DEPARTAMENTO DE ELECTRONICA

Microprocesadores
1121060
Tema 5.
Programación Modular.

Microprocesadores 1121060

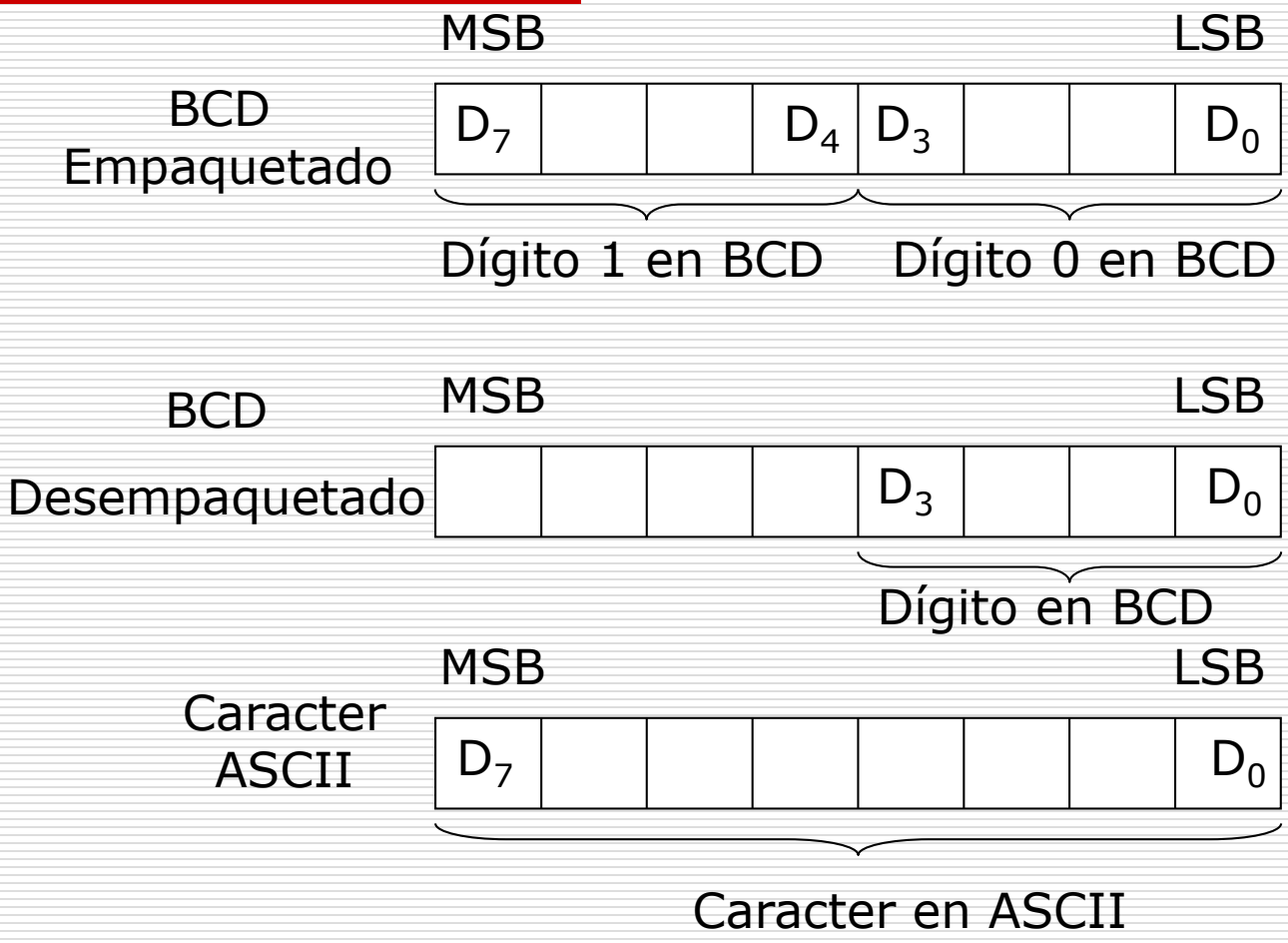
Tema 5. Programación Modular.

1. Tipos de datos en ensamblador
   - 1.1 Enteros, booleanos y caracteres
   - 1.2 Arreglos y Registros
2. Estructuras de control en ensamblador
   - 2.1 Estructuras de selección (tipo if-then-else) y de selección múltiple (tipo case)
   - 2.2 Estructuras de repetición (tipo for, while y repeat)
3. Directivas
4. Diseño de Programas
   - 3.1 Programación Modular
   - 3.2 Macros
5. Depuración

Tema 5. Programación Modular.
Tipos de datos en ensamblador

- **Enteros**
  - Sin signo
    - Tamaño byte 0 a 255
    - Tamaño palabra 0 a 65535
  - Con signo
    - Tamaño byte -128 a +127
    - Tamaño palabra -32768 a +32767
- **Números codificados en BCD (0 al 9)**
  - Almacenados en forma Empaquetada
  - Almacenados en forma Desempaquetada
- **Datos codificados en ASCII** (American Standard Code for Information Interchange)

Tema 5. Programación Modular.
Tipos de datos en ensamblador



Tema 5. Programación Modular.

Tipos de datos en ensamblador

- Caracteres ASCII

ASCII	Hex	Símbolo
0	0	NUL
1	1	SOH
2	2	STX
3	3	ETX
4	4	EOT
5	5	ENQ
6	6	ACK
7	7	BEL
8	8	BS
9	9	TAB
10	A	LF
11	B	VT
12	C	FF
13	D	CR
14	E	SO
15	F	SI

ASCII	Hex	Símbolo
16	10	DLE
17	11	DC1
18	12	DC2
19	13	DC3
20	14	DC4
21	15	NAK
22	16	SYN
23	17	ETB
24	18	CAN
25	19	EM
26	1A	SUB
27	1B	ESC
28	1C	FS
29	1D	GS
30	1E	RS
31	1F	US

ASCII	Hex	Símbolo
32	20	(espacio)
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/

ASCII	Hex	Símbolo
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

ASCII	Hex	Símbolo
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O

ASCII	Hex	Símbolo
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

ASCII	Hex	Símbolo
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o

ASCII	Hex	Símbolo
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	□

Tema 5. Programación Modular
Estructuras de control en ensamblador

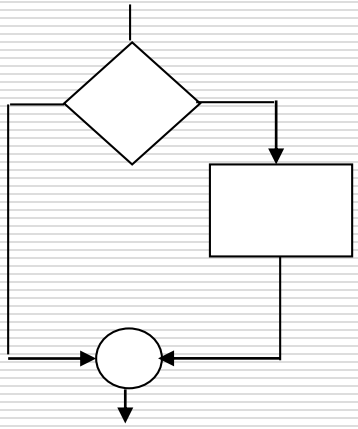
- EL término estructurado aplica a ciertas declaraciones estándar de un programa de alto nivel que pueden ser traducidas cuidadosamente a código ensamblador. Los programadores de lenguaje de alto nivel tratan de usar algunas estructuras de programación básicas para resolver la mayoría de sus problemas. Alguien que haya programado en BASIC, Pascal o C estará familiarizado con estructuras de programación estándar como las siguientes:
  - IF.... THEN
  - IF ....THEN .....ELSE
  - DO....WHILE
  - DO....UNTIL
  - CASE
  - FOR

Tema 5. Programación Modular.
Estructuras de control en ensamblador

- IF simple

Ejemplo:

```
        mov al,A
        cmp al,B
        jae salir
then:inc cx
        mov bx,cx
salir:--------
```

