Periféricos Avanzados
Práctica 2. Funciones del ratón.

Conceptos básicos

La manera de usar los servicios que ofrece un driver de ratón se parece mucho a la manera de usar los distintos servicios del DOS o de la BIOS, existe una interrupción dedicada a esta tarea. La 33H.

Un driver que respete el estándar de Microsoft debe usar esta interrupción como interfaz donde los distintos servicios se distribuyen en funciones cuyo número de identificación se pasa en el registro AX al invocar la interrupción software.

Haciendo honor a la verdad no solo existe el estándar de Microsoft, sino que en realidad existen una sucesión de estándares empezando por el 1.0 y llegando al 8.x que es el que tenemos ahora. Antes de entrar en profundidad con el ratón hay que saber que el driver de ratón debe proporcionar dos datos importantes, como son la posición del ratón y el estado de los botones teniendo en cuenta además que existen ratones con dos y tres botones, el leer el estado de los botones no tiene excesiva complicación pero en cuanto a la posición del cursor debemos tener en cuenta algunos factores ya que bajo DOS podemos ejecutar una aplicación tanto en modo texto como en modo gráfico sin embargo la posición del ratón se referirá siempre a los pixeles de la pantalla. Es decir, en un modo gráfico serían simplemente los pixeles de la pantalla gráfica a los que accedemos ya de por sí pero en los modos de texto habrá que dividir ambas coordenadas dadas por el driver entre 8 ya que las celdas de los caracteres se componen de 8 x 8 pixeles . Se dice también en este sentido que el ratón usa una pantalla grafica virtual.

Las distancias del recorrido del ratón se miden en Mickeys, cada mickey equivale a 1/200 pulgadas en ratones tradicionales y a 1/400 pulgadas en los mas modernos. Nosotros usamos el sistema métrico pero esto no nos afecta por que es el driver el encargado de realizar las conversiones a pixels.

Empezando a programar

El paso previo a cualquier actividad con el ratón es su inicialización con la función 00h que realiza un reset del driver. Además es útil para averiguar si efectívamente se ha cargado en el sistema un driver de ratón, el valor de retorno FFFFh confirmará su presencia (valor que se almacena en el registro AX) dando en BX el numero de botones del ratón. El valor 0000h en el registro AX confirmará la ausencia de ratón .

Efecto de reset en el driver de raton :

1.- El cursor se lleva al centro de la pantalla.

2.- Se oculta el puntero, por lo que para usarlo tendremos que activar su visibilidad , tendremos que usar la función 01h en AX para mostrarlo, no devuelve ningun resultado.
Nota: el número de llamadas a la funcion 01h ha de estar equilibrado con el número de llamadas a la función 02h (02h es para ocultar el puntero) para que surta efecto. Por ejemplo :

> Dos llamadas a la funcion 01h cuando antes he hecho 3 llamadas a la función 02h tendrá como resultado el puntero del raton oculto en lugar de visible.

3.- Se restablecen posibles manejadores.
4.- La emulación de lápiz óptico se ajusta a 8 Mickeys / punto en horizontal y 16/8 en vertical.
5.- El umbral de duplicación de velocidad se pone en 64 Mickeys/segundo.

La apariencia del cursor depende del modo gráfico, en modo texto será un rectángulo y en modo gráfico una flecha tal y como la conocemos , en ambos casos en la página 0.

Información sobre el raton y el driver

Funciones 24h y 25h (36 y 37). Si colocamos en el registro AX un 24h estaremos preguntándole al ratón sobre sus datos técnicos, en BH y BL deja el numero de versión del driver que lleva instalado. En el registro CH lleva la información del tipo de raton que es (1 = BUS; 2=SERIE, 3=Inport 4=PS2) y en CL indica el IRQ del mismo.

Si ponemos en AX el 25h o 37 (en decimal) estamos obteniendo información sobre el driver en si y no acerca del ratón, El resultado lo da en el registro AX que si es inferior a 32000 quiere decir que es un dispositivo instalado en config.sys y si no será un driver instalado en COM.

Visualizar el cursor del raton

Registro AX = 2 si deseamos ocultar, registro AX=1 si queremos ver el cursor.

Movimiento del cursor y velocidad

La posición actual del cursor del raton la podemos obtener poniendo un 3 en AX y recogiendo la información (cx,dx).

