LABORATORIO DE PROGRAMACIÓN EN LENGUAJE ENSAMBLADOR x86-16bits

Interrupciones del sistema. Servicios de entrada/salida de caracteres (INT 21h)

Objetivo

El objetivo de esta práctica es conocer y aprender a utilizar los servicios de entrada/salida de caracteres que ofrece la interrupción del sistema operativo INT 21h. Estos servicios toman como entrada el teclado y como salida la pantalla. Existen servicios diferenciados para trabajar sobre caracteres individuales o sobre cadenas de caracteres.

Funciones de entrada de caracteres de la INT 21h

La entrada se realiza desde el teclado. Disponemos de varios servicios para hacer la lectura de un carácter. Se diferencian en función de si la entrada es bloqueante o no y de si se atiende o no la combinación de teclas *Ctrl-Break*.

Entrada bloqueante

Una entrada/salida es bloqueante si el proceso principal se detiene hasta que se satisfaga alguna condición (normalmente hasta que los datos se hayan leído o escrito). Será no bloqueante si el proceso principal no se detiene.

En el caso que nos ocupa, un servicio de lectura de carácter es bloqueante si espera hasta que se lea algún carácter. Será no bloqueante si lee el *buffer* de teclado devolviendo o un código de carácter o 0 si no hay carácter.

El uso de un servicio bloqueante o no bloqueante depende de lo que necesite el programador en cada caso. En ocasiones será conveniente detener la ejecución del programa en tanto no haya un carácter de entrada mientras que en otros momentos bastará con comprobar si hay o no un carácter disponible.

Combinación *Ctrl-Break*

La combinación de teclas *Ctrl-Break* (*Ctrl-C*) se puede usar para invocar una interrupción especial conocida como *break* de teclado (interrupción BIOS INT 1Bh) que sirve para terminar el proceso en curso. La rutina de atención a la INT 1Bh por defecto sólo contiene una instrucción IRET, es decir, que no realiza ninguna tarea. Sin embargo, se puede cambiar dicha rutina para que realice otra tarea. Existen servicios de entrada de carácter que atienden la interrupción de *break* de teclado y otros que no.

Servicio 01h – entrada bloqueante con eco y atención a *Ctrl-Break*

El servicio 01h de la INT 21h espera a que se pulse una tecla y la escribe en la pantalla (lectura con eco) comprobando si es *Ctrl-Break* en cuyo caso dispara la interrupción de *break* de teclado que suele finalizar el programa en curso. El código ASCII del carácter se devuelve en el registro AL.

argumentos: ah = 01h **valores devueltos**: al = código ASCII leído

Servicio 06h – DL = FFH – entrada no bloqueante sin eco y omisión de *Ctrl-Break*

El servicio 06h de la INT 21h puede ser de entrada o salida. Si DL es igual a FFh, lee un carácter del *buffer* del teclado sin eco y omitiendo un posible *Ctrl-Break*.

argumentos	**valores devueltos**
ah = 06h	zf = 0 si hay carácter y 1 si no hay carácter disponible
dl = FFh (lectura de consola)	al = código ASCII leído

Servicio 07h – entrada bloqueante sin eco y omisión de *Ctrl-Break*

El servicio 07h de la INT 21h espera a que se pulse una tecla sin eco y omitiendo un posible *Ctrl-Break*. El código ASCII del carácter se devuelve en el registro AL.

argumentos: ah = 07h **valores devueltos**: al = código ASCII leído

Servicio 08h – entrada bloqueante sin eco y atención a *Ctrl-Break*

El servicio 08h de la INT 21h espera a que se pulse una tecla sin eco y comprobando si es *Ctrl-Break* en cuyo caso dispara la interrupción de *break* de teclado. El código ASCII del carácter se devuelve en el registro AL.

argumentos: ah = 08h **valores devueltos**: al = código ASCII leído

Servicio 0Bh – lee el estado del teclado con atención a *Ctrl-Break*

El servicio 0Bh lee el estado del teclado con atención a *Ctrl-Break*. El registro AL devuelve el estado.