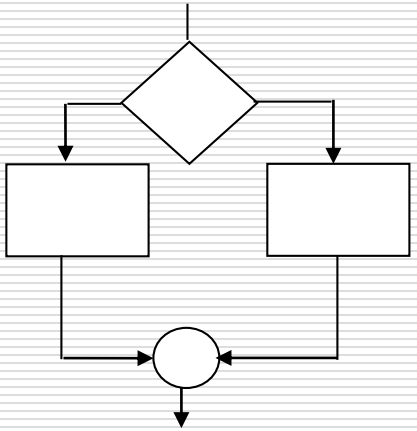


IF.... THEN

Tema 5. Programación Modular.
Estructuras de control en ensamblador

- IF ....THEN .....ELSE

```
            mov al,A
            cmp al,B
            jbe opcionA
opcionB:    dec cx
            jmp salir
opcionA:    inc cx
Salir:      ----------
```

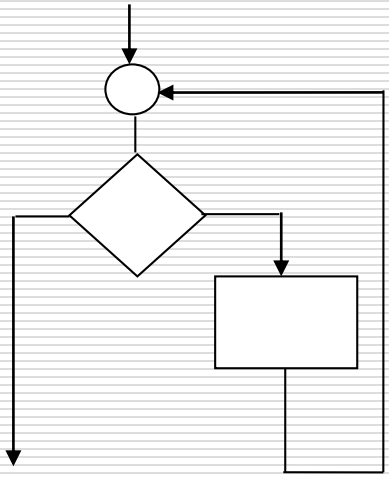


IF ....THEN .....ELSE

Tema 5. Programación Modular.
Estructuras de control en ensamblador

- DO....WHILE con prueba al inicio

```
ciclo:      mov al,A
            cmp al,B
            je salir
            dec cx
              :
            jmp ciclo
salir:      ----------
```

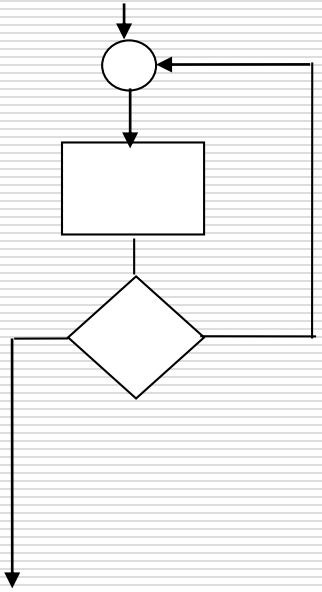


DO....WHILE con prueba al inicio

Tema 5. Programación Modular.
Estructuras de control en ensamblador

DO.....WHILE con prueba al final

```
ciclo:      dec cx
               ⋮

if:         mov al,A
            cmp al,B
            je salir
            jmp ciclo
salir:      ----------
```



DO.....WHILE con prueba al final

Tema 5. Programación Modular.
Estructuras de control en ensamblador

- **CASE**

```
            cmp AH,01
            jne dos
            ;codigo de AH=1
               ⋮
            jmp salirIF
dos:  cmp AH,02
            jne tres
            ;codigo de AH=2
               ⋮
            jmp salirIF
tres:  cmp AH,03
            jne cuatro
            ;codigo de AH=3
               ⋮
            jmp salirIF
cuatro:     cmp AH,04
            jne otra
            ;codigo de AH=4
               ⋮
salirIF:    -------------
```

- **FOR**

```
        mov cx,20
xx:     -------
        -------
        -------
        -------
        loop xx
```

```
        mov cont,1
for:    -------
        -------
        -------
        dec cont
        jnz for
```

Tema 5. Programación Modular.
Diseño de Programas

3. Directivas
4. Diseño de Programas
   - 3.1 Programación Modular
   - 3.2 Macros
5. Depuración

Tema 5. Programación Modular.
Diseño de Programas

- A medida que los programas crecen en tamaño y complejidad se vuelve difícil depurarlos. Por lo que resulta conveniente dividir el programa en varias partes llamadas procedimientos.
- Un **procedimiento** es una colección de instrucciones relacionadas que realiza una tarea específica. También un procedimiento puede contener un conjunto de instrucciones que se ejecutan en varias partes del programa.
- Los procedimientos del lenguaje ensamblador tienen su contraparte en los lenguajes de alto nivel, por ejemplo, en el lenguaje C estos procedimientos se llaman funciones. Aunque en los lenguajes de alto nivel, el mecanismo empleado por los procedimientos para implementar su llamada, ejecución y regreso es transparente para el usuario, en ensamblador se requiere entender ese mecanismo. Un componente indispensable empleado por ese mecanismo es la pila del programa.
- La pila de un programa también se emplea para implementar el paso de parámetros a los procedimientos así como para crear las variables locales al procedimiento.

Tema 5. Programación Modular.
Diseño de Programas

Procedimientos

Un procedimiento es una colección de instrucciones que realizan una tarea específica: Sumar o restar dos valores, desplegar un carácter en la pantalla, leer un carácter del teclado, inicializar un puerto, etc.

Un programa puede contener uno o varios procedimientos dependiendo de su extensión y complejidad. Para emplear un procedimiento en un programa se requiere definir el procedimiento y llamarlo. Al definir a un procedimiento escribimos las instrucciones que contiene.

Al llamar al procedimiento transferimos el control del flujo del programa al procedimiento para que sus instrucciones se ejecuten.

Definición de un procedimiento

La sintaxis de la definición de un procedimiento es la siguiente:

nomProc **proc**
;instrucciones del procedimiento
...
ret
nomProc **endp**

Las directivas **proc** y **endp** marcan el inicio y el final del procedimiento. No generan código.

nomProc es el nombre del procedimiento y etiqueta la primera instrucción del procedimiento.

Al menos una de las proposiciones de un procedimiento es la instrucción **ret**, la cual permite regresar el control del programa a la siguiente instrucción de la llamada al procedimiento.

Llamada a un procedimiento

La llamada a un procedimiento normalmente tiene la siguiente forma:

call *nomProc*

En donde *nomProc* es el nombre que se le dio al procedimiento al definirlo.

Tema 5. Programación Modular.
Diseño de Programas

Consideraciones sobre el diseño de un procedimiento

Un procedimiento bien diseñado debe llenar los siguientes requisitos:

- Debe hacer una sola tarea.
- Debe ser tan pequeño como sea posible y tan largo como sea necesario.
- Debe contener un comentario describiendo su propósito, sus datos de entrada y de salida.
- Debe entenderse por sí solo sin necesidad de saber que hace el programa completo.
- El comportamiento interno del procedimiento debe ser transparente para el usuario. El usuario sólo debe saber cómo llamar al procedimiento, esto es, qué datos se le deben suministrar y cómo regresa el resultado. A esto se le llama la **interface del procedimiento**.

❖ Para lograr esa transparencia se requiere que:

Un procedimiento, desde el punto de vista del usuario, funcione como lo haría una instrucción del microprocesador. Por lo tanto un procedimiento sólo puede alterar los registros en los que recibe los datos, los registros en los que regresa el resultado o el registro de banderas.

Un procedimiento tampoco debe utilizar variables globales ni para recibir datos o regresar un resultado, ni para almacenar temporalmente resultados.

Tema 5. Programación Modular.
Diseño de Programas

Uso de los registros en un procedimiento

Un procedimiento puede utilizar los registros de propósito general de tres formas diferentes:

- Para recibir valores o direcciones del código que lo llama.
- Para regresar valores o direcciones al código que lo llama.
- Como variables locales del procedimiento o como registros de trabajo de las instrucciones del procedimiento.
- Sólo debe modificar los registros que emplea para recibir datos o regresar resultados. Si un procedimiento emplea un registro como variable local o registro de trabajo, debe guardar el valor original de ese registro en la pila del programa al entrar al procedimiento y recuperar su valor antes de salir del procedimiento.

Tema 5. Programación Modular.
Programación Modular

Programación Modular

Se puede realizar programas en ensamblador donde todo el código del programa reside en un sólo archivo. Sin embargo es posible y en muchos casos deseable **dividir en código de un programa en varios archivos llamados módulos**.