En ocasiones interesa que el programa pueda mover directamente el cursor, esto lo realiza la función 04h. Toma sencillamente la ordenada (x) en CX y la abcisa (y) en DX.

Trabajar con el raton en modo gráfico o modo texto conlleva una conversión en las unidades de medida, mientras que en modo gráfico se miden pixels , en modo texto se miden Mickeys. Muy útil es tambien la posibilidad de ajustar la velocidad del ratón, la manera de hacerlo es indicar una relación de Mickeys por ocho puntos de pantalla por separado en sentido horizontal y vertical con la función 0Fh .

Es posible limitar el movimiento horizontal del raton con la funcion AX=7 y colocando en los registros cx ydx el minimo y el máximo del rango en el que el ratón podrá moverse, para limitar el movimiento vertical la función a usar en el registro AX es la 8.

Estado del ratón

Lo que más interesa del ratón es sin duda su posición exacta en la pantalla y el estado de los botones, para este propósito se ha ideado la función 03h. En los registros CX y DX almacena la posición en el sentido horizontal y vertical respectivamente.

Cambio del aspecto del cursor del ratón en modo texto

AX = 10 y en los registros bx = tipo, cx = screen, dx = cursor;
Donde tipo es 0 ò 1 el cero indica que el cursor es modo texto y el 1 que es modo gráfico, screen es el color que tendrá el cursor, y cursor será un carácter que sustituye al actual cursor del raton.

Los rangos entre los que podemos seleccionar valores para screen y cursor son :

Screen → [0-65535]
Cursor → [0-255]

Probad los parámetros: tipo= 0 , screen=12012 , cursor = 200

Sensibilidad del ratón

La sensibilidad del ratón es por defecto 8 en horizontal y 16 en vertical, (es decir que un mickey en horizontal es 8 y en vertical es 16) podemos alterar este comportamiento con la función 15, asi que se deberá colocar en AX = 15, en CX=la sensibilidad horizontal y en DX la sensibilidad vertical.

Aumentar el umbral del factor de aceleración

En AX debemos poner un 19 (función usada para tal tarea) y en DX el umbral de velocidad, el valor por defecto es 64.

Además de la funcionalidad básica que hemos visto, la programación del ratón ofrece funciones avanzadas con posibilidades muy interesantes. La más destacada es la posibilidad de enlazar el driver con un manejador o handler propio.

Con las funciones vistas hasta ahora podemos averiguar el estado del ratón en cualquier momento, pero tenemos que llamarlas expresamente desde nuestro programa (practica) . En un entorno de ventanas esto supondría vigilar con un bucle de llamadas repetidas. La mayor parte de las veces ni siquiera se habrá tocado el ratón y la información obtenida será altamente redundante. Un enfoque alternativo a este enfoque de polling es la de usar la gestión del ratón por **eventos**.

Eventos :

En este caso es el ratón el que obliga a las llamadas a la interrupción sólo cuando ocurre algo como la pulsación de uno de sus botones o el desplazamiento de su puntero. De esta manera solo habrá llamadas con información útil y su número se reducirá drásticamente con el consiguiente ahorro de CPU.

Para este fin el driver de ratón dispone de varios servicios que permiten pasarles un manejador (handler) , o sea , una función desarrollada por nosotros que será llamada por el driver del raton bajo determinadas circunstancias. Veamos un ejemplo :

Ejemplo (comparación de sendos métodos) :

Monitorizar el estado del ratón:

```
Void MiFuncion()
{
        union REGS registroE, registroS;
        .....
        .....
        int86(0x33,&registroE,&registroS);

        printf("Estado botones : .... %X", registroS.x.bx );

        ......

}
```

Metodo 1 : Polling	Metodo 2: Eventos

MiPrograma
{
….
 	
for(;;)
{
MiFuncion();
 	
if (getch()='x') break();
}
 	
}
 	
Para poder ver el estado del ratón en cada momento es necesario que ejecutemos esta aplicación (MiPrograma) y esta aplicación lo que hace es estar continuamente preguntando al ratón por su estado, si el ratón no esta haciendo nada nos dará la información de la ultima vez que se movió o de botón inactivo, esto a la vez que estamos preguntando continuamente al ratón estamos generando un consumo de cpu importante. La ventaja de este método es que es muy sencillo de programar , tanto que se limita a meter en un bucle la llamada a la interrupción 0x33, la gran desventaja es lo perjudicial que es en términos de rendimiento.
	0.- Le indico al driver que ante el servicio de pulsación de boton llame a la funcion MiFuncion.
	