argumentos: ah = 0Bh **valores devueltos**: al = FFh si hay carácter; 0 en caso contrario

Servicio 0Ch – borra el *buffer* del teclado e invoca un servicio de entrada de carácter

El servicio 0Ch de la INT 21h borra el *buffer* del teclado e invoca un servicio de entrada de carácter de los explicados más arriba (01h, 06h, 07h, 08h) o de entrada de cadenas que se verá seguidamente (0Ah). El registro AL se utiliza para seleccionar el servicio a invocar y devuelve el valor propio de dicho servicio.

argumentos
ah = 0Ch
al = número de servicio a invocar

valores devueltos
al = valor devuelto por el servicio invocado

Funciones de salida de caracteres de la INT 21h

Como hemos indicado, la salida se realiza sobre la pantalla. Disponemos de un único servicio para escribir un carácter en la pantalla con la INT 21h. Es el siguiente:

Servicio 02h – salida de carácter

El servicio 02h presenta un carácter en pantalla. El registro DL contiene el ASCII del carácter a escribir:

argumentos
ah = 02h
dl = código ASCII del carácter a escribir

valores devueltos
al = código ASCII escrito

Servicio 06h – DL ≠ FFH – salida de carácter

Este servicio puede ser de entrada o salida. Si DL es distinto de FFh, escribe el carácter codificado en DL.

argumentos
ah = 06h
dl = código ASCII del carácter a escribir (≠FFh)

valores devueltos
al = código ASCII escrito

Funciones de entrada/salida de cadenas de caracteres de la INT 21h

Una cadena de caracteres (*string* en inglés) es una secuencia ordenada de longitud arbitraria de códigos alfanuméricos. El sistema de codificación puede ser ASCII, EBCDIC o UNICODE.

Físicamente las cadenas se pueden almacenar colocando cada código en posiciones consecutivas de memoria o enlazando carácter a carácter. El primer sistema es el más habitual y eficiente.

Para diferenciar una cadena de otra, podemos reservar espacio en memoria de tamaño fijo, pero este método es ineficiente ya que desaprovecha espacio de almacenamiento. Habitualmente, lo más eficiente es manejar espacios de almacenamiento de longitud variable utilizando alguno de estos métodos:

- indicando el comienzo y final de cada cadena mediante un carácter separador
- indicando la longitud de la cadena antes de la misma
- indicando el final de la cadena mediante un carácter terminador

En el caso de los servicios de entrada/salida de cadenas de la interrupción del sistema operativo INT 21h se utiliza el almacenamiento consecutivo en memoria de los códigos alfanuméricos y se permite la longitud arbitraria de la cadena gracias al carácter terminador '*enter*' para entrada de cadena y '$' para salida de cadena.

Servicio 0Ah – entrada de cadena de caracteres

El servicio 0Ah de la INT 21h lee una secuencia de caracteres desde el teclado y los almacena en memoria hasta que se pulse la tecla '*enter*'. El área de memoria es una estructura de datos con 3 campos: el primer byte es el tamaño máximo permitido de la cadena incluido el código de la tecla '*enter*', el segundo byte recibirá el número de caracteres leídos sin contar el '*enter*' y el tercer campo es el *buffer* de memoria en el que se almacenará la cadena incluyendo el 'enter'. Evidentemente, el tamaño del *buffer* deberá coincidir con el valor del primer campo. El puntero del área de memoria se pasa como parámetro en DS:DX.

argumentos	**valores devueltos**
ah = 0Ah	- número de caracteres leídos
ds:dx = puntero del área de memoria	- cadena

Ejemplo:

```
.data
cadena DB 11,?,11 DUP(?)  ;reserva para una cadena de 10 caracteres

.code
mov ax, @DATA
mov ds, ax


lea dx, cadena
mov ah, 0Ah
int 21h
```

Servicio 09h – salida de cadena de caracteres

El servicio 09h de la INT 21h escribe en la pantalla una cadena de caracteres almacenada en memoria y finalizada con el carácter '$'. El puntero del área de memoria se pasa como parámetro en DS:DX.

