INGENIERÍA DE TELECOMUNICACIÓN CURSO 2009-2010

ESTRUCTURA DE COMPUTADORES

Prácticas de Laboratorio

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

DEPARTAMENTO DE AUTOMÁTICA

Universidad de Alcalá

1: Instrucciones aritméticas. Direccionamiento inmediato y directo.

1. Programa que suma dos números enteros de 8 bits sin signo situados en memoria y almacena el resultado en memoria.

2. Programa que suma dos números enteros de 16 bits sin signo situados en memoria y almacena el resultado en memoria.

3. Programa que suma dos números enteros de 32 bits sin signo situados en memoria y almacena el resultado en memoria. El programa se hará con las dos variantes siguientes:
- Declarando cada dato como una tabla de dos palabras, por ejemplo, `Sumando1 DW 1235h,4565h` y acceder a cada palabra utilizando la dirección más un índice. Por ejemplo, `Sumando1` para la primera palabra del dato y `Sumando1+2` para la segunda.
- Declarando cada dato como doble palabra, por ejemplo, `Sumando1 DD 45651235h` y acceder a cada palabra con la directiva `WORD PTR`. Por ejemplo, `mov ax,WORD PTR Sumando1` cargaría en el registro AX la palabra de menor peso de `Sumando1`.

4. Programa que suma dos números enteros de 64 bits sin signo situados en memoria y almacena el resultado en memoria.

5. Programa que resta dos números de 16 bits situados en memoria y almacena el resultado en memoria.

6. Programa que resta dos números enteros de 64 bits sin signo situados en memoria y almacena el resultado en memoria.

7. Programa que multiplica un número de 8 bits sin signo por otro número del mismo tipo, almacenados ambos en la memoria. El resultado se almacena en la memoria.

8. Programa que multiplica un número de 16 bits sin signo por otro número del mismo tipo, almacenados ambos en la memoria. El resultado se almacena en la memoria.

9. Programa que divide un número de 8 bits sin signo por otro número del mismo tipo, almacenados ambos en la memoria. El cociente y el resto se almacenan por separado en dos posiciones de memoria.

10. Programa que divide un número de 16 bits sin signo por otro número del mismo tipo, almacenados ambos en la memoria. El cociente y el resto se almacenan por separado en dos posiciones de memoria.

2. Instrucciones de manejo de bits y de transferencia de control. Direccionamiento relativo.

11. Programa que multiplica cada uno de los datos de una tabla de 16 números enteros de 8 bits sin signo por un número de 8 bits del mismo tipo almacenado todo ello en memoria. El resultado se almacena en una tabla de 16 elementos de 16 bits. Condición: se tienen que utilizar los registros base BX e índice, SI y DI, para acceder a los datos de la tabla de forma relativa.

12. Programa que divide los 16 elementos de una tabla de números enteros de 8 bits sin signo por los elementos de otra tabla de 16 elementos del mismo tipo. El resultado se almacena en otra tabla de la misma cantidad de elementos situada en la memoria.

13. Programa que devuelve el valor máximo y mínimo en las direcciones de memoria `Max` y `Min` respectivamente de una tabla de 8 números de 16 bits sin signo almacenada en memoria que comienza en la dirección `Datos`.

14. Modificar el programa anterior para que trabaje con números con signo.

15. Programa que haga las operaciones fundamentales (suma, resta, multiplicación y división) con números enteros sin signo. Los operandos de 32 bits se encuentran en las direcciones `Ope1` y `Ope2`, el operador ('+', '–', '*','/') de 8 bits se encuentra en la dirección `Oper`. En multiplicaciones y divisiones `Ope1` y `Ope2` deben ser de 16 bits (se utiliza solo la parte baja de los datos como operandos). El resultado se almacena en la dirección `Resul` de 32 bits. Si la operación resulta errónea o fuera de rango (operador erróneo, acarreos, etc.) la variable situada en memoria denominada Error, inicialmente a cero, se deberá poner a -1. El programa dispondra de una estructura que decida la operación a realizar según el contenido de la dirección `Oper`. Se utilizará el comando E del CODEVIEW para cambiar el operador almacenado en `Oper`.

16. Programa que convierte un número binario de 8 bits almacenado en la dirección `NumBin` en su equivalente en código Gray y lo almacene en la dirección `NumGray`.