Como ventajas de emplear la técnica llamada programación modular tenemos:

- Hay un límite en el tamaño de un archivo que un programa ensamblador puede manejar. Si el código del programa crece mucho, al dividirlo en módulos, tendremos archivos de menor tamaño. Como cada módulo puede ensamblarse por separado, esta técnica nos permite ensamblar programas más grandes.
- El proceso de ensamblado de un módulo es menor que el de un programa completo. Cuando se está en la etapa de construcción / depuración de un programa, por lo general uno escribe y depura un procedimiento a la vez, por lo que sólo se está haciendo cambios a un módulo, y sólo ese módulo requiere volver a ensamblarlo.
- Para obtener un programa ejecutable, los códigos objeto obtenidos al ensamblar cada módulo deben ligarse. Normalmente, cada módulo contiene un conjunto de procedimientos relacionados. Estos procedimientos pueden emplearse para formar diferentes programas, ligándolos con otros módulos.
- Podemos utilizar otros módulos escritos por terceros para resolver problemas específicos.

Tema 5. Programación Modular.

Programación Modular (continuación)

- Podemos escribir programas escritos en lenguaje de alto nivel y que utilicen módulos escritos en ensamblador.
- Cada módulo en ensamblador puede contener declaraciones de constantes, declaraciones de variables y definiciones de procedimientos. La estructura de cada módulo es similar a la estructura de un programa sin módulos con las siguientes diferencias:

- Sólo uno de los módulos contiene las instrucciones que inicializan el segmento de datos y las instrucciones que terminan el programa y regresan el control al sistema operativo. A este módulo se le conoce como el módulo principal.

- Todos los módulos tiene la directiva **end** que indica el final del código de ese módulo. Sin embargo sólo en el módulo principal la directiva **end** va seguida de la etiqueta que apunta al punto de entrada del programa. En los otros módulos la directiva **end** aparece por sí sola.

- Las constantes, variables y procedimientos declaradas y definidos en un módulo pueden ser conocidas sólo en el módulo donde fueron declaradas o pueden ser conocidas en otros módulos también.

Tema 5. Programación Modular.
Programación Modular (continuación)

- Los símbolos de las constantes, variables y procedimientos de un módulo que deseamos que se conozcan en otros módulos deben de exportarse utilizando la directiva **public**.
- La sintaxis de la directiva public es:

  public *nomSimb*[, *nomSimb*] ...

  donde *nomSimb* es el nombre de la constante, variable o procedimiento cuyo símbolo se va exportar. Los símbolos de las constantes declaradas con la directiva **equ** no pueden exportarse.
- Los símbolos de las constantes, variables y procedimientos declaradas y definidos en otro módulo y que fueron exportados con la directiva **public** tienen que importarse en el módulo en que se van a usar.
- Para importar un símbolo se utiliza la directiva **extrn**, cuya sintaxis es:

  extrn *nomSimb***: tipo**[**,** *nomSimb***: tipo**] ...

  donde *nomSimb* es el nombre de la constante, variable o procedimiento cuyo símbolo se va importar.
- **tipo** especifica el tipo de símbolo. Para los símbolos de las variables a importar los tipos son: **byte**, **word**, **dword**, **qword**, etc. y para los procedimientos el tipo es **proc**.

Tema 5. Programación Modular.
Programación Modular (continuación)

```
title Test Alphabetic Input
        (Pract1.ASM)
; This program reads and displays
        characters
; until a non-alphabetic character is
        entered.

pila segment stack
        DB 100h DUP ('stack')
   TOS  EQU THIS WORD
pila ends
codigo segment 'code'
    assume ss:pila, cs: codigo
main proc far
     mov ax, seg pila
     mov ss,ax
L1: mov  ah,1       ; input a character
    int  21h        ; AL = character
    call  Isalpha    ; test value in AL
    jnz  exit       ; exit if not alphabetic
    jmp  L1         ; continue loop

exit:
    mov   ax,4C00h  ; return to DOS
    int   21h
main endp
```

```
; Isalpha sets ZF = 1 if the character
; in AL is alphabetic.

; Isalpha sets ZF = 1 if the character
; in AL is alphabetic.

Isalpha proc far

    push ax           ; save AX
    and  al,11011111b ; clear bit 5
    cmp  al,'A'       ; check 'A'..'Z'
        range
    jb   B1
    cmp  al,'Z'
    ja   B1
    test ax,0         ; ZF = 1
B1: pop ax            ; restore AX
    ret
Isalpha endp
codigo ends
end main
```

```
Microsoft (R) Macro Assembler Version 6.13.7299
Test Alphabetic Input        (Pract1.ASM)                    Page 1 - 1
                                        title Test Alphabetic Input        (Pract1.ASM)
                                        ; This program reads and displays characters
                                        ; until a non-alphabetic character is entered.
 0000                                   pila segment stack
 0000  0100 [                                   DB 100h DUP ('stack')
       73 74 61 63 6B
       ]
 0500 = 0500                                    TOS  EQU THIS WORD
 0500                                   pila ends
 0000                                   codigo segment 'code'
                                           assume ss:pila, cs: codigo
 0000                                   main proc far
 0000  B8 ---- R                              mov ax, seg pila
 0003  8E D0                                  mov ss,ax
 0005  B4 01                             L1: mov  ah,1       ; input a character
 0007  CD 21                                  int  21h       ; AL = character
 0009  0E E8 0009                             call  Isalpha    ; test value in AL
 000D  75 02                                  jnz  exit      ; exit if not alphabetic
 000F  EB F4                                  jmp  L1         ; continue loop
 0011                                   exit:
 0011  B8 4C00                               mov   ax,4C00h  ; return to DOS
 0014  CD 21                                 int   21h
 0016                                   main endp
                                        ; Isalpha sets ZF = 1 if the character
                                        ; in AL is alphabetic.
 0016                                   Isalpha proc far
 0016  50                                     push ax          ; save AX
 0017  24 DF                                  and  al,11011111b ; clear bit 5
 0019  3C 41                                  cmp  al,'A'      ; check 'A'..'Z' range
 001B  72 07                                  jb   B1
 001D  3C 5A                                  cmp  al,'Z'
 001F  77 03                                  ja   B1
 0021  A9 0000                                test ax,0        ; ZF = 1
 0024  58                               B1:  pop ax           ; restore AX
 0025  CB                                     ret
 0026                                   Isalpha endp
 0026                                   codigo ends
                                        end main
```

Programa Ensamblado

```
Microsoft (R) Macro Assembler Version 6.13.7299
Test Alphabetic Input         (Pract1.ASM)                    Symbols 2 - 1
Segments and Groups:
                N a m e                 Size     Length   Align   Combine Class
codigo . . . . . . . . . . . . .     16 Bit     0026      Para      Private 'CODE'
pila . . . . . . . . . . . . . .     16 Bit     0500      Para      Stack
Procedures,  parameters and locals:
                N a m e                 Type     Value    Attr
Isalpha  . . . . . . . . . . . .     P Far      0016       codigo  Length= 0010 Private
main . . . . . . . . . . . . . .     P Far      0000       codigo  Length= 0016 Private
Symbols:
                N a m e                 Type     Value    Attr
B1 . . . . . . . . . . . . . . .     L Near     0024       codigo
L1 . . . . . . . . . . . . . . .     L Near     0005       codigo
TOS  . . . . . . . . . . . . . .     Number     0500h
exit . . . . . . . . . . . . . .     L Near     0011       codigo
        0 Warnings
        0 Errors
```

Tema 5. Programación Modular.
Programación Modular (continuación)

