Periféricos Avanzados

Práctica 1. Programación del teclado.

Introducción

Cuando pulsamos una tecla del teclado de un PC se genera un código cuando pulsamos la tecla y otro cuando la soltamos. La información del teclado se manda al controlador de teclado que reside en el sistema y se envía "serializado" , es decir, en serie. Cada byte de la pulsación de tecla se envía al controlador sincronizado con una señal de reloj que se encarga de proporcionar el mismo teclado.

Los teclados de los sistemas XT son muy diferentes (modos de comunicación, bidireccionalidad exclusiva en los AT....).

¿Qué ocurre en el sistema cada vez que pulsamos una tecla?

Cuando pulsamos una tecla, el procesador del teclado determina para esta tecla un "scan code" que es un código estrechamente relacionado con la posición de la tecla en la matriz mas que con el carácter que contiene. Cuando pulsamos una tecla, se manda con una señal de reloj el "scan code" de esa tecla al controlador de teclado y cuando este recibe toda la información de la tecla (8bits en el teclado XT o 11bits en el AT) entonces el sistema genera una INTERRUPCIÓN HARDWARE EN **IRQ1** y pasará a ejecutar la rutina de servicio del teclado.

Interrupciones y rutinas de servicio

Cuando el sistema recibe la interrupción hardware en IRQ1 (generada por el teclado) el sistema ejecuta un programa de búsqueda de interrupciones para saber qué interrupción exactamente ha generado la petición de servicio. Una vez que tiene el numero de la interrupción busca dentro de una tabla llamada Tabla de Vectores de Interrupción al vector que tiene el mismo numero que la interrupción causante de la petición de servicio, este vector apunta a una rutina de servicio . En nuestro caso el teclado al generar la interrupción , el sistema localiza el numero de interrupción involucrado y con ese numero se dirige a la tabla de vectores de interrupcion, localiza el que tenga el numero de la interrupcion en cuestion y ejecuta el programa que contenga este vector (lo que hemos denominado "rutina de servicio de teclado".

Rutina de servicio de teclado.

Una vez que hemos localizado el vector en cuestión, comenzamos a ejecutar la rutina de servicio del teclado, esta toma el código scan (o scan code) que posee el controlador a consecuencia de la pulsación de la tecla. Esta rutina usa el código scan o scan code para generar un código de dos bytes , este código lo coloca en un área de memoria RAM llamado buffer de teclado.

Las características del buffer de teclado a parte de que es un área de memoria RAM , son que tiene 32 bytes de RAM desde la dirección 41E hex hasta la dirección 43D hex. Este buffer es circular y los punteros que hacen referencia a la dirección de inicio del buffer se almacena en las direcciones 41A y 41B y el puntero a la dirección final se almacena en 41C y 41D.

La información que tienen estos dos bytes que el controlador de teclado deja en un area de la RAM contienen la siguiente información :

- El byte menos significativo : contiene el código ASCII.
- El byte más significativo : contiene el código SCAN.

Por ejemplo, si pulsamos la A :

Tecla pulsada	**Byte mas sign**	**Byte menos significativo**
Una A mayúscula :	1E hex	41 hex
Una a minúscula :	1E hex	61 hex
Un "ALT A" genera:	1E hex	00 hex (Caracter especial)

SECUENCIA DE EVENTOS RELACIONADA CON EL TECLADO

A modo de repaso y con la intención de esquematizar el comportamiento del teclado vamos a ver el orden en el que se suceden los eventos en una operación normal de teclado .

1.- El procesador de teclado monitoriza una y otra vez el teclado a la espera de la pulsación de una tecla. Cuando se pulsa una, este procesador obtiene el scan code de la tecla pulsada. Un Scan code es independiente de la letra impresa en la tecla (tened en cuenta que los teclados según el pais distribuyen las letras en el teclado de una manera u otra), asi que el scan code identifica la posición de la tecla pulsada.

2.- El procesador de teclado le pasa al controlador de teclado el codigo scan (scan code) de la tecla pulsada . Los datos se le pasan al PC en serie y en forma síncrona.