El código Gray es un código de error mínimo que se utiliza en el control de articulaciones de robots y en la construcción de *encoders*. Su principal característica es que sólo cambia un bit entre números consecutivos. Esto lo hace muy útil en aquellos sistemas que puedan tener estados de ambigüedad al cambiar dos o más bits a la vez. Mientras que unos bits pasan a cero, los otros pasan a uno. Cuando una posición pasa de 0111 a 1000 (de 7 a 8), el dispositivo conectado puede recibir datos falsos según cómo interprete los estados intermedios de 0 a 1 y 1 a 0, especialmente si los tiempos de transición entre niveles son distintos.

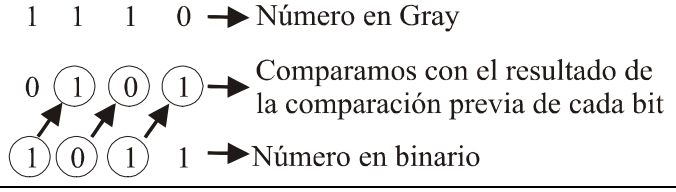
DECIMAL	GRAY
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

Un procedimiento para convertir un numero binario a código Gray consiste en desplazar un bit a la derecha el número binario rellenando el hueco resultante con un cero y a continuación se comparan bit a bit el numero original y el resultante del desplazamiento. Si los bits son iguales en su lugar se pone un cero y si son distintos se pone un uno.

Por ejemplo: convertir el número binario 1011 a Gray

1011 ; Binario
0101 ; Desplazamos 1 bit a la derecha y rellenamos con cero
------ ; Comparamos dígitos, si son iguales ponemos 0 y si no 1
1110 ; Gray

17. Programa que haga la conversión opuesta a la del programa anterior. Un procedimiento consiste en comparar el bit de mayor peso con cero, si son iguales se pone el bit de mayor peso del número binario resultante a cero y si son distintos se pone a uno. El siguiente bit del número codificado en Gray se compara con el bit del resultado anterior y siguiendo la misma norma, si son iguales el bit del resultado se pone a cero y si son distintos se pone a uno. Así sucesivamente hasta recorrer todos los bits del número codificado en Gray. Ejemplo de conversión del número codificado en Gray 1110:



18. Programa que analiza un dato de 16 bits situado en memoria para contar los bits puestos a uno que contiene. El resultado se almacena en la memoria.

19. Programa que a partir de los datos contenidos en una tabla de 10 números enteros de un byte, distintos de cero, cuente los datos positivos, los negativos, los pares y los impares. Los resultados se almacenan en la memoria.

20. Programa que analiza un dato de 16 bits, cuyos 15 bits de menor peso corresponden al dato, estando reservado el bit de mayor peso para el bit de paridad. El programa genera el bit de paridad y lo incorpora como bit de mayor peso al dato. La paridad consiste en contar los bits a uno del dato y generar un bit que es cero si la cantidad de unos es par o uno si la cantidad de unos es impar.

21. Se pretende realizar la transmisión de una serie de caracteres de 8 bits a un dispositivo lejano. Para evitar posibles errores en la transmisión, se obtendrá el bit de paridad de cada carácter. Como el dispositivo de transmisión sólo nos permite transmitir bytes, los bits de paridad se agruparán de ocho en ocho formando bytes con objeto de mejorar la eficiencia de la transmisión. De esta manera, por cada ocho caracteres que enviemos, enviaremos un byte de paridades que, además, se mandará duplicado para facilitar la detección de los errores, en este dato, por parte del receptor.

Nuestro programa debe codificar una tabla de 40 caracteres intercalando adecuadamente los bytes de paridad, así pues deberemos crear una nueva tabla que es la que en realidad se enviará.

Ejemplo:

```
Tabla1 DB 45h,27h,18h,0E5h,1Eh,2Fh,3Eh,10h,05h,3Dh,34h,...

Tabla2 DB 45h,27h,18h,0E5h,1Eh,2Fh,3Eh,10h,97h,97h,05h,3Dh,34h,...
```

;Tabla de 40 caracteres.
Tabla transmitida de 50 caracteres donde 97h es el byte formado con los bit de paridad de los ocho primeros bytes.

NOTA: Para la detección de paridad **NO** se pueden utilizar los saltos condicionados JP, JPE, JPO, JNP.