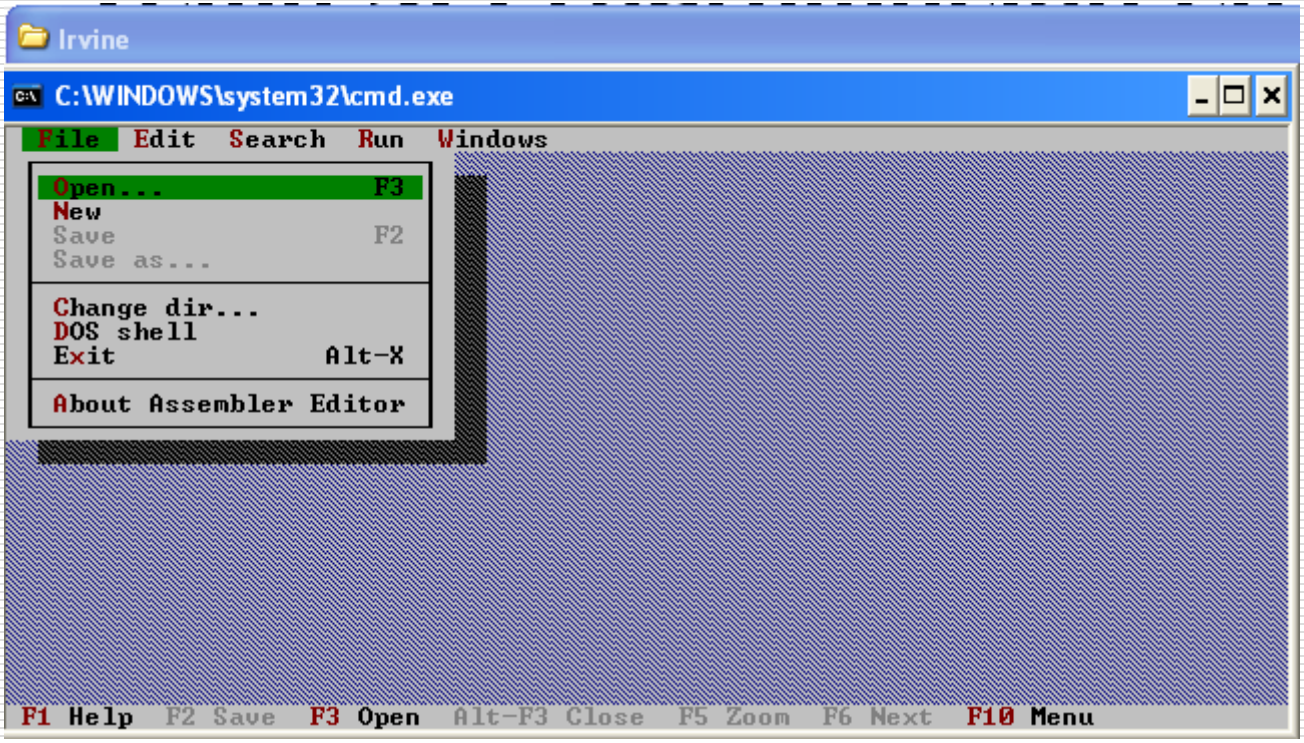
```
title Test Alphabetic Input          (Pract1.ASM)
; This program reads and displays characters
; until a non-alphabetic character is entered

pila1 segment stack
        DB 100h DUP ('stack')
    TOS EQU THIS WORD
pila1 ends

s_code segment 'code'
assume ss:pila1, cs:s_code
extrn Isalpha: far
 main proc far
     mov ax, seg pila1
     mov ss,ax
L1: mov  ah,1        ; input a character
    int  21h         ; AL = character
    call  Isalpha    ; test value in AL
    jnz  exit        ; exit if not alphabetic
    jmp  L1          ; continue loop
exit:
    mov   ax,4C00h  ; return to DOS
    int   21h
main endp
s_code ends
end main
```

```
title Test Alphabetic Input          (Pract1.ASM)
; Isalpha sets ZF = 1 if the character
; in AL is alphabetic

codigo segment 'code'

public Isalpha

Isalpha proc far
    assume  cs: codigo
    push ax           ; save AX
    and  al,11011111b ; clear bit 5
    cmp  al,'A'       ; check 'A'..'Z' range
    jb   B1
    cmp  al,'Z'
    ja   B1
    test ax,0         ; ZF = 1
B1: pop ax            ; restore AX
    ret
Isalpha endp
codigo ends
end
```

Tema 5. Programación Modular.
El Lenguaje de Máquina y el Lenguaje Ensamblador.

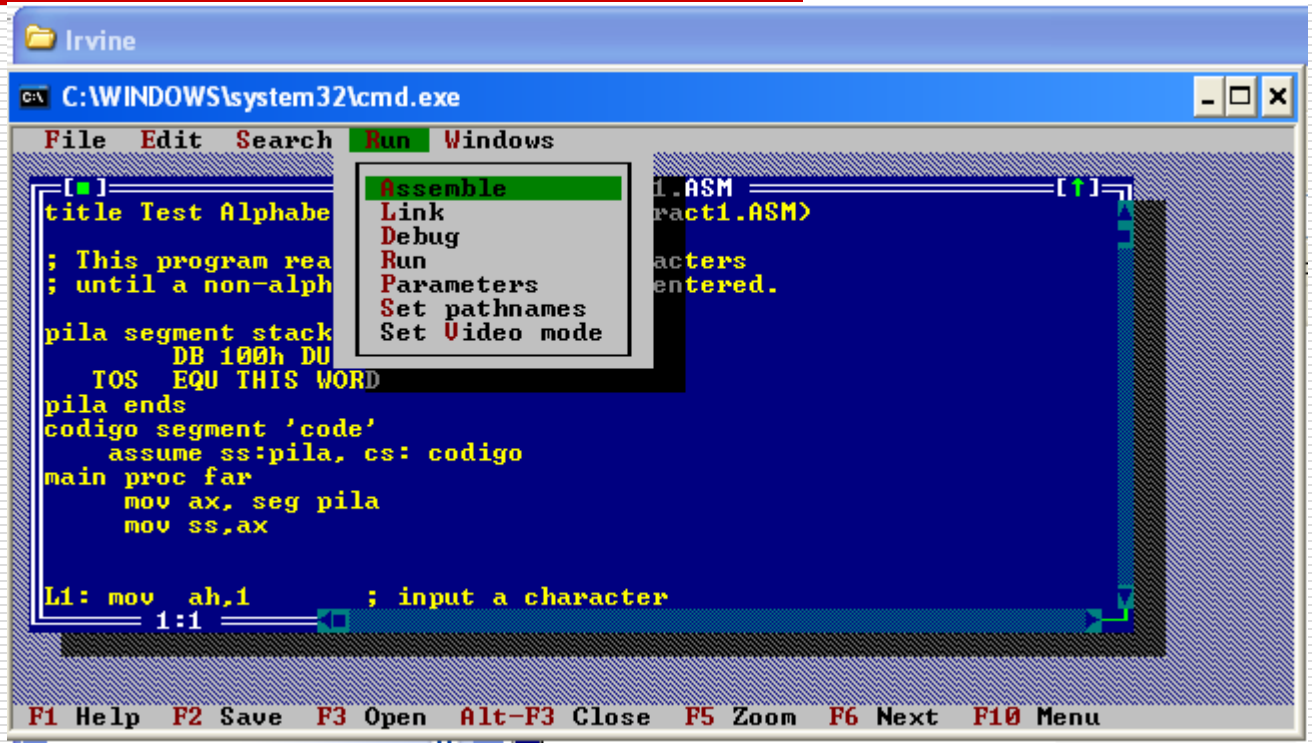
- **El Lenguaje de Máquina** consta de una serie de instrucciones, que son las únicas que pueden ser reconocidas y ejecutadas por el microprocesador a través de su circuitería interna.
- El lenguaje máquina de un microprocesador no puede ser ejecutado por otro microprocesador de arquitectura distinta, a menos que haya cierto tipo de compatibilidad prevista.
- Con el lenguaje máquina obtenemos un control total del microprocesador, sin embargo la programación resulta difícil. No sólo por que las instrucciones son números, sino por que se debe calcular y trabajar con las direcciones de memoria de los datos, los saltos y las direcciones de llamadas a subrutinas, además de que para poder hacer ejecutable un programa, se deben enlazar las rutinas y servicios del sistema operativo. Para facilitar la elaboración de programas a este nivel, se desarrollaron los **Ensambladores (software que traduce de lenguaje simbólico a lenguaje máquina)** y el **Lenguaje Ensamblador.** Existe una correspondencia 1 a 1 entre las instrucciones del **lenguaje máquina** y la del **lenguaje ensamblador**. Cada valor numérico del lenguaje de máquina tiene una representación simbólica de 3 a 5 letras como instrucción del lenguaje ensamblador.
- El **Ensamblador** proporciona un conjunto de **pseudo-operaciones** (**directivas**) que sirven para definir datos, rutinas y todo tipo de información para que el programa ejecutable sea creado de determinada forma y en un determinado lugar. Estas directivas son únicamente reconocidas por el ensamblador.
- La diferencia entre los ensambladores radica en la forma de generar el código y las directivas con que cuentan. Los programas que se crean con un ensamblador en particular podrán ser ensamblados en otro, cambiando las pseudo-operaciones no reconocidas por sus equivalentes.