	  Cualquier acción del usuario sobre el ratón genera un evento, el driver de ratón debe responder ante este evento y bien lo hace ejecutando "manejadores" propios del driver o bien ejecutando los que nosotros hayamos implementado. De esta forma, la pulsación de un botón del ratón llama al driver, este llama a nuestra función y en ese momento se imprimirá por pantalla el estado de los botones del ratón sin necesidad de :
	
	1- tener que ejecutar una aplicación especial para monitorizar el ratón por que ya sabe el driver lo que tiene que hacer.
	2- Tener que estar ejecutando la función dentro de un bucle y por tanto obteniendo información que no nos sirve de nada por ejemplo si el ratón no se toca el bucle sigue ejecutándose, pidiendo información al ratón así que la información que obtenemos en este caso es redundante.

Eventos a fondo

La filosofía de la gestión de eventos del ratón es evitar que el ratón sea interrogado continuamente por nuestro programa para conocer el estado de los botones. Es mucho mejor delegar en el propio ratón la facultad de "avisar" cuando se produce la pulsación de un botón.

Para ello se usan las interrupciones. El driver de ratón, por ejemplo, se pone en marcha cuando llega una señal del ratón físico al puerto serie y provoca la IRQ3 (puerto serie 1) , IRQ4 (puerto serie 2) o IRQ12 (PS/2).

Eventos e interrupciones

El vector de interrupción correspondiente a estas IRQs ha sido modificado previamente al cargar el driver, lee ocho bytes del puerto serie 1.200 baudios 7 bits sin paridad que contienen la información sobre el estado de los botones y la distancia recorrida horizontal y verticalmente.

Sólo en el caso de que ocurra algo, se activa y se atiende a la interrupción correspondiente que se encuentra vinculada al driver. Microsoft (que fue el padre del estándar del driver del ratón) pensó en esto y preparó el driver en este sentido : su estándar define una interfaz para "conectar" alguna rutina nuestra con el driver de ratón .

Manejadores de eventos para el ratón

Podemos ir más lejos a la hora de declarar el manejador de eventos del ratón y es que podemos diseñar el manejador exclusivamente para que responda a determinados eventos del ratón. La manera de hacer que el manejador responda a determinados eventos es a través del uso de lo que se conoce cómo máscara. Cada tipo de evento tiene en esa máscara un bit asociado , si ponemos a uno el bit en la mascara entonces estaremos habilitando al driver a que actue con nuestra función cuando se produzca ese evento.

Instalación de un manejador de eventos

Esto es, ¿ cómo se conecta la función que hemos desarrollado con el driver del ratón ?.
A la hora de conectar nuestra función con el driver del ratón lo mas complejo es afrontar el modo de direccionamiento de memoria de los intel x86, (básicamente los intel direccionan con direcciones de memoria lineales de 20 bits que se trasladan de seg:off a dirección lineal de la forma = seg*16 + offset .

La instalación del manejador requiere que usemos la función 0Ch (o 12 en decimal). Se deben pasar en la forma ES:DX las direcciones de segmento y desplazamiento del manejador (se pueden usar las funciones FP_SEG() y FP_OFF() de c para tal tarera) y en CX la máscara de eventos a los que debe responder ese manejador. A partir de ahí el driver llamará al manejador siempre y cuando ocurran los eventos que recoge la máscara.
(ver Tabla A).

Para que la instalación de nuestro manejador sea limpia y eficaz, deberemos cumplir las siguientes condiciones :

1.- Procedimiento FAR
2.- Salvar y restaurar el estado de los registros.
3.- Tiene que cargar el segmento de datos del programa en el lenguaje de alto nivel en DS.

NOTA :
DireccionLineal : (ver modo direccionamiento en los intel (al final))

```
((long)FP_SEG(funcion) * 16L) + (long)FP_OFF(funcion);
```

donde función es una función del tipo "void FAR interrupt" que debe cumplir varias condiciones antes de ser instalada.
FP_SEG y FP_OFF devuelven el segmento y el desplazamiento de la funcion (mas detalle en anexo).

EJEMPLO :

Como se ha descrito antes, la función del driver de ratón que hemos de usar es la 0Ch (o la 12 en decimal) a continuación se muestra la rutina necesaria para instalar el manejador que implementéis enlazado al driver.