22. Programa que recoge un byte de la memoria e invierte el orden de los bits.

Ejemplos: Dato: 10110011 --> Resultado: 11001101
Dato: 01101011 --> Resultado: 11010110

3. Interrupciones de la BIOS y del DOS.

23. Programa que convierte un número de 16 bits, almacenado en memoria en la dirección `NumBin`, en su equivalente codificado en ASCII y lo almacena en la dirección `NumASCII`. Por ejemplo, si `NumBin` = 0D25Ah , cuyo valor decimal es 53850, entonces `NumASCII` = 35h,33h,38h,35h,30h.

Un carácter ASCII ocupa 1 byte en memoria. La codificación en ASCII de la representación decimal de un número se obtiene sumando 30h (48 decimal) a los dígitos que componen el número. Por ejemplo, el número 65535 codificado en ASCII es: 3**6** 3**5** 3**5** 3**3** 3**5** y ocupa 5 bytes de memoria. Para realizar la conversión de un número a su representación decimal codificada en ASCII, se divide sucesivamente el número por diez y se suma 30h a los restos de cada división, que corresponden a los dígitos decimales del número.

24. Programa que haga la conversión opuesta a la del programa anterior. El procedimiento más usual es obtener la representación decimal de cada dígito restando 30h (el carácter ASCII "0") a cada byte de la representación ASCII . A continuación se suman los dígitos empezando por el más significativo, multiplicando cada suma por diez.

Por ejemplo, si NumASCII = 31h,36h,34h,32h,33h que es el número 16423. La conversión sería:

31 - 30 = 1	→1x10=	10		
36 - 30 = 6	10+6=16→16x10=	160		
34 - 30 = 4	160+4=164→164x10=	1640		
32 - 30 = 2	1640+2=1642→ 1642x10=	16420		
33 - 30 = 3	16420+3=16423			

25. Programa que escribe en el centro de la pantalla un texto, utilizando las interrupciones de la BIOS. Para el modo de vídeo se definirá una constante, por ejemplo, `ModoVideo` EQU 3, que se cargará en el registro correspondiente en el programa.

El programa debe seleccionar el modo de video (normalmente el modo 03h) a continuación situar el cursor y, finalmente, imprimir la secuencia de caracteres uno a uno utilizando la función 0Eh.

26. Programa que escribe un texto en el centro de la pantalla utilizando las interrupciones del DOS. La selección del modo de vídeo y la situación del cursor se hace con la BIOS.

Se podría utilizar el servicio 02h que es similar al 0Eh de la interrupción 10h de la BIOS pero se facilitan las cosas si se utiliza la función 09h puesto que se encarga de sacar el mensaje completo. No hay que olvidar cerrar el mensaje con el símbolo $. Incluir al final del mensaje los caracteres ASCII de "avance de línea (LF)" (10 ó 0Ah) y "retorno de carro (CR)" (13 ó 0Dh) para ver su efecto.

27. Programa que lee un mensaje del teclado, compuesto por 6 caracteres, utilizando la interrupción 16h de la BIOS. Una vez leído se sacará por pantalla mostrando un mensaje previo de presentación.

Se pueden utilizar dos procedimientos distintos según el efecto deseado.

El primero consiste en usar el servicio 0h o el 10h, que espera hasta que se pulsa una tecla para efectuar la lectura, con lo cual el programa se detiene en la lectura de cada carácter (este servicio es similar al proporcionado por la función 01h de la interrupción 21h, con la diferencia que esta última saca el carácter por pantalla -tiene eco- al mismo tiempo que lo lee).

El segundo sistema consiste en leer el teclado sólo cuando se ha pulsado una tecla, para ello se utiliza el servicio1h que detecta la pulsación, activando el *flag* de cero si no hay ningún carácter en el buffer del teclado o desactiva el *flag* de cero si lo hay, momento en el que se puede efectuar la lectura mediante el servicio 0h o 10h mencionados anteriormente. De esta forma no se detiene la ejecución del programa.

28. Repetir el programa anterior utilizando la función 0Ah de la interrupción 21h del DOS. Una vez leído se sacará por pantalla mostrando un mensaje previo de presentación.