Tema 5. Programación Modular.
El Lenguaje de Máquina y el Lenguaje Ensamblador.



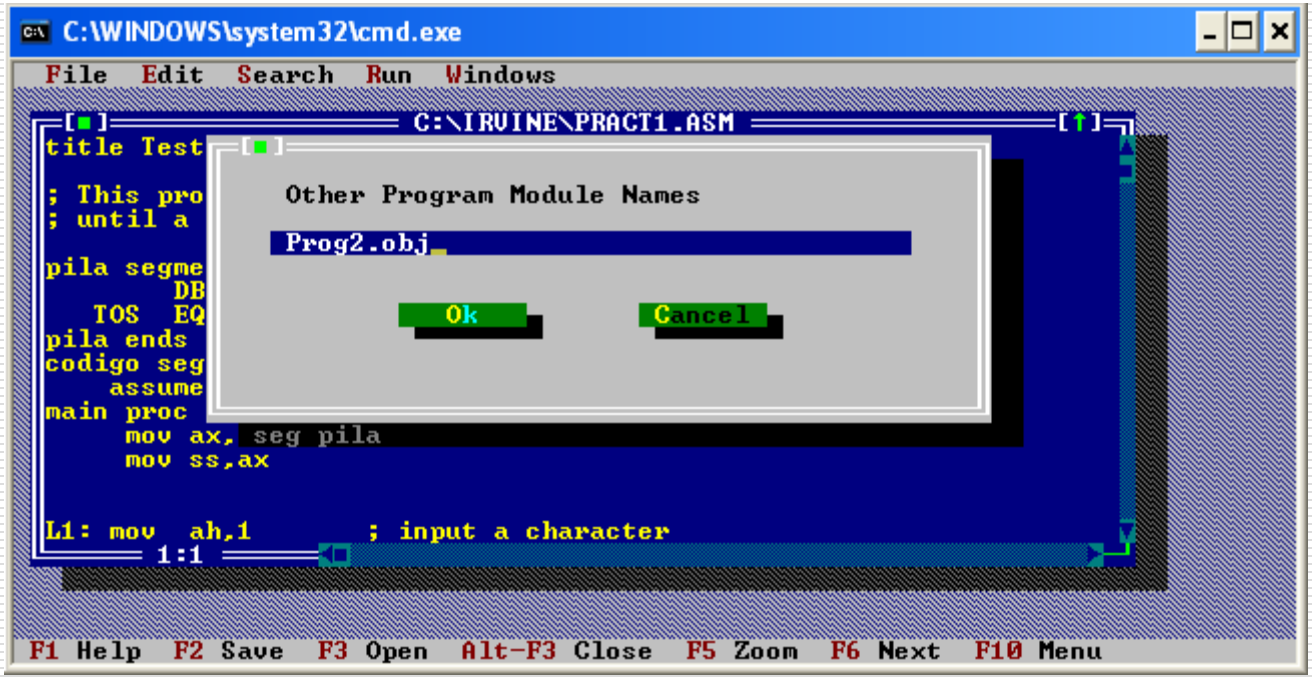
- Edición del archivo
- Creación de un archivo en un editor Archivo.asm

Tema 5. Programación Modular. El Lenguaje de Máquina y el Lenguaje Ensamblador.



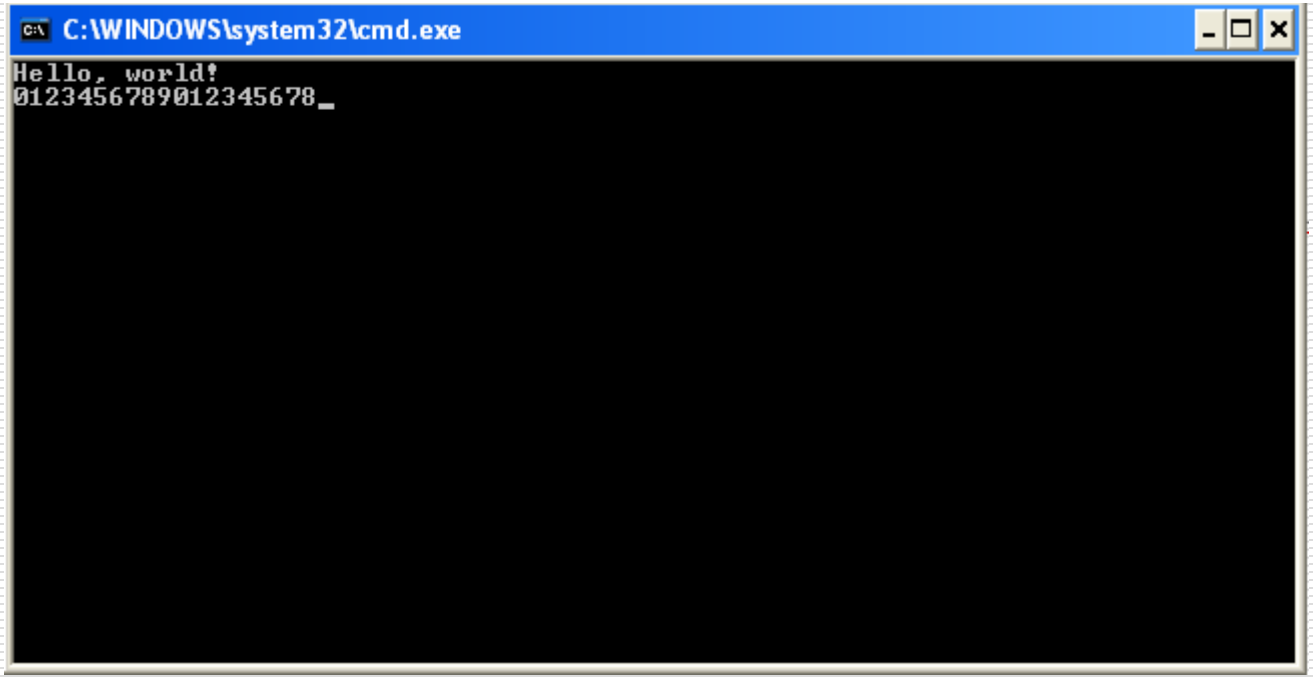
- Ensamblado del archivo
- Programa ensamblador: MASM, TASM
- Genera un Archivo.obj

Tema 5. Programación Modular.
El Lenguaje de Máquina y el Lenguaje Ensamblador.



- Ligador
- Enlaza los módulos para generar un programa
- Genera Archivo.exe

Tema 5. Programación Modular. El Lenguaje de Máquina y el Lenguaje Ensamblador.



