ASCIIZ con el número a convertir
; BASE: Base de numeración (2 a 16)

; Parámetros de salida:
;
; SI&DI: valor del n° convertido
; AX: Código de retorno
; 0 = el valor es cero
; 1 = se puede expresar con un byte
; 2 = requiere una word
; 4 = requiere doble word
; -1 = el valor es mayor de doble word
; -2 = se detectó un carácter erróneo en la base elegida
```

```
OFF_STRING EQU [BP+6]
BASE EQU [BP+4]

CODE SEGMENT PUBLIC
ASSUME CS:CODE
ASCII_TO_UNSIGN_DOUB PROC NEAR
PUBLIC ASCII_TO_UNSIGN_DOUB

    PUSH BP
    MOV BP,SP
    PUSH BX
    PUSH CX
    PUSH DX
    PUSHF
    XOR SI,SI
    XOR DI,DI
    XOR CX,CX
    MOV BX,STRING

BUCLE1:
    CMP BYTE PTR [BX],20H
    JNE BUCLE2
    INC BX
    JMP SHORT BUCLE1

BUCLE2:
    MOV CL,[BX]
    INC BX
    TEST CL,CL
    JZ FIN1
    SUB CL,'0'
    CMP CL,9
    JBE SIGUE
    SUB CL,7
    AND CL,0DFH
    CMP CL,0AH
    JB NO_NUM
    CMP CL,0FH
    JA NO_NUM

SIGUE:
    CMP CL,BASE
    JAE NO_NUM
    PUSH BX
    MOV BX,BASE
    MOV AX,DI
    MUL BX
    MOV DI,AX
    MOV AX,SI
    MOV SI,DX
```

MUL BX

```
JC OVERF
ADD SI,AX
JC OVERF
ADD DI,CX
ADC SI,0
JC OVERF
POP BX
JMP SHORT BUCLE2
NO_NUM:
MOV AX,-2
JMP SHORT FIN2
OVERF:
POP BX
MOV AX,-1
JMP SHORT FIN2
FIN1:
MOV AX,4
TEST SI,SI
JNZ FIN2
SHR AX,1
CMP DI,0FFH
JA FIN2
SHR AX,1
TEST DI,DI
JNZ FIN2
XOR AX,AX
FIN2:
POPF
POP DX
POP CX
POP BX
POP BP
RET 4
ASCII_TO_UNSIGN_DOUB ENDP
CODE ENDS
END
```

## ASCII\_TO\_SIGN\_DOUB

```

; Rutina de utilidad general que convierte el string ASCIIZ que está en la
; variable OFF_STRING en su valor binario.
; El valor del string es con signo, por lo que se admite el signo '-' o '+'
; delante. Si no se pone signo, se entiende que es positivo.
; El resultado se devuelve en Ca2
;
;
; Parámetros de entrada (por pila)
;
; OFF_STRING: Dirección de un string ASCIIZ con el número a convertir
; BASE: Base de numeración (2 a 16)

; Parámetros de salida:
;
; SI&DI: valor del nº convertido
; AX: Código de retorno
;
; 0 = el valor es cero
; 1 = se puede expresar con un byte
; 2 = requiere una word
; 4 = requiere doble word
; -1 = el valor es mayor de doble word
; -2 = se detectó un carácter erróneo en la base elegida

```

```

OFF_STRING      EQU    [BP+6]
BASE            EQU    [BP+4]

CODE            SEGMENT      PUBLIC
                ASSUME CS:CODE
ASCII_TO_SIGN_DOUB  PROC  NEAR

PUBLIC          ASCII_TO_SIGN_DOUB

                PUSH  BP
                MOV   BP,SP
                PUSH  BX
                PUSH  CX
                PUSH  DX
                PUSHF
                XOR   SI,SI
                XOR   DI,DI
                XOR   CX,CX
                MOV   BX,OFF_STRING

BUCLE1:
                CMP   BYTE PTR[BX],20H
                JNE   SIGUE1
                INC   BX
                JMP   SHORT BUCLE1

SIGUE1:
                PUSH  SI
                MOV   CL,[BX]
                CMP   CL,'+'
                JE    BUCLE2
                CMP   CL,'-'
                JNE   SIGUE2
                POP   AX
                INC   AX
                PUSH  AX

BUCLE2:
                INC   BX
                MOV   CL,[BX]

SIGUE2:
                TEST  CL,CL
                JZ    FIN1
                SUB   CL,'0'
                CMP   CL,9
                JBE   SIGUE3
                SUB   CL,7
                AND   CL,0DFH

```

CMP CL,0AH

```

                JB  NO_NUM
                CMP CL,0FH
                JA  NO_NUM

SIGUE3:
                CMP CL,BASE
                JAE NO_NUM
                PUSH BX
                MOV  BX,BASE
                MOV  AX,DI
                MUL  BX
                MOV  DI,AX
                MOV  AX,SI
                MOV  SI,DX
                MUL  BX
                JC   OVERF
                ADD  SI,AX
                JC   OVERF
                ADD  DI,CX
                ADC  SI,0
                JC   OVERF
                TEST SI,8000H
                JNZ  OVERF
                POP  BX
                JMP  SHORT BUCLE2

NO_NUM:
                MOV  AX,-2
                JMP  SHORT FIN2

OVERF:
                MOV  AX,-1
                POP  BX

FIN2:
                POP  BX
                JMP  SHORT FIN3

FIN1:
                MOV  AX,4
                TEST SI,SI
                JNZ  SIGUE4
                TEST DI,8000H
                JNZ  SIGUE4
                SHR  AX,1
                TEST DI,0FF80H
                JNZ  SIGUE4
                SHR  AX,1
                TEST DI,DI
                JNZ  SIGUE4
                XOR  AX,AX

SIGUE4:
                POP  BX
                TEST BX,BX
                JZ   FIN3
                NOT  SI
                NOT  DI
                ADD  DI,1
                ADC  SI,0

FIN3:          POPF

                POP  DX
                POP  CX
                POP  BX
                POP  BP
                RET  4

ASCII_TO_SIGN_DOUB
CODE          ENDP
                ENDS

                END

```