argumentos	**valores devueltos**
ah = 09h	al = 24h
ds:dx = puntero de la cadena	

Ejemplo:

```
.data
cadena DB 'Hola mundo$'

.code
mov ax, @DATA
mov ds, ax


lea dx, cadena
mov ah, 09h
int 21h
```

Funciones de entrada/salida de dispositivos de la INT 21h

El sistema operativo proporciona una serie de servicios a través de la INT 21h para hacer transferencias de caracteres con dispositivos que bien pueden ser ficheros o la consola. Por consola entendemos la entrada y salida estándar, es decir, la entrada por teclado o *stdin* y la salida por pantalla o *stdout*.

Estos servicios permiten crear, borrar, abrir, cerrar, leer, escribir y realizar otras acciones, sobre ficheros. En este momento sólo estamos interesados en la entrada/salida por consola:

- leer fichero o consola: servicio 3Fh
- escribir fichero o consola: servicio 40h

Estos servicios representan una alternativa a los de entrada/salida de cadenas ya que no utilizan carácter terminador en las cadenas sino que trabajan con el número de caracteres a transferir como argumento.

Servicio 3Fh – lectura de dispositivo

El servicio 3Fh de la INT 21h lee caracteres de un dispositivo. Si el dispositivo es un fichero, se le pasa el manejador del fichero (*handler*) y el número de caracteres a leer. Si el dispositivo es la consola, el número de manejador es el 1 y se lee el teclado hasta que aparezca el carácter CR (*Carriage Return*) al pulsar la tecla '*enter*'.

argumentos	**valores devueltos**
ah = 3Fh	cf = 0 si no hay error y 1 si se produce error
bx = manejador del dispositivo (1 si consola)	ax = número de caracteres leídos o código de error en su caso
cx = número de caracteres a leer	
ds:dx = puntero del área de memoria	

Servicio 40h – escritura en dispositivo

El servicio 40h de la INT 21h escribe un número de caracteres especificado como argumento en un dispositivo. Si el dispositivo es la consola, el número de manejador es el 1 y se escribe en pantalla. El puntero del área de memoria se pasa como parámetro en DS:DX.

argumentos
ah = 40h
bx = manejador del dispositivo (1 si consola)
cx = número de caracteres a escribir
ds:dx = puntero del área de memoria

valores devueltos
cf = 0 si no hay error y 1 si se produce error
ax = número de caracteres escritos o código de error en su caso

Prácticas

A) Realiza un programa que capture 4 caracteres numéricos y almacene el valor numérico en una variable de tipo *word*.

Los caracteres numéricos del 0 al 9 se codifican en ASCII del 30h al 39h (véase tabla ASCII más abajo). Esto hace que sea muy sencillo su detección y conversión a entero. Cada vez que no se introduzca un carácter numérico, el programa emitirá un mensaje de error y volverá a comenzar.

Para calcular el valor numérico, acumularemos los resultados parciales sobre el registro DX inicialmente a cero. Por cada carácter, multiplicamos DX por 10 y le sumamos la nueva cifra. Una vez finalizado la captura de las 4 cifras ABCD, tendremos:

DX = ((((DX * 10 + A) *10 + B) * 10) + C *10) + D

B) Realiza un programa que emita un mensaje por pantalla en función de la tecla que se pulse.

Programa un bucle que emita un mensaje en pantalla de manera indefinida. El mensaje cambiará en función de la tecla que se pulse. Si se pulsa ‘F’ el programa finaliza. La lectura del teclado será no bloqueante.

C) Realiza un programa lea 2 caracteres hexadecimales y almacene su valor entero en una variable de tipo byte. Sólo se hará eco de los caracteres hexadecimales.

La entrada se programará sin eco. Una vez validado el carácter leído, se hará eco. Si no corresponde a un carácter hexadecimal se seguirá esperando un nuevo carácter. Una vez almacenada la variable, se mostrará en pantalla tomándola como ASCII. Son caracteres hexadecimales: ‘0’ – ‘9’, ‘a’ – ‘f’ y ‘A’ – ‘F’.