- Ejecución del Programa Ligador

Formato de un programa en Lenguaje Ensamblador

Formato de un programa que incluye procedimientos (subrutinas)

; Comentarios introductorios

SSEG SEGMENT STACK

- Contenido del segmento de pila-

SSEG ENDS

DSEG SEGMENT

- Contenido del segmento de datos-

DSEG ENDS

ESEG SEGMENT

- Contenido del segmento extra-

ESEG ENDS

CSEG SEGMENT 'CODE'

MAIN PROC FAR

- Contenido del programa principal-

MAIN ENDP

PROC1 PROC

- Contenido del procedimiento 1-

PROC1 ENDP

PROC2 PROC

- Contenido del procedimiento 2-

PROC2 ENDP

CSEG ENDS

Segmento de código

END MAIN

DIRECTIVAS

Las directivas indican cómo procesará el ensamblador un operando o una sección de programa. Algunas directivas generan y almacenan información en la memoria

DIRECTIVA	FUNCION
ASSUME	Indica los nombres de cada segmento al ensamblador, no carga los registros de segmento.
AT	Indica la dirección física de segmento que se emplea con el enunciado SEGMENT
BYTE	Indica un operando de tamaño byte, se usa con BYTE PTR O THIS BYTE
DB	Define (reserva en memoria) uno o más bytes
DD	Define (reserva en memoria) dobles palabras
DQ	Define (reserva en memoria) cuádruples palabras
DT	Define (reserva en memoria) 10 bytes (80 bits)
DUP	Genera duplicados de caracteres o números
DW	Define (reserva en memoria) palabras (16 bits)
DWORD	Indica un operando de tamaño doble palabra cono en THIS DWORD
END	Indica el final del programa
ENDM	Indica el final de una secuencia de MACRO
ENDP	Indica el final de un procedimiento
ENDS	Indica el final de un segmento
EQU	Iguala los datos con los de una etiqueta

DIRECTIVAS

Las directivas indican cómo procesará el ensamblador un operando o una sección de programa. Algunas directivas generan y almacenan información en la memoria

DIRECTIVA	FUNCION
FAR	Especifica una dirección lejana como en JMP FAR
MACRO	Define el nombre, parámetros e inicio de un MACRO
NEAR	Especifica una dirección cercana como en JMP NEAR
OFFSET	Especifica una dirección de desplazamiento
ORG	Inicializa el origen dentro de un segmento
PROC	Define el inicio de un procedimiento
PTR	Indica un apuntador a la memoria
SEGMENT	Define el comienzo de un segmento de memoria
STACK	Indica que un segmento es segmento de pila
THIS	Se emplea con EQU para establecer una etiqueta de un byte, palabra o doble palabra
USES	Directiva de la versión 6.0 de MASM que salva en forma automática los registros utilizados en un procedimiento
WORD	Actúa como operando tamaño palabra como en WORD PTR o THIS WORD
EXTRN	Indica al ensamblador que el elemento llamado – dato, procedimiento o etiqueta – está definido en otro programa ensamblado. EXTRN nombre:tipo
PUBLIC	Indica al ensamblador y al enlazador que la dirección de un símbolo especificado en el ensamblado actual estará disponible para otros módulos.

Tema 5. Programación Modular. MACROS

- Un **macro** es un nombre simbólico dado a uno o más enunciados en lenguaje ensamblador, una vez definido, el **macro** puede ser invocado en el programa las veces que se desee. Al invocar al macro, una copia de las declaraciones del macro se inserta directamente en el programa.
- Un **macro** generalmente se ejecuta más rápido que un procedimiento, ya que las instrucciones se encuentran de manera consecutiva, mientras que en un procedimiento se utiliza la instrucción CALL, la cual guarda en la pila información de dirección de retorno, salta al procedimiento y cuando encuentra la instrucción RET regresa a la siguiente instrucción del CALL, esto implica un cierto tiempo de procesamiento.
- En el caso de un macro existe la desventaja de que al invocar macros complejos se tiende a aumentar el tamaño del programa debido a que cada llamada del macro inserta en el programa una copia de todos los enunciados que forman el macro.

  Los macros pueden incluir parámetros.

```
despCaracter macro Par1
        mov dl,Par1
        mov ah,2
        int 21h
Endm
```

En el segmento de código, el macro puede invocarse de la siguiente manera;

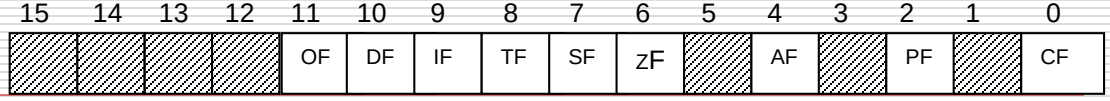
```
----------
despCaracter 'A'
---------
despCaracter al
```

Expansión del macro:

```
----------
mov dl, 'A'
mov ah,2
int 21h
---------
mov dl,al
mov ah,2
int 21h
```

Tema 5. Programación Modular. Depuración

- La interrupción tipo 1 Es la interrupción modo paso a paso o de trampa: ocurre después de la ejecución de cada instrucción si la bandera T está en uno. Se usa para aplicaciones de depuración de un programa al permitir avanzar por paso a través de la ejecución de un programa.
- Cuando queremos depurar un programa es importante visualizar el contenido de los registros.



Tema 5. Programación Modular. Depuración

```
title Programa Depurador (Prac.asm)
; Este programa permite la ejecución paso a
; paso mientras TF=1.

pila segment stack
        DB 500h DUP (0)
  TOS  EQU THIS WORD
pila ends
codigo segment 'code'
   assume ss:pila, cs: codigo
main proc far
       mov ax, seg pila
       mov ss,ax
       mov sp, offset TOS
; codigo para cambio de vector
; codigo para prender bandera T
; programa a depurar
       mov ax,0fffh
       mov bx,00
       mov cx,4
Ciclo: nop
      mov bp,bx
      inc bx
      dec ax
      loop Ciclo
;codigo para apagar bandera T
exit:
   mov   ax,4C00h  ; return to DOS
   int   21h
main endp

; datos para desplegar letreros

SubInt01 proc

;codigo de subrutina para visualizar registros

SubInt01 endp

Codigo ends
end main
```

Tema 5. Programación Modular. Depuración, Código de subrutina de atención a interrupción.

```
LetreroReg DB `AX = `, `BX = `, `CX = `, `DX = `
 DB  `SP = `, `BP = `, `SI = `, `DI = `
 DB  `IP = `, `FL = `, `CS = `, `DS = `
 DB  `ES = `, `SS = `
    DESP    MACRO  PAR1
                PUSH AX
                PUSH DX
                MOV DL,PAR1
                MOV AH,6
                INT 21H
                POP DX
                POP AX
                ENDM
  CRLF   MACRO
                DESP 13
                DESP 10
                ENDM
SUBINT1 PROC FAR USES AX BP BX
                MOV BX,OFFSET LetreroReg
                CRLF
                CALL DREG
                POP AX
                PUSH AX
                CALL DREG
                MOV AX,CX
                CALL DREG
                MOV AX,DX
                CALL DREG
                MOV AX,SP
                ADD AX,12
                CALL DREG
                MOV AX,BP
                CALL DREG
                 MOV AX,SI
                CALL DREG
                MOV AX,DI
                CALL DREG
                MOV BP,SP
                MOV AX,[BP+6]
                CALL DREG
                MOV AX,[BP+10]
                CALL DREG
                MOV AX,[BP+8]
                CALL DREG
                MOV AX,DS
                CALL DREG
                MOV AX,ES
                CALL DREG
                MOV AX,SS
                CALL DREG
                MOV AH,7
                INT 21H
                IRET
SUBINT1 ENDP
DREG PROC NEAR USES CX
                MOV CX,5
        DREG1:  DESP CS:[BX]
                INC BX
                LOOP DREG1
                MOV CX,4
        DREG2:  ROL AX,1
                ROL AX,1
                ROL AX,1
                ROL AX,1
                PUSH AX
                AND AL,0FH
                CMP AL,9
                JNA SIGUE
                ADD AL,7
        SIGUE:  ADD AL,30H
                DESP AL
                POP AX
                LOOP DREG2
                DESP ` `
                RET
DREG ENDP
```

Tema 5. Programación Modular.