NOTA:
Usar con extremo cuidado ya que la función cuya dirección vamos a pasar al driver del raton debe terminar con un return del tipo FAR, debe ser compilada usando el modelo de memoria de TurboC "LARGE" y no deben usarse dentro de la funcion "manejador" de la que estamos hablando ni servicios del DOS ni servicios de la BIOS

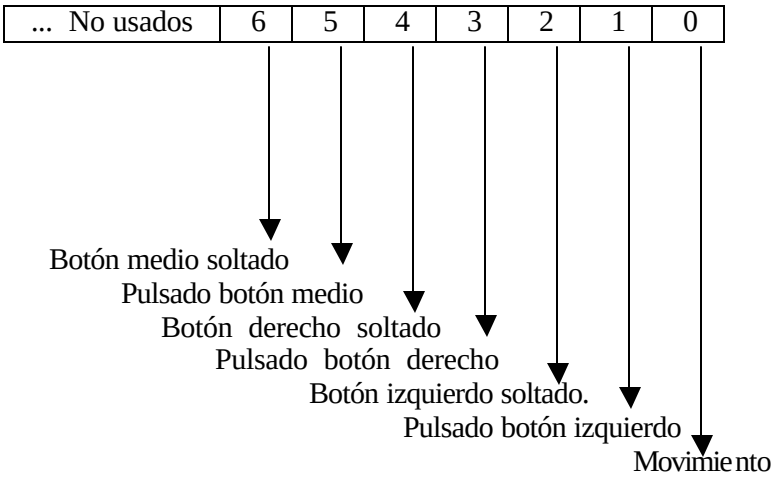
```
void ms_instala_subrutina(int mascara,
                          unsigned segmento_rutina,
                          unsigned offset_rutina)
{
      union REGS registros;
      struct SREGS s_registros;

      registros.x.ax = 12;
      registros.x.cx = mascara;
      registros.x.dx =  offset_rutina;       /* fp_off(funcion); */
      s_registros.es  = segmento_rutina;     /* fp_seg(funcion);*/
      int86x(0x33,registros, registros ,s_registros);
}
```

Aclaración :
Mascara = indica ante que eventos actuara el driver con el manejador.

Segmento_rutina y offset_rutina son el segmento y el desplazamiento de la funcion dentro del segmento.

Máscara del manejador :



ANEXOS

Tabla A (funciones básicas en detalle).

Función	Entrada	Salida	Descripción
00h – reset	AX=0000h	AX=FFFFh : Hay un driver instalado y BX contiene el número de botones. AX=0000h : no hay driver instalado.	El cursor se mueve al centro de la pantalla y se oculta. En modo texto es un rectángulo invertido y en modo gráfico una flecha. Se usa la pantalla gráfica 0 y los manejadores de eventos se desactivan
01h-mostrar cursor	AX=0001h	No hay	El numero de llamadas (01h-02h) ha de estar equilibrado.
02h-ocultar cursor	AX=0002h	No hay	Idem 01h
03h-estado botones y posición cursor	AX=0003h	BX=Estado Bit 0: Boton izquierdo pulsado Bit 1: Derecho pulsado Bit 2: Central pulsado Bit 3-15: Sin significado CX=Coordenada X DX=Coordenada Y	El bit 2 no tiene significado cuando el ratón no tiene tres botones. Nota: En la tabla de BX, el Estado es ese cuando el bit respectivo está a 1, el opuesto si está a 0.
04h-mover cursor	AX=0004h CX=coord. X DX=coord. Y	No hay	Si las coordenadas superan los limites de la pantalla impuesta por las funciones 07h y 08h se adaptan las coordenadas para que estos límites se sigan respetando.
05h-Núm pulsaciones	AX=0005h BX=boton examinado. 0=izq, 1=der, 2=centro	AX=estado de todos los botones. BX=num pulsaciones desde la ultima llamada a esta funcion. CX=coordenada X DX=coordenada Y	El contador de pulsaciones para el boton especificado se vuelve a poner a cero.
06h-numero de veces que se ha solicitado un botón determinado.	AX=0006h Análogo función 05h.	Análogo funcion 05h	Vease funcion 05h
07h-limite horizontal cursor	AX=0007h CX=Xmin DX=Xmax	No hay	Si el cursor no se encuentra en el área especificada en el momento de la llamada se encarga el driver de desplazarlo. Si el valor que hay en DX es menor que el valor que hay en CX, se intercambian.
08h-limite vertical	AX=0008h CX=Ymin DX=Ymax	No hay	Idem 07h
0Fh-ajustar velocidad	AX=000Fh CX= numero mickeys horizontales que representan 8 puntos. DX=Numero de mickeys verticales que representan 8 puntos.	No hay	Se permiten valores entre 1 y 32767. Un reset establecerá los valores por defecto.
1Dh-Establecer pagina para el cursor	AX=001Dh BX=Numero pagina	No hay	Valor por defecto es 0.
1Eh-Pagina cursor	AX=001Eh	BX=numero pagina	-