Para utilizar esta función hay que reservar un espacio en memoria para el buffer del teclado indicando en el primer byte de dicho buffer el número máximo de caracteres que se prevé leer más uno, para incluir el *enter*, en el segundo byte la función nos devuelve el número real de caracteres leídos, sin incluir el *enter*, y a partir del tercer byte están los caracteres incluyendo al final de la cadena el carácter *enter* (código ASCII 13).

Con la instrucción LEA se almacena en DX el desplazamiento del *buffer* del teclado, por ejemplo:

```
lea dx , Buffer
```

También se puede utilizar la directiva OFFSET, por ejemplo:

```
mov dx , OFFSET Buffer
```

El formato más útil del *buffer* de datos para utilizar esta función es:

```
Buffer      DB   NMaxCar          ;NMaxCar : número máximo de caracteres a leer + enter
Carleados   DB   ?                ;El DOS devuelve aquí la cantidad de caracteres leídos
Caracteres  DB   NMaxCar DUP(?)   ;Cadena de caracteres leída
```

Comprueba lo que ocurre cuando se introducen más y menos datos de los especificados.

29. Programa que lee hasta 10 caracteres, como máxmio, del teclado sin escribirlos en la pantalla. Para confirmar la entrada del carácter leído se contesta presentando un carácter específico, por ejemplo un asterisco, en la pantalla. Se trata de utilizar la función o servicio 08h de la interrupción 21h, que permite leer caracteres del teclado sin eco en la pantalla. Una vez leídos, el programa mostrará un mensaje de presentación y mostrará dichos caracteres.

30. Programa que espera la pulsación de una tecla para mostrarla con el mensaje 'La tecla pulsada es: '. El programa repite la pregunta hasta que se pulsa la barra de espacio y termina con el mensaje 'Adiós' en la pantalla.

31. Programa que lee una cadena de 15 caracteres como máximo y los saca por pantalla, en la línea siguiente, y en orden inverso.

32. Programa que solicita una cadena de texto de 20 caracteres como máximo, y la muestra en diagonal, comenzando en la esquina superior izquierda

33. Programa que convierte una secuencia de 8 letras introducidas por teclado en mayúsculas / minúsculas en función de cómo entran. El programa pide la palabra y después muestra el resultado.

34. Programa que pide un texto y escribe en la pantalla el número de palabras que contiene. Se limitará la longitud del texto a 254 caracteres.

35. Programa que devuelve la representación hexadecimal de un número de 16 bits expresado en decimal. El programa pide un número en representación decimal, responde con su código hexadecimal y pregunta si se quiere convertir otro número. El proceso se reitera hasta que se responda negativamente a la pregunta de continuar.

36. Programa que pide introducir por el teclado un número decimal de 16 bits, realiza el complemento a 2 de dicho número y lo almacena en la variable `Compa2` y finalmente ponerlo en exceso a 2^{15} y almacenarlo en la variable `Exceso`. No es necesario presentar en la pantalla los resultados.

4. Procedimientos y macros.

37. Programa que calcula el factorial de un número. El programa pedirá y leerá del teclado un número entre 0 y 9 y calculará su factorial. Si se introduce un número fuera de rango se mostrará un mensaje de aviso y se pedirá de nuevo un número. El programa presentará el resultado en la pantalla y preguntará si se quiere realizar un nuevo cálculo.

38. Programa que busca la existencia de un patrón en una cadena. El programa leerá de teclado una primera cadena de caracteres. A continuación, leerá (también por teclado) una segunda cadena de caracteres que servirá como patrón. El resultado, en caso de que el patrón se halle presente en la cadena, será mostrar la primera cadena con el patrón resaltado en video inverso. (Si hubiera varias ocurrencias del patrón, se resaltarán todas). En caso de que el patrón no se encuentre presente en la cadena, se mostrará un mensaje que lo indique.

39. Programa que muestre en un lugar aleatorio de la pantalla la hora, los minutos y los segundos del sistema, con el formato siguiente: HH:MM:SS. La posición de presentación aleatoria cambiará cada dos segundos. El programa detendrá la ejecución al pulsar la barra espaciadora.

La función 2Ch de la interrupción 21h del MS-DOS devuelve cada vez que se la llama la hora actual y su formato es el siguiente:

Entrada: AH = 2Ch Salida: CH = Hora (0...23)
CL = Minutos (0...59)
DH = Segundos (0...59)
DL = Centésimas de segundo (0...99)

Para la conversión a ASCII se utilizará un **procedimiento** al que se le pasarán los parámetros a través de la pila.