OPCIONES DE INT 21H (LLAMADAS A FUNCIONES DE DOS)

AH	Propósito	Tipo	Descripción.
0	Terminación del programa	Control	Termina la ejecución de un programa.
1	Entrada desde el teclado	Teclado	Espera entrada proveniente del teclado, la exhibe y la coloca en el registro AL.
2	Exhibe salida	Display	Exhibe el carácter en DL.
3	Entrada auxiliar	Diversos	Espera un carácter proveniente del puerto COM y lo coloca en AL.
4	Salida auxiliar	Diversos	Envía puerto COM al carácter en DL
5	Salida a impresora	Impresora	Envía a la impresora el carácter en DL.
6	I/O directo de consola	Teclado	Espera hasta recibir un carácter proveniente del teclado (no verifica ctrl.-Break).
7	Entrada de consola directa con eco desactivado	Teclado	Espera hasta recibir un carácter desde del teclado y lo coloca en AL.
8	Entrada desde la consola	Teclado	Espera hasta recibir un carácter desde el teclado, entrega en Al y se ejecuta una interrupción Ctrl.-Break.
9	Impresión en cadena	Display	Presenta una cadena de caracteres en la pantalla. La cadena debe finalizar en $, apuntando DS:DX.
A	Entrada desde el teclado a través del buffer	Teclado	Lee los caracteres que provienen del teclado en un buffer. DS:DX apunta al buffer. El primer byte es el numero máximo de caracteres mientras que el segundo byte indica el numero de caracteres leídos.
B	Verifica el estado de entrada normal	Teclado	Verifica si existe un carácter disponible proveniente del teclado. (AL=0 NO ,AL=0FFH SI)
C	Limpia el buffer del teclado e invocar una función del teclado	Teclado	Limpia el buffer del teclado y ejecuta la llamada a la función AL (únicamente 01H, 06H, 07H, 08H o 0AH).
D	Restablece el disco	Disco	Se pierden todos los archivos que no han sido cerrados.
E	Selección del disco	Disco	Selecciona la unidad del disco en DL (0 =A, 1 = B, etc).
F	Abre archivo	Archivo	Busca el directorio para apuntar el archivo que entra en DS:DX. AL = FFH (no se encuentra) o AL = 00H (encontrado). Si se encuentra se llena FCB.
10	Cierra archivo	Archivo	Cierra el archivo después de una operación de escritura. DS:DX apunta a FCB.
11	Búsqueda para la primera entrada	Disco	Busca en el directorio la primera ocurrencia en que igual el nombre del archivo. Si no se encuentra AL = FFH.
12	Búsqueda para la siguiente entrada	Disco	Después de haber encontrado el nombre del archivo, esta llamada continuara la búsqueda para la siguiente ocurrencia.
13	Borrar archivo	Archivo	Borra del directorio todas las entradas que señala el apuntador DS:DX.

Tema 5. Programación Modular.

OPCIONES DE INT 21H (LLAMADAS A FUNCIONES DE DOS)

14	Lectura secuencial	Disco	Carga el registro direccionado por el bloque actual y la graba en DTA e incrementa la dirección del registro.
15	Escritura secuencial	Disco	Lo contrario a 14H.
16	Crear archivo	Archivo	Busca en el directorio la entrada deseada, si la encuentra la utiliza nuevamente, de lo contrario abre un archivo nuevo.
17	Renombra un archivo	Archivo	Cambia el nombre del archivo por el nombre DS:DX + 11.
19	Unidad de disco actual	Disco	Determina el default de la unidad del disco la AL.
1A	Coloca la DTA del disco	Disco	Coloca la dirección de transferencia de disco en DS:DX.
1B	Información de la tabla de asignación	Disco	Entrega un apuntador contenido en DS:BX apunta al byte descriptor del medio, DX = numero de la unidad de asignación , AL = numero del sector / unidad de asignación y CX = tamaño del sector.
1C	Tabla de información de asignación para la unidad de disco	Disco	DL = numero de la unidad de disco; esta función proporciona el mismo parámetro que 1CH.
21	Lectura aleatoria	Disco	Lee la grabación direccionada por el bloque actual y registra los campos en área de memoria, correspondiente a DTA.
22	Escritura aleatoria	Disco	Lo contrario a la 21H
23	Tamaño de archivo	Archivo	Busca en el directorio una entrada a igualar según DS:DX y coloca el registro de grabación aleatorio FCB igual al numero de grabaciones en el archivo.
24	Campo de registro relativo	Archivo	Coloca el campo de registro aleatorio en la misma dirección que el bloque actual y los campos del registro
25	Coloca vector de interrupción	Diversos	Coloca el vector de la interrupción en AL en la dirección DS:DX
26	Crea nuevo segmento de programa	Diversos	Esta llamada nunca debe utilizarse
27	Lectura de bloque aleatorio	Disco	Lee el numero de registrasen CX desde DS:DX, en DTA.
28	Escritura de bloque aleatorio	Disco	Lo contrario a la 27H
29	Analizar el nombre de archivo	Archivo	Véase el manual DOS Technical referente.
2A	Obtener la fecha	Diversos	Regresa AL = día de la semana (Dom=0, Lun=1,….Sab=6) CX = año, DH = mes DL = día del mes.
2B	Coloca la fecha	Diversos	Inverso a la 2AH
2C	Obtener la hora	Diversos	Regresa CH = hora, CL = minutos, DH = segundos y DL = centésimos de segundo.
2D	Coloca la hora	Diversos	Servicio opuesto al 2CH
2E	Activa / desactiva switch de verificación	Diversos	Cuando se encuentra activada, DOS realiza la verificación para cada operación de escritura en disco. AL = 0 desactivar; AL = activar

Tema 5. Programación Modular.

OPCIONES DE INT 21H (LLAMADAS A FUNCIONES DE DOS)

2F	Obtiene DTA	Disco	Regresa la dirección de transferencia en ES:BX
30	Obtener la versión del DOS	Diversos	Regresa en AL el numero superior que corresponde ala versión de DOS; AH contiene el numero inferior
31	Terminación del proceso / conserva residente	Diversos	Véase el manual DOS Technical referente.
33	Verificar ctrl.-Break	Diversos	Solicita coloca el estado break AL = 0 y AL = 1. si DL =0 desactiva si DL = 1 activa
35	Obtener vector	Diversos	Para el numero de interrupción de AL, regresa el apuntador en ES:BX
36	Obtiene espacio libre en disco	Disco	Regresa para DL: en BX, los espacios disponibles; en DX, cluster/unidad de disco; en CX, los bytes/sector; y en AX los sector / cluster.
38	Información dependiente del país	Diversos	Véase el manual DOS Technical referente.
39	Crear subdirectorio	Disco	Genera la función MKDIR; con DS:DX apuntando a una cadena ASCIIZ que contiene la unidad de disco.
3A	Elimina subdirectorio	Disco	Función RMDIR; DS:DX apunta a la cadena que contiene los nombres de la unidad de disco y la ruta.
3B	Cambio de directorio	Disco	Función CHDIR: DS:DX apunta a la cadena que contiene los nombres de la unidad de disco y ruta.
3C	Crear archivo	Archivo	Función CREATE; si el archivo al que apunta DS:DX si existe se abre un nuevo archivo.
3D	Abre archivo	Archivo	DS:DX apunta al archivo; AL = 0 (solo lectura), 1 (solo escritura) o 2 (escritura / lectura).
3E	Cierra manejador de archivo	Archivo	BX = manejador de archivo; se cierra el archivo, se actualiza el directorio y se remueven los buffers internos del archivo.
3F	Lectura desde archivo / dispositivo	Archivo	BX = manejador de archivo, CX = numero de bytes que se desea leer y DS:DX = buffer a ser cargado; después de la llamada, AX = numero de bytes leídos.
40	Escritura en un archivo / dispositivo	Archivo	Operación inversa a 3F.
41	Borra archivo del directorio	Archivo	Elimina una entrada del directorio asociada con el nombre del archivo apuntado en DS:DX.
42	Mueve el apuntador de lectura / escritura del archivo	Archivo	Véase el manual DOS *Technical Reference*.
43	Cambia modo de un archivo	Archivo	Véase el manual DOS *Technical Reference*.
44	Control de I/O para archivos	I/O	Véase el manual DOS *Technical Reference*.
45	Manejador de archivo duplicado	Archivo	A la entrada BX = manejador de archivo al terminar, AX = duplicado.
46	Fuerza duplicación en el manejador de archivo	Archivo	Fuerza que el manejador en CX se refiera al mismo archivo en la misma posición que el manejador en BX.