Servicios que interactúan con el driver del ratón.			
Función	Entrada	Salida	Descripción
0Ch	AX=000Ch CX = máscara eventos ES:DX = Dirección Manejador	-	Permite instalar un manejador de eventos. En las llamadas al manejador se pasa la siguiente información: AX: Máscara evento actual. BX: Estado botones.(ver función 03h) (CX,DX): posición en la pantalla (x,y). SI/DI: Longitud horizontal/vertical del ultimo movimiento. DS: Segmento de datos del driver.
14h	Idéntica a la anterior salvo que en AX=0014h	CX : Máscara de eventos del manejador anterior . ES:DX : dirección FAR del manejador anterior.	Cambia el manejador por otro y permite averiguar la dirección del antiguo. La información que se pasa al manejador es idéntica que en la función 0Ch.
18h	AX=0018h ES:DX : dirección manejador	AX: 0018h → Todo bien. AX: FFFFh → No se pudo instalar el manejador.	Instala un manejador alternativo, es posible instalar hasta tres manejadores simultaneos siempre que sus máscaras de eventos sean diferentes. Los parámetros devueltos son idénticos a la función 0Ch.
19h	AX=0019h CX=mascara de eventos manejador referenciado	AX: 0000h → Todo bien, en este caso CX contiene la máscara de eventos. ES:DX Dirección manejador.	Averigua la dirección de un manejador alternativo, la máscara de eventos toma el formato de la usada en la función 18h.
1Ah	AX: 001Ah BX: Mickeys horizontales CX:Mickeys verticales DX: Umbral para duplicar la velocidad en Mickeys/segundos		Permite calibrar la sensiblidad del ratón. El umbral de duplicación de velocidad se encuentra inicializado a 64 Mickeys/seg.
1Bh	AX=001Bh	BX = Mickeys horizontalesque representan ocho puntos. CX = Mickeys verticales que representan ocho puntos. DX = Umbral duplicación velocidad.	Averigua la sensibilidad del ratón.
28h	AX=0028h CX modo de video DX tamaño de la fuente	CX : 0 → OK	Establecer el modo de video. El parámetro DX sólo se necesita dónde el byte de más peso representa al tamaño en vertical y el de menor peso el tamaño en horizontal.
29h	AX=0029h CX=modo de video	BX=dirección de segmento de un string CX=Número del modo de video DX=offset del string.	Averiguar los modos de video disponibles. Si se quieren conocer todos los modos de video disponibles, hay que usar la función varias veces. La primera llamada se haría con CX=0 y las siguientes distintas de 0. En cada llamada se describe otro modo de video hasta que un cero en CX indique que se han visto todos los modos disponibles. No siempre se devuelve el String, si la pareja BX:DX contiene ceros es que no se ha devuelto, en el caso contrario, da información adicional sobre las propiedades del modo y termina con el signo $.

Resumen de las funciones del estándar de Microsoft	
00h	Reset del driver.
01h	Mostrar el cursor en pantalla.
02h	Quitar el cursor de la pantalla.
03h	Averiguar la posición del ratón y el estado de los distintos botones.
04h	Mover el cursor.
05h	Averiguar el numero de pulsaciones de un boton.
06h	¿Cuántas veces se ha soltado un botón?.
07h	Determinar el limite horizontal del movimiento del ratón.
08h	Determinar el limite vertical del movimiento del ratón.
09h	Definir el aspecto del cursor del ratón en modo gráfico.
0Ah	Definir el aspecto del cursor de l ratón en modo texto.
0Bh	Averiguar la distancia entre la posición actual y la posición en la ultima llamada a esta función.
0Ch	Instalar manejador de eventos.
0Dh	Activar emulación