40. Programa que pide el nombre de pila del usuario y lo muestra desplazándose continuamente por el centro de la pantalla de arriba abajo o lateralmente, apareciendo por el centro del lateral izquierdo de la pantalla y saliendo por el lateral derecho. Cuando se pulse la barra de espacio alternará la dirección del desplazamiento, de arriba abajo o de izquierda a derecha. El programa termina al pulsar cualquier tecla diferente de la barra de espacio.

41. Programa que solicite dos números en hexadecimal de dos dígitos como máximo cada uno, y que luego presente el menú siguiente:

1. Sumar A + B
2. Restar A - B
3. Multiplicar A x B
4. Dividir A / B
0. Salir

Introduzca una opción: 1, 2, 3, 4 ó 0:

En función de la opción escogida, deberá presentar el resultado de la operación en hexadecimal en pantalla. Si el usuario escoge un carácter diferente a las opciones, el programa deberá emitir un pitido y presentar nuevamente el menú. Si la operación elegida es la división, deberá proporcionar el cociente y el resto.

---

42. Programa que convierte números en binario, de 16 bits como máximo, en su correspondiente representación decimal y viceversa. El programa presentará un menú de selección del tipo de conversión:

1. Binario a Decimal;
2. Decimal a Binario;
3. Salir.

Cuando se pida un número en binario admitirá cualquier cantidad de ceros y unos inferior o igual a 16. Cuando se pida un número decimal admitirá cualquier número en el rango de 16 bits sin signo. En cualquier caso se comprobará la entrada de caracteres no válidos emitiendo un mensaje de aviso y volviendo al menú inicial. Finalmente, mostrará el resultado y cuando se pulse cualquier tecla mostrará el menú inicial.

43. Programa que muestre en la pantalla en modo texto (80 x 25) un rectángulo definido por sus vértices superior izquierdo e inferior derecho cuyas coordenadas vendrán dadas en filas y columnas. El programa tiene que pedir las coordenadas de los dos vértices y a continuación mostrar el rectángulo y en la fila inferior de la pantalla mostrar un mensaje preguntando si se quiere continuar o salir. Se sugiere el uso de los ASCII 218 '┌', 217 '┘', 192 '└', 191 '┐', 179 '│' y 196 '─'.

44. Programa que muestra un carácter en movimiento (por ejemplo el carácter ASCII 219, '█') por la pantalla, partiendo del centro y siguiendo el movimiento marcado por las teclas del cursor del teclado extendido: flecha arriba (código identificación de tecla: 48h), flecha abajo (50h), flecha izquierda (4Bh) y flecha derecha (4Dh). Cuando se mueva el carácter debe dejar un "rastro" en la pantalla (para simular el "rastro" se imprime el carácter █ y en las dos posiciones previas se imprimen el carácter ▓ y el carácter ▒; cuando se deja de pulsar la tecla se detendrá el movimiento imprimiendo únicamente el carácter █). El programa termina cuando se pulsa la tecla ESC (ASCII:1Bh).

45. Programa para trazar dos ejes de coordenadas en un punto de la pantalla en modo texto, tanto en modo 3 (texto 16 colores, 80x25) como en modo 1 (texto 16 colores, 40x25), utilizando los símbolos ASCII de trazado de líneas y recuadros: 179 '│' , 196 '─' y 197 '┼' . El programa debe pedir el modo de vídeo que se desea y las coordenadas del punto para trazar los ejes. Una vez trazados los ejes cualquier pulsación de una tecla devolverá la pantalla al modo de video inicial y mostrará el menú de petición de datos. En el menú de petición de datos inicial se incluirá una opción para salir del programa.

46. Programa que emite un sonido durante dos segundos al pulsar las teclas A...L de la fila central del teclado, según la tecla pulsada la frecuencia del sonido es diferente: la tecla A emite un sonido de 400 Hz, la tecla S, (2*400 =) 800 Hz, la tecla D, (3*400 =) 1200 Hz y así sucesivamente hasta la tecla L (9*400=) 3600 Hz. Al iniciar el programa se presenta un mensaje recordatorio de las teclas utilizables. El programa termina cuando se pulsa la barra de espacio.