3.- Cuando el controlador de teclado recibe los 11 bits (en teclados AT) genera una interrupcion Hardware en el nivel 1 (IRQ1) para poder ejecutar la rutina de servicio de teclado.

4.- Cuando el procesador del sistema recibe la señal de IRQ1, comienza a buscar el numero de la interrupción que ha causado la interrupción.

5.- El sistema localiza con este número el vector de interrupcion adecuado (dentro de la tabla de vectores de interrupción) este vector nos da la dirección de inicio de la rutina de servicio del teclado (en nuestro caso).

6.- El procesador accede a esa dirección obtenida en 5 y comienza a ejecutar el programa que atiende a la interrupcion de teclado (rutina de servicio de teclado).

7.- La rutina de servicio de teclado toma el scan code del controlador de teclado .

8.- La rutina de servicio de teclado genera un codigo de dos bytes que almacena en un area RAM llamada bufer de teclado.

9.-La correspondencia entre scan code y código ASCII no siempre se da, sea el caso de la tecla SHIFT que al ser pulsada genera el scan code (make code) 42 y no tiene correspondencia ASCII. Es evidente que se hace necesario conocer el estado de la tecla SHIFT ya que combinada con cualquier otra tecla por ejemplo “a” generará un ASCII diferente ya sea para la letra minúscula a o para la letra mayúscula A. Si se pulsa la tecla “shift” el teclado genera un código llamado make code (que es el código scan de la tecla cuando se pulsa) y transfiere este código a la interfaz del teclado. Alli se genera una interrupción hardware 9 vía IRQ1 y así el controlador de teclado lee el scan code, el resultado será procesado de manera diferente en función de que la tecla shift esté o no pulsada., sólo el driver de teclado combinará los dos codigos scan (el 42 (de shfit) y el de la tecla correspondiente) para generar el código ASCII de la letra en minúsculas o de la letra en mayúsculas, es decir, que el driver generará el código ASCII correspondiente cuando compruebe los scan codes que le llegan. De aquí se desprende que algunas teclas como la tecla de mayúsculas o la de shift no se almacenan en el buffer de teclado por que no generan códigos ASCII o por que sólo sirven para obtener otros ASCII a partir de los que ya tenemos, el estado de estas teclas se pueden chequear leyendo de un área de memoria determinada en la que el estado de un conjunto de bits (status bits) indican si la tecla shift esta pulsada o no .... .

TABLA CON LOS CODIGOS SCAN (SCAN CODES)

Scan code	Base case	Upper case	Scan code	Base case	Upper case	Scan code	Base case	Upper case
29	`	~	**02**	1	!	**03**	2	@
04	3	#	**05**	4	$	**06**	5	%
07	6	^	**08**	7	&	**09**	8	*
0A	9	(	**0B**	0	)	**0C**	-	_
0D	=	+	**0E**	Backspace	Backspace	**0F**	Tab	Back Tab
10	q	Q	**11**	w	W	**12**	e	**E**
13	r	R	**14**	t	T	**15**	y	Y
16	u	U	**17**	i	I	**18**	o	O
19	p	P	**1A**	[	{	**1B**	]	}
2B	\	|	**3A** note 1	Caps Lock	na	**1E**	a	A
1F	s	S	**20**	d	D	**21**	f	F
22	g	G	**23**	h	H	**24**	j	J
25	k	K	**26**	l	L	**27**	;	:
28	'	"	**2B** note 2	#	~	**1C**	Enter	Enter
2A note 1	Left Shift	na	**D5** note 2	\	|	**2C**	z	Z
2D	x	X	**2E**	c	C	**2F**	v	V