Tema 5. Programación Modular.

OPCIONES DE INT 21H (LLAMADAS A FUNCIONES DE DOS)

47	Obtiene el directorio actual	Disco	DL = numero de la unidad del disco; DS:SI = puntador al área del usuario de 64 bytes, la que contiene el directorio; AX contiene el código de error.
48	Asigna memoria	Memoria	BX = numero de párrafos, y AXL0000 apunta a los bloques de asignación.
49	Libera memoria asignada	Memoria	Libera la memoria asignada con 48H.
4A	Modifica los bloques de memoria asignada	Memoria	Modifica los bloques para contener el tamaño de un bloque nuevo. ES = bloque del segmento BX = tamaño en párrafos del nuevo bloque.
4B	Carga / ejecuta programa	Control	Permite que un programa de aplicación ejecute otro. Al término de este, el control vuelve al primer programa. DS:DX apunta al programa y ES:BX apunta a un bloque de parámetros. Para la carga.
4C	Termina proceso	Control	Finaliza el proceso de ejecución.
4D	Obtiene el código de retorno	Diversos	Véase el manual DOS *Technical Reference*.
4E	Encuentro del primer archivo indicado	Archivo	Encuentro del primer nombre de archivo indicado que se iguala al nombre de archivo que apunta DS: DX. CX = atributo de búsqueda.
4F	Encuentra el siguiente archivo indicado	Archivo	Es igual al 4AH con la excepción de que encuentra el segundo archiva. La DTA contiene, en este caso información proporcionada por 4EH o por una llamada previa a 4FH
54	Obtiene el estado de verificación	Diversos	Regresa el valor de la verificación con 2EH en AL
56	Renombra un archivo	Archivo	Modifica el nombre de un archiva DS:DX con ES:DI
57	Obtiene/coloca fecha y hora de un archivo	Diversos	A la entrada AL = 0 (obtener) o AL = 1 (Fijar), BX = manejador del archivo, CX = hora y DX = fecha.
59	Obtiene error extendido (DOS 3.00 y 3.10)	Error	Proporciona información adicional de un error. Véase el manual DOS *Technical Reference*.
5A	Crea archivo único	Archivo	Genera un archivo apuntado por DS:DX (la ruta debe terminar con /), CX = atributo
5B	Crea un nuevo archivo	Archivo	Genera un nuevo archivo apuntado por DS:DX DX = atributo
5C	Abre / cierra acceso de un archivo	Archivo	AL = 0 (abre) o AL = 1 (cierra), BX manejador de archivo, CX = byte mas significativo del desplazamiento, DX = byte menos significativo del desplazamiento, SI = parte mas significativa, DI = parte menos significativa
62	Obtiene PSP	Diversos	Proporciona en BX el segmento prefijo del programa.

Tema 5. Programación Modular.
OPCIONES DE INT 10H (E/S DE VIDEO)

OPCIONES DE INT 10H (I/O DE VIDEO)		
AH	Propósito	Descripción.
0	Modo	El registro AL contiene el modo de video: AL = 0-40 x 25 caracteres blanco y negro, AL = 1-40 x 25 caracteres en color, AL = 2-80 x 25 caracteres blanco y negro, AL= 3-80 x 25 caracteres color, AL = 4-320 x 200 píxeles color, AL = 5-320 x 200 píxeles blanco y negro, AL = 6-640 x 200 píxeles blanco y negro,
1	Selección de tipo de cursor	Esta opción utiliza los registros CH y CL. Los bits 4 a 0 de CH indican la línea donde comienzan el cursos mientras que los bits 4 a 0 de CL señalan donde termina. Los demás bits deben ponerse en cero con el fin de evitar comportamientos erráticos. .
2	Selección de posición del cursos	(DH,DL) = (renglón, columna) donde se colocara el cursos. La esquina superior izquierda corresponde a la posición (0,0). El registro BH contiene el numero de pagina (0 para gráficos).
3	Lectura de posición del cursos	(DH,DL) = (renglón, columna) donde se encuentra el cursos. (CH, CLl) = dimensiones del cursos.
4	Lectura la posición del lector óptico	Véase el manual IBM technical Referente.
5	Selección de pagina desplegada activa	Cuando existen varias paginas en la memoria de video, esta opción permite seleccionar una de ellas para su exhibición en pantallas de 40 x 25 y 80 x 25. AL = 0 -7 para 40 x 25 mientras que para 80 x 25 AL = 0 – 3.
6	Cambio a la pagina anterior activa	AL = el numero de líneas. Las líneas de la parte inferior son puestas en blanco. Si AL = 0 entonces toda la pantalla se pone en blanco. (CH , CL) = coordenadas de la esquina superior izquierda (renglón, columna); (DH,DL) = coordenadas de la esquina superior derecha (renglón, columna). El registro BH = atributo a utilizar para las líneas en blanco.
7	Cambio a al pagina siguiente activa	Identifica ala anterior con la diferencia de que las líneas se ponen en blanco desde la parte superior hacia la parte inferior.
8	Lee atributo y carácter en el cursor	BH = pagina en exhibición, AL = carácter y AH = atributo. Esta opción trabaja únicamente en 80 x 25 y 40 x 25.

Tema 5. Programación Modular.
OPCIONES DE INT 10H (E/S DE VIDEO)

AH	Propósito	Descripción.
9	Escribe atributo y carácter en el cursor	BH = pagina en exhibición, CX =conteo de caracteres, AL = carácter a escribir, BL = atributo del carácter.(Fondo Primer Plano)
A	Escribe carácter en la posición del cursor	Igual que la anterior pero sin atributo.
B	Selección de la pantalla de color	Coloca paleta de color. El usuario debe experimentar con esta opción para seleccionar los registros. Consulte el manual IBM technical Referente.
C	Escribe un punto	DX = numero de región, CX = numero de columna, AL = color (para monitores de alta resolución AL varia la intensidad).
D	Lectura de un punto	DX = numero de región, CX = numero de columna, AL = punto leído.
E	Estructura de caracteres en modo alfanumérico y grafica.	AL = carácter, BL = color de fondo en modo grafico, BH = despliega pagina en modo alfanumérico.
F	Estado actual del video	AL = modo, AH = numero de columnas en pantalla y BH = despliega pagina activa.
13	Despliega una cadena de caracteres	AL = Subfunción, BH = #página, BL = Atributos, dirección de la cadena ES:BP, CX = Longitud, DX = posición en la pantalla (DH,DL) = (renglón, columna)

Tema 5. Programación Modular.
OPCIONES DE INT 10H (E/S DE VIDEO)

Atributos:

Color	I R G B	Color	I R G B
Negro	0 0 0 0	Gris	1 0 0 0
Azul	0 0 0 1	Azul Claro	1 0 0 1
Verde	0 0 1 0	Verde Claro	1 0 1 0
Cian	0 0 1 1	Cian Claro	1 0 1 1
Rojo	0 1 0 0	Rojo Claro	1 1 0 0
Magenta	0 1 0 1	Magenta Claro	1 1 0 1
Café	0 1 1 0	Amarillo	1 1 1 0
Blanco	0 1 1 1	Blanco Brillante	1 1 1 1

Atributo:		Fondo			Frente			
Número de Bit:	BL	R	G	B	I	R	G	B
	7	6	5	4	3	2	1	0

Bit 7: (BL) Establece intermitencia

Bit 6-4: determina el fondo de la Pantalla

Bit 3: (I) Establece la intensidad Alta

Bits 2-0 Determinan el frente o Primer plano (para el Carácter que será Desplegado.