INTRODUCCIÓN

El interface de teclado del PC se ocupa de rastrear continuamente el estado de todas las teclas, para detectar si ha ocurrido algún cambio de estado en cualquiera de ellas. De ser así, determina si lo que ha ocurrido es una pulsación o una liberación de tecla, y que tecla ha sido la que ha cambiado.

Cada una de las teclas tiene asociado un numero diferente para que el controlador de teclado pueda reconocerlas, este numero tiene 7 bits y se llama *scancode* y depende únicamente de la posición que la tecla ocupa en le teclado.

Cuando una tecla ha sido pulsada o liberada, el interface de teclado guarda el scancode de ésta en una pequeña memoria interna. Además de esta memoria interna, el teclado tiene algunos registros, dos de ellos son accesibles desde el exterior :

Registro de Estado : Contiene información acerca del interface.

Registro de Datos : Si en la memoria interna del teclado hay pulsaciones registradas, en este

registro se guarda una copia del *scancode* correspondiente a la primera

que se pulsó.

En el IBM−PC y compatibles, la gestión de las interrupciones hardware las hace un circuito especializado llamado *PIC*. Este circuito recibe las peticiones de interrupción de todos los periféricos del sistema y se las pasa de una en una a la CPU, en la arquitectura PC, el interface de teclado está conectado a la línea llamada *IRQ1* del PIC y es identificado ante la CPU como el *vector 9.* Cada vez que se pulse o libere una tecla en el PC, se produce una interrupción hardware, con vector 9.

En definitiva existen dos módulos software encargados de la entrada por teclado, uno de ellos se encarga de leer una tecla, convertirla en ASCII y guardarla en un buffer en memoria; el otro se encarga de examinar el buffer y retornar su estado cada vez que el usuario lo solicite.

Ambas rutinas están implementadas el la BIOS del ordenador, la primera de ellas de activa mediante una interrupción hardware generada por el interface del teclado (int 9), que pasa a la CPU a través del PIC. La segunda es llamada por los programas de usuario mediante la ejecución de la interrupción software int 16h.

IMPLEMENTACIÓN

Se pretende escribir una versión propia de ambas rutinas, que sustituirán a las que suministra el BIOS, con ello tomaremos el control del teclado, sobre los códigos ASCII a generar por cada tecla y sobre las acciones que han de ocurrir ante la pulsación de teclas especiales.

PRÁCTICA 1 Sustituir int 16h

```
#include <stdio.h>

#include <dos.h>

#include <stdlib.h>
```

```
#define vector_16 0x16
void (_interrupt _far *antigua)(void);
void _interrupt _far new_interrup( unsigned es, unsigned ds,unsigned di,
unsigned si,unsigned bp, unsigned sp,unsigned bx, unsigned dx,
unsigned cx, unsigned ax )
{
/* Declaración de punteros para manejar el buffer circular y acceder al área de datos del */
/* BIOS. */
unsigned ah=ax;
char _far *p_comienzo = (char _far *)0x00400000;
char _far *p_cabeza = (char _far *)0x0040001a;
char _far *p_cola = (char _far *)0x0040001c;
ah=((ah) & (0xFF00));
if ( (ah==0x0000)||(ah==0x1000) )
/* Si ah no es 0 ni 10h, se está solicitando un servicio del BIOS, en este caso no hacemos */
/* nada y encadenamos con la rutina BIOS original. En otro caso estamos esperando */
/* por una tecla para retornar su valor. */
{
while (*p_cabeza==*p_cola)
{
}
ax=*(p_comienzo+*p_cabeza);
/* En el momento en que la cabeza y la cola del buffer sean diferentes, hay tecla pulsada . */
/* En ax guardamos el valor apuntado por cabeza, cabeza contiene solo el offset, la dirección */
/* se obtendrá con p_comienzo + p_cabeza */
*p_cabeza+=2;
```

```
if (*p_cabeza==0x003E)
*p_cabeza=0x001E;
ax++;
/* Incrementamos la cabeza en 2, teniendo en cuenta la circularidad, es decir, si cabeza */
/* llega al final debe volver a entrar por el principio del buffer circular. */
/* Incrementamos ax antes de retornar, de este modo, mientras se ejecuta COMMAND, */
/* el teclado aparecerá revuelto, pero al finalizar nuestro programa todo volverá a la */
/* normalidad */
}
else
_chain_intr (antigua);
}
main()
{
/* El programa principal hace : */
/* Salvar en la variable antigua el valor del vector 16h */
/* Modificar el vector 16h para que apunte a new_interrup" */
/* Llamar a la función system ("command") */
/* Restaurar el vector 16h a su valor inicial */
/* Terminar */
antigua= _dos_getvect(vector_16);
_dos_setvect(vector_16,new_interrup);
system ("\COMMAND.COM");
_dos_setvect(vector_16,antigua);
/* Recuérdese que para abandonar el COMMAND.COM es necesario teclear DWHS */
/* seguido de Control−L, lo que se traducirá en EXIT seguido de Control−M, que es el */
```

```
/* retorno de carro */

printf (" * VUELTA A LA NORMALIDAD *\n");

return 0;

}
```