30	b	B	**31**	n	N	**32**	m	M
33	,	<	**34**	.	>	**35**	/	?
36 note 1	Right shift	na	**1D** note 1	Left Ctrl	na	**38** note 1	Left Alt	na
39	Spacebar	Spacebar	**E0,38** note 1	Right Alt	na	**E0,1D** note 1	Right Ctrl	na
E0,52	Insert	na	**E0,53**	Delete	na	**E0,4B**	Left Arrow	na
E0,47	Home	na	**E0,4F**	End	na	**E0,48**	Up Arrow	na
E0,49	Pg Up	na	**E0,51**	Pg Dn	na	**E0,4D**	Right Arrow	na
45,C5 note 1	Num Lock	na	**47**	Keypad 7	Home	**4B**	Keypad 4	Left Arrow
4F	Keypad 1	End	**E0,35**	Keypad /	Keypad /	**48**	Keypad 8	Up Arrow
4C	Keypad 5	na	**50**	Keypad 2	Dn Arrow	**52**	Keypad 0	Insert
E0,37	Keypad *	Keypad *	**49**	Keypad 9	Pg Up	**4D**	Keypad 6	Right Arrow
51	Keypad 3	Pg Dn	**53**	Keypad .	Delete	**4A**	Keypad -	Keypad -
4E	Keypad +	Keypad +	**E0,1C**	Keypad Enter	Keypad Enter	**01**	Escape	Escape
3B	F1	note 3	**3C**	F2	note 3	**3D**	F3	note 3
3E	F4	note 3	**3F**	F5	note 3	**40**	F6	note 3
41	F7	note 3	**42**	F8	note 3	**43**	F9	note 3
44	F10	note 3	**D9**	F11	note 3	**DA**	F12	note 3
2A,37	Prnt, Scrn	na	**46**	Scroll Lock	na			

Modos de acceso al teclado

1.- Uso de interrupciones para acceder a las funciones del teclado

Fundamento Teórico

El teclado de un PC cuenta en su interior con una matriz de contactos y un pequeño circuito integrado (habitualmente un 8741,8742 o alguno compatible) que es el encargado, entre otras funciones de transmitir al ordenador lo que se denomina scan code o código de la tecla que se ha pulsado. Este código no tiene nada que ver con el código ASCII de un carácter, ya que de hecho algunas teclas no tienen correspondencia con caracteres ASCII. Es el controlador del teclado, normalmente el programa KEYB , el que se encarga de traducir cada código de teclado a su correspondiente código ASCII, en caso de que esto sea necesario.

¿Qué es una interrupción?

Las interrupciones son eventos que implican la ejecución de una determinada rutina por parte de la CPU. Frecuentemente, la comunicación entre CPU y periféricos se realiza vía interrupciones.

Cuando se llama a una rutina software (desde un programa) debemos especificar qué rutina debe ejecutar. Esto se realiza indicando el número de interrupción y la función específica dentro de esa interrupción.

Además, una determinada rutina de interrupción suele necesitar unos parámetros de entrada, y normalmente devuelve una serie de valores. Para ello se usan algunos registros de la CPU, por lo que es necesario inicializarlos con los parámetros de entrada antes de ejecutar la rutina de servicio de interrupción y leer los valores devueltos tras la ejecución de la misma.

> Las llamadas a las interrupciones por software se pueden realizar en los lenguajes de alto nivel mediante funciones ya implementadas. En esta primera práctica vamos a usar una función llamada **int86** que la podemos encontrar en las librerías de los compiladores que usemos.
>
> ***¿Dónde está la función int86 declarada?,*** depende del compilador de C que vayamos a usar, en la mayoría de los compiladores existe una librería llamada dos.h como es el caso del compilador Turbo C 2.01, pero en otros casos como el del compilador DJGPP esta función se encuentra dentro de la librería pc.h . Ambos compiladores están instalados en los pc´s del aula debiéndose usar preferentemente Turbo C.

¿Cómo empezar a usar int86?, para poder usar la función int86 se debe incluir la librería que la define, pongamos como ejemplo que se esté usando TurboC, en tal caso :

```
#include <dos.h>  (si se decide usar djgpp : #include <pc.h>)
....
....
int main()
{
 .....
 int86(0x16, &entrada, &salida);
…  }
```

Sintaxis de la función int86 :

int int86(**int** numero, **union REGS *** entrada, **union REGS *** salida)

Como se ha adelantado en el ejemplo, la función int86 necesita como

Parámetros de entrada :

int número : Este número es el que indica el vector de interrupción que vamos a usar, en el ejemplo es el vector de interrupción 16 (interrupción de teclado).

union REGS *entrada : registros de entrada a la función , aquí se cargan los datos que le pasamos de entrada a int86. Este tipo de datos es un registro que a su vez contiene mas registros (por eso se denomina union) y que emulan a los registros de una cpu 80x86 muy básica.