El PC dispone de un temporizador interno que genera 18,2 pulsos por segundo aproximadamente, en concreto genera una señal de frecuencia 1 193 180 Hz. Este temporizador dispone de un canal para controlar el altavoz que permite generar una señal de frecuencia programable. Además el altavoz se puede activar y desactivar a través de un puerto de control.

Para hacer que el altavoz emita un sonido de una frecuencia determinada durante un tiempo hay que seguir los siguientes pasos:

1. Cargar el periodo de la frecuencia deseada en el temporizador:

```
mov al,0B6h          ;prepara el temporizador para recibir
out 43h,al           ;el valor del periodo (contador = 1193180 / frecuencia)
mov ax,periodo
out 42h,al           ;parte baja del periodo al puerto 42h
xchg al,ah
out 42h,al           ;parte alta del periodo al puerto 42h
```

2. Activar el altavoz a través del puerto 61h. Se lee el puerto y se modifican los bits 0 y 1 poniéndolos a 1.
3. Se pone en marcha el retardo de 2 segundos. Se puede utilizar la función 2Ch de la interrupción 21h, descrita en el programa 36, para contar los segundos.
4. Desactivar el altavoz poniendo a cero los bits 0 y 1 del puerto 61h.

Resumen de funciones de la BIOS INT 10h

Registro AH	Descripción de la función
00h	Establece el modo de vídeo. AL = 0 40x25(1) Texto 16 colores AL = 2 80x25(1) Texto 16 colores AL = 1 40x25 Texto 16 colores AL = 3 80x25 Texto 16 colores (1) Sin ráfaga de color. Sin efecto en monitores RGB
01h	Tipo de cursor en modo texto. Selecciona líneas inicial y final del parpadeo del cursor. CH (bits 0 a 4) = Línea inicial CL (bits 0 a 4) = Línea final Bits 5 a 7 de los dos registros a cero. Si CX = 2000h desactiva el cursor.
02h	Sitúa el cursor en una posición de la pantalla empleando coordenadas de texto. DH = Fila (coordenada y : 0...24) DL = Columna (coordenada x: 0...39/79 según modo vídeo) BH = Página (0...3 Modos de vídeo 0 y 1, 0...7 Modos de vídeo 2 y 3)
03h	Obtiene la posición actual del cursor en modo texto. Llamada con BH = Página Devuelve: DH = Fila DL = Columna CL = Línea inicial cursor CH = Línea final cursor
05h	Selecciona la página activa para ser visualizada en la pantalla. AL = Página (0...3 Modos de vídeo 0 y 1, 0...7 Modos de vídeo 2 y 3)
06h	Desplazar (*scroll*) hacia arriba una ventana en la pantalla. AL = Número de líneas a desplazar. SI AL = 0 SE BORRA LA VENTANA. CH = Fila esquina superior izquierda CL = Columna esquina superior izquierda DH = Fila esquina inferior derecha DL = Columna esquina inferior derecha BH = Atributo a usar en área borrada con caracteres blancos. (p.ej.: 07h=blanco sobre negro).
07h	Desplazar (*scroll*) hacia abajo una ventana en la pantalla. AL = Número de líneas a desplazar. SI AL = 0 SE BORRA LA VENTANA. CH = Fila esquina superior izquierda CL = Columna esquina superior izquierda DH = Fila esquina inferior derecha DL = Columna esquina inferior derecha BH = Atributo a usar en área borrada con caracteres blancos. (p.ej.: 07h=blanco sobre negro).
08h	Leer carácter y atributo en la posición actual del cursor. Llamada: BH = Página. Devuelve: AL = ASCII del carácter leído AH = Atributo (p.ej.: 07h=normal, 0Fh=alta intensidad, 01h=vídeo inverso)
09h	Escribir carácter y atributo en la posición actual del cursor. AL = ASCII del carácter BH = Página. BL = Atributo (p.ej.: 07h=normal, 0Fh=alta intensidad, 01h=vídeo inverso) CX = Número de caracteres a escribir (factor de repetición)
0Ah	Escribir carácter en la posición actual del cursor con el atributo que tenía el anterior. AL = ASCII del carácter BH = Página. BL = Color (modo gráficos) CX = Número de caracteres a escribir (factor de repetición)
0Eh	Escribir un carácter en modo teletipo (escribe carácter y avanza cursor a la columna siguiente). AL = ASCII del carácter a escribir BH = Página BL = Color (modo gráficos)
0Fh	Leer modo de vídeo actual. Devuelve: AL = Modo de vídeo (ver función 00h) AH = Anchura de pantalla en columnas BH = Página activa