Para obtener el valor numérico de un carácter hexadecimal codificado en mayúsculas ASCII basta con restar ‘A’ – 10, es decir, restarle su código ASCII y sumar su valor numérico en decimal.

D) Realiza un programa que pida una cadena de un máximo de 10 caracteres, convierta las mayúsculas a minúsculas y viceversa, y presente la nueva cadena en pantalla.

En la tabla ASCII vemos que los caracteres de las mayúsculas y de las minúsculas sólo se diferencias en el bit del peso 2^5. Basta con complementar dicho bit para que el carácter cambie de tamaño. Usaremos el servicio 0Ah para leer la cadena y el servicio 09h o el 40h para presentar la nueva cadena.

000d	00h		(nul)	016d	10h	►	(dle)	032d	20h	␣	048d	30h	0	064d	40h	@	080d	50h	P	096d	60h	‘	112d	70h	p
001d	01h	☺	(soh)	017d	11h	◄	(dc1)	033d	21h	!	049d	31h	1	065d	41h	A	081d	51h	Q	097d	61h	a	113d	71h	q
002d	02h	☻	(stx)	018d	12h	↕	(dc2)	034d	22h	"	050d	32h	2	066d	42h	B	082d	52h	R	098d	62h	b	114d	72h	r
003d	03h	♥	(etx)	019d	13h	‼	(dc3)	035d	23h	#	051d	33h	3	067d	43h	C	083d	53h	S	099d	63h	c	115d	73h	s
004d	04h	♦	(eot)	020d	14h	¶	(dc4)	036d	24h	$	052d	34h	4	068d	44h	D	084d	54h	T	100d	64h	d	116d	74h	t
005d	05h	♣	(enq)	021d	15h	§	(nak)	037d	25h	%	053d	35h	5	069d	45h	E	085d	55h	U	101d	65h	e	117d	75h	u
006d	06h	♠	(ack)	022d	16h	▬	(syn)	038d	26h	&	054d	36h	6	070d	46h	F	086d	56h	V	102d	66h	f	118d	76h	v
007d	07h	•	(bel)	023d	17h	↨	(etb)	039d	27h	'	055d	37h	7	071d	47h	G	087d	57h	W	103d	67h	g	119d	77h	w
008d	08h	◘	(bs)	024d	18h	↑	(can)	040d	28h	(	056d	38h	8	072d	48h	H	088d	58h	X	104d	68h	h	120d	78h	x
009d	09h		(tab)	025d	19h	↓	(em)	041d	29h	)	057d	39h	9	073d	49h	I	089d	59h	Y	105d	69h	i	121d	79h	y
010d	0Ah	◙	(lf)	026d	1Ah		(eof)	042d	2Ah	*	058d	3Ah	:	074d	4Ah	J	090d	5Ah	Z	106d	6Ah	j	122d	7Ah	z
011d	0Bh	♂	(vt)	027d	1Bh	←	(esc)	043d	2Bh	+	059d	3Bh	;	075d	4Bh	K	091d	5Bh	[	107d	6Bh	k	123d	7Bh	{
012d	0Ch		(np)	028d	1Ch	└	(fs)	044d	2Ch	’	060d	3Ch	<	076d	4Ch	L	092d	5Ch	\	108d	6Ch	l	124d	7Ch	|
013d	0Dh	♪	(cr)	029d	1Dh	↔	(gs)	045d	2Dh	-	061d	3Dh	=	077d	4Dh	M	093d	5Dh	]	109d	6Dh	m	125d	7Dh	}
014d	0Eh	♫	(so)	030d	1Eh	▲	(rs)	046d	2Eh	.	062d	3Eh	>	078d	4Eh	N	094d	5Eh	^	110d	6Eh	n	126d	7Eh	~
015d	0Fh	☼	(si)	031d	1Fh	▼	(us)	047d	2Fh	/	063d	3Fh	?	079d	4Fh	O	095d	5Fh	_	111d	6Fh	o	127d	7Fh	⌂