Parámetros de salida:

union REGS *salida : registros de salida de la función, aquí podemos leer el resultado de la ejecución de la función int86. Cuando invocamos a la función int86 y le pasamos un numero de interrupción para usar y un conjunto de registros con información, la función hace las operaciones que le hayamos pedido y luego deja los resultados en los registros de la unión de salida.

Además de que la función int86 devuelve un código de error (entero) que da información sobre el éxito de la operación que se ha intentado llevar a cabo.

¿Cuáles son las componentes de **union REGS**?

Al hablar de la sintaxis de la función **int86** hemos citado un tipo de datos llamado **union REGS** este es un union (en C un union es un registro que a su vez tiene muchos mas registros dentro) además se ha indicado que union REGS emulan a los registros de una cpu 80x86 básica ahora se va a mostrar cómo están dispuestos estos registros y cómo acceder a ellos (es una emulación software).

```
union REGS
{
        struct WORDREGS  x;         /* registros de 16 bits */
        struct BYTEREGS    h;       /*  registros de   8 bits */
}
```

Los registros WORDREGS están definidos de la siguiente forma :

```
struct WORDREGS
{
        unsigned int ax , bx , cx , dx;
        unsigned int si , di , cflag, flags;
};
```

```
struct BYTEREGS
{
        unsigned char al, ah, bl, bh;
        unsigned char  cl, ch, dl, dh;
};
```

Vamos a ver un ejemplo para clarificar tanto la sintaxis de la función int86 como el uso de la misma y de los registros:

En este ejemplo vamos a desarrollar una función que desplaza el cursor del teclado a cualquier punto (x,y) de la pantalla.

```
#include <stdio.h>
#include <dos.h>

void desplazaCursor (int x, int y)
{
        union REGS registrosE;       /* Registros de entrada */
        union REGS registrosS;       /* Registros de salida   */


        registrosE.h.ah = 2;
        registrosE.h.dh = y;
        registrosE.h.dl         = x;
        registrosE.h.bh = 0;

        int86(0x10,&registrosE,&registrosS);
}

void main()
{
        desplazaCursor(1,1);
        desplazaCursor(10,10);
        desplazaCursor(10,20);
}
```

Como podemos apreciar, en este código llamamos a la interrupción 10h (0x10) que es la interrupción de video. Para que se ejecute correctamente la función desplazarCursor necesitamos colocar en el registro ah (registro byte) un dos, por eso accedemos al conjunto de registros h (que son los declarados como BYTEREGS) y luego dentro de estos accedemos al ah que es el que leerá int86 para saber que hacer, en ah colocamos un 2 para que mueva el cursor, para saber a dónde debe desplazar el cursor hay que dar coordenadas enteras , la coordenada x en el registro dl y la coordenada y en el dh. En este caso la funcion int86 no genera información de salida por tanto la información que coloque en registros no nos interesa, tal es asi que para ahorrarnos la definición podriamos haberla llamado asi:

int86 (0x10, &registrosE, &registrosE);

La interrupción de teclado

Para acceder al teclado se usa la interrupción 16h. Dicha interrupción dispone de varias funciones a las que sea accede inicializando el registro **ah** con un determinado valor. Vamos a describir brevemente algunas de ellas:

Leer carácter desde el teclado

Num. Interrupción	16h
num. función	0
Entrada	AH=0
Salida	AL=ASCII tecla

Detección de tecla pulsada

Num. Interrupción	16h
num. función	1
Entrada	AH=1
Salida	Flag Z=0 tecla pulsada Flag Z=1 No hay tecla
Nota: Flag Z es el bit numero 6 del registro flags.	

Estado del teclado

Num. Interrupción	16h
num. función	2
Entrada	AH=2
Salida	Estado del teclado.*

Ajuste factor de repetición

Num. Interrupción	16h
num. función	3
Entrada	AH=3 AL=5 BH=retardo BL=factor repetición
Salida	No tiene.