Resumen de funciones de la BIOS INT 16h

Registro AH	Descripción de la función
00h 10h(1)	Leer un carácter del *buffer* del teclado. Si el *buffer* está vacío espera a que se pulse una tecla. AL = ASCII del carácter leído AH = Código de identificación de tecla
01h 11h(1)	Devuelve el estado del *buffer* del teclado. Devuelve: Si el *buffer* está vacío pone el flag de cero a uno ZF = 1. Si hay tecla esperando en el *buffer* pone ZF = 0, entonces: AL = ASCII del carácter AH = Código de identificación de tecla
(1) En teclados expandidos (teclas F1 a F12, control de cursor, etc.)	

Resumen de funciones del DOS INT 21h

Registro AH	Descripción de la función
01h	Lectura de un carácter del teclado con eco en pantalla. Espera la pulsación de una tecla. Devuelve: AL = ASCII del carácter leído.
02h	Escribe un carácter en la pantalla. Llamada: DL = ASCII del carácter a escribir.
07h	Lectura directa de un carácter sin escribir en pantalla (sin eco). Lee el carácter del teclado si está disponible. Devuelve: AL = ASCII del carácter leído.
08h	Lectura de un carácter sin escribir en pantalla (sin eco). Espera la entrada de un carácter del teclado. Devuelve: AL = ASCII del carácter leído.
09h	Escribe una cadena de caracteres en la pantalla. La cadena debe terminar en el carácter $. La dirección efectiva de la cadena se carga en DX P.ej.: si se declara Cadena DB 'Texto$', entonces se carga en DX la dirección efectiva con: *lea dx,Cadena* o con *mov dx,OFFSET Cadena*. Asume como registro de segmento DS .
0Ah	Lectura de una cadena de caracteres del teclado con *buffer*. La lectura de la cadena se termina al pulsar *Intro* (Retorno de carro ASCII = 13). Se tiene que declarar el *buffer* como una tabla de bytes, inicializando el primer byte con el número máximo de caracteres a leer incluido el *Intro*. La función devuelve en el segundo byte del *buffer* el número real de caracteres leídos (no incluye el *Intro*) y a partir del tercer byte los códigos ASCII de la cadena incluido el *Intro*. La dirección efectiva del *buffer* se carga en DX Asume como registro de segmento DS . El tamaño del *buffer* puede ser de 255 bytes como máximo.
4Ch	Terminar un programa y volver al DOS. En AL se puede poner el código de retorno deseado.

Atributos para imprimir caracteres con colores, video inverso, etc. con la función 09h de la int 10h:

Byte de atributo para los modos 0, 1, 2 y 3 de vídeo								
Bit	7	6	5	4	3	2	1	0
	P	Fondo			I	Primer plano		

P = Parpadeo o intensidad de fondo (por defecto = parpadeo)

I = Intensidad de primer plano o selección de caracteres (por defecto = intensidad)

Valores por defecto para programar los colores y controlar los bits P o I:

Negro	Azul	Verde	Cyan	Rojo	Magenta	Marrón	Blanco	Gris	Azul pálido	Verde pálido	Cyan pálido	Rojo pálido	Magenta pálido	Amarillo	Blanco intenso
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Tabla de códigos ASCII