PRÁCTICA 2 Sustituir int 9h

```
#include <stdio.h>

#include <dos.h>

#include <stdlib.h>

#define vector_9 0x9

char tabla[]={'?','?','1','2','3','4','5','6','7','8','9','0','?','?','?','?','q','w','e','r','t','y','u','i','o','p','?','?',13,'?',

'a','s','d','f','g','h','j','k','l','?','?','?','?','?','z','x','c','v','b','n','m','?','?','?','?','?','?',' ','?','?','?','?','?','?','?',

'?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','

/* Esta tabla referencia a cada tecla, en ella el índice del array es el scancode */

/* y el elemento de cada índice es el carácter ascii correspondiente a cada scancode */

/* Los ? son para las teclas distintas de letras y números o retornos de carro */

void (_interrupt _far *antigua)();

void _interrupt _far new_interrup()

{

/* Declaración de punteros para manejar el buffer circular y acceder al área de datos */

/* del BIOS. Declaramos la variable scancode y bit */

char _far *p_comienzo = (char _far *)0x00400000;

char _far *p_cabeza = (char _far *)0x0040001a;

char _far *p_cola = (char _far *)0x0040001c;

int scancode;

int bit_menos;
```

```
/* Miramos el bit inferior del puerto 64h del mapa de E/S (reg. de control), si es un 1 es que */

/* se ha pulsado una tecla, en este caso se mira el scancode del a tecla pulsada, éste estará */

/* en el puerto 60h del mapa de E/S (reg de datos) */

bit_menos=inp(0x64);

_asm {

cmp bit_menos,1

je etiqueta

}

etiqueta:

scancode=inp(0x60);

/* Para considerar sólo las pulsaciones de teclas e ignorar las teclas soltadas es necesario que */ /* scancode sea menor o igual a 128 */

if (scancode<=128)

{

. if (*p_cola!=*p_cabeza−2) /* Miramos que el buffer circular no esté lleno. */

{

*(p_comienzo+*p_cola)=tabla[scancode];

/* colocamos el ASCII (sacado de la tabla)que se corresponde con el scancode */

/* obtenido, en la posición a la que apunta p_cola . */

/* Nótese que p_cola es el offset, le dirección completa se obtendrá combinado */

/* p_cola y p_comienzo (segmento 0040h) */

*p_cola=*p_cola+2; /*incrementamos p_cola en dos unidades */

if (*p_cola==0x003E) *p_cola=0x001E;

/* si la cola llega al final, hacemos que apunte */

/* de nuevo al principio del buffer */

/* circularidad del buffer */

}
```

```
}
outp(0x20,0x20); /* Reinicializamos el PIC, enviamos el valor 20h al puerto 20h */
}
main()
{
/* El programa principal hace : */
/* Salvar en la variable antigua el valor del vector 9h */
/* Modificar el vector 9h para que apunte a new_interrup" */
/* Llamar a la función system ("command") */
/* Restaurar el vector 9h */
/* Terminar */
antigua= _dos_getvect(vector_9); /* guardamos al antigua rutina */
_dos_setvect(vector_9,new_interrup); /* el vector 9h apunta a la nueva_rutina */
system ("\COMMAND.COM"); /* Llamada a la función system (command)*/
_dos_setvect(vector_9,antigua); /* restauramos la rutina original */
/* Recuérdese que para abandonar el COMMAND.COM es necesario teclear */
/* EXIT+ENTER */
printf (" * VUELTA A LA NORMALIDAD *\n");
return 0;
}
```