Retardo:
- 0 retarda 0.25 s
- 1 retarda 0.50 s
- 2 retarda 0.75 s
- 3 retarda 1.00 s

Repetición:
- 0 = 30 caracteres por seg.
- .....hasta.....
- 31= 2 caracteres por seg.

*Estado del teclado: El estado del teclado se interpreta según los valores del registro AL que pueden ser los que se muestran....

BIT 0	Shift derecho pulsado
BIT 1	Shift izquierdo pulsado
BIT 2	Control pulsado
BIT 3	Alt pulsado
BIT 4	Bloq Despl pulsado
BIT 5	Bloq Num pulsado
BIT 6	Bloq Mayusc pulsada
BIT 7	Insert pulsada

2.- Acceso a puertos de E/S de teclado

Una alternativa al acceso a las funciones de teclado es el uso de los puertos. Todo periférico se "mapea" a un conjunto de direcciones del sistema, a este conjunto de direcciones se le denomina puertos. Los puertos son accesibles para lectura y escritura y en el caso del teclado, las direcciones asociadas son 0x60 para datos y 0x64 para control. Veamos un ejemplo del acceso al teclado a través de sus puertos.....

```
.....
.....
unsigned char codigoScan;
...
...
codigoScan = inportb (0x60);
….
printf("codigo scan leido: %u", codigoScan);
```

este fragmento de código incompleto lee de la dirección 0x60 lo que el teclado haya dejado alli, que es el codigo de escaneo de una tecla pulsada que luego habrá de convertirse en un codigo ASCII.

Las funciones de las que disponemos para acceder a la informacion de los puertos del teclado son :

```
inportb(PUERTO);
outportb(PUERTO, DATO);
```

Este tipo de alternativas es útil a la hora de mandar datos puntuales y poco urgentes al puerto de control de teclado o para rutinas de usuario de baja prioridad..

3.- Acceso al área de memoria del buffer de teclado para alterar su estado

Cómo se ha comentado durante toda la práctica, las teclas especiales como la tecla SHIFT o la tecla de MAYÚSCULAS no generan código ASCII (aunque si generan scan codes) , es posbile examinar un conjunto de bits que se denominan bits de estado y que reflejan el estado de estas teclas especiales, estos bits residen en un área de memoria especial (buffer de teclado) así cuando llega una pulsación de tecla, la BIOS chequea estos bits para saber si la tecla es mayúscula o minúscula.

0x00400017	
Bit	**1=activo, 0=inactivo**
7	Modo Inserccion
6	Mayusculas
5	Teclado Numerico
4	Scroll
3	Alt
2	Control
1	Shift izquierdo
0	Shift derecho

El byte que nos interesa modificar para la realización del ejercicio 3 reside en la dirección 40:17h o **0x00400017**.

REALIZACIÓN PRÁCTICA

1 Interrupciones

Haciendo uso de interrupciones se deben implementar las siguientes funciones dentro de una aplicación desarrollada en lenguaje C, se pueden elegir los compiladores TurboC y DJGPP, para esta primera práctica es aconsejable el compilador TurboC 2.01.

1.- Mostrar los códigos scan o bien los ASCII de la tecla que se pulse.
2.- Modificar la velocidad del teclado.

2 Acceso a puertos de E/S de teclado

Para que sean capaces de comparar las técnicas de acceso al teclado, deben familiarizarse con el manejo de los puertos en los que se mapea el teclado.

1.- Completar y explicar con detalle el código fuente que se suministra y cuyo objetivo es implementar la lectura de los códigos scan pero en lugar de usar interrupciones se hace accediendo a los puertos de teclado.

- Para este punto es necesario manejar los conceptos :
  - Vector de interrupciones.
  - Función setvect y getvect.
  - Bit EOI.

++ El código fuente para este ejercicio es puertos.c

3 Acceso al buffer de teclado

1.- Completar el programa que se proporciona y con las indicaciones que se proporcionan altere el comportamiento del teclado, según se indica.

++ El código fuente es teclado.c