Caracteres No Imprimibles (de control)				Caracteres Imprimibles									
Nombre	Dec	Hex	Car	Dec	Hex	Car	Dec	Hex	Car	Dec	Hex	Car
nulo	0	00	NUL	32	20	espacio	64	40	@	96	60	`
comienzo cabecera	1	01	SOH	33	21	!	65	41	A	97	61	a
comienzo texto	2	02	STX	34	22	"	66	42	B	98	62	b
fin texto	3	03	ETX	35	23	#	67	43	C	99	63	c
fin transmisión	4	04	EOT	36	24	$	68	44	D	100	64	d
pregunta	5	05	ENQ	37	25	%	69	45	E	101	65	e
respuesta	6	06	ACK	38	26	&	70	46	F	102	66	f
sonido	7	07	BEL	39	27	'	71	47	G	103	67	g
retroceso	8	08	BS	40	28	(	72	48	H	104	68	h
tab. horizontal	9	09	HT	41	29	)	73	49	I	105	69	i
avance línea	10	0A	LF	42	2A	*	74	4A	J	106	6A	j
tab. vertical	11	0B	VT	43	2B	+	75	4B	K	107	6B	k
avance página	12	0C	FF	44	2C	,	76	4C	L	108	6C	l
retorno carro	13	0D	CR	45	2D	-	77	4D	M	109	6D	m
shift inactivo	14	0E	SO	46	2E	.	78	4E	N	110	6E	n
shift activo	15	0F	SI	47	2F	/	79	4F	O	111	6F	o
susp. enlace datos	16	10	DLE	48	30	0	80	50	P	112	70	p
control disp. 1	17	11	DC1	49	31	1	81	51	P	113	71	q
control disp. 2	18	12	DC2	50	32	2	82	52	R	114	72	r
control disp. 3	19	13	DC3	51	33	3	83	53	S	115	73	s
control disp. 4	20	14	DC4	52	34	4	84	54	T	116	74	t
respuesta neg.	21	15	NAK	53	35	5	85	55	U	117	75	u
sincronismo idle	22	16	SYN	54	36	6	86	56	V	118	76	v
fin bloque transm.	23	17	ETB	55	37	7	87	57	W	119	77	w
cancelar	24	18	CAN	56	38	8	88	58	X	120	78	x
fin de medio	25	19	EM	57	39	9	89	59	Y	121	79	y
sustituir	26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
escape	27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
separador fichero	28	1C	FS	60	3C	<	92	5C	\	124	7C	|
separador grupo	29	1D	GS	61	3D	=	93	5D	]	125	7D	}
separador registro	30	1E	RS	62	3E	>	94	5E	^	126	7E	~
separador unidad	31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Caracteres Imprimibles (ASCII Extendido)												
----Dec--------	----Hex--------	----Car----	----Dec--------	----Hex--------	----Car----	----Dec--------	----Hex--------	----Car----	----Dec--------	----Hex--------	----Car----
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	β
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	Ø
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	•
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	η
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

SUGERENCIAS PARA LA INSTALACIÓN DEL PROGRAMA MASM

1. Descomprimir el fichero descargado en un directorio creado al efecto (por ejemplo, C:\MASM51)
2. Crear un acceso directo en el escritorio del Procesador de comandos de Windows (es el icono denominado Símbolo del sistema, situado en la entrada Incio → Todos los Porgramas → Accesorios).
3. Sobre el icono del acceso directo creado, hacer clic con el botón derecho del ratón y seleccionar Propiedades. Hacer clic en la pestaña Acceso directo. En la entrada Iniciar en: escribir la ruta del directorio donde se ha instalado el programa, por ejemplo C:\MASM51. Botón Aceptar.
4. Abri el Panel de control → Categoría Rendimiento y mantenimiento → Icono Sistema → → Pestaña: Opciones avanzadas → Botón: Variables de entorno
5. En la ventana de Variables de entorno, situarse en la zona de Variables del sistema , localizar la variable PATH y modificarla: PATH → Botón Modificar → situarse al final de la línea, añadir un punto y coma y la ruta donde se encuentra la carpeta del programa MASM, por ejemplo: ;C:\MASM51\BIN
6. En la misma ventana de Variables de entorno añadir dos variables del sistema nuevas:
   Botón Nueva → Nombre de la variable MASM → Valor : /Zi /L
   Botón Nueva → Nombre de la variable LINK → Valor : /Co
7. Aceptar para terminar.

NOTA: Se puede optar por no modificar ninguna variable del sistema operativo, realizando sólo el apartado 1 (descomprimir el fichero en un directorio). En ese caso, cada vez que se quiera trabajar con el ensamblador, hay que iniciar una sesión del Procesador de comandos de Windows (icono Símbolo del sistema) y una vez en ella, situarse en el directorio donde se ha descomprimido dicho programa. Cuando se esamble un programa hay que ejecutar el ensamblador con los parámetros /Zi /L (por ejemplo: MASM /Zi /L progr.asm) y cuando se enlace, ejecutar el enlazador con el parámetro /Co (por ejemplo: LINK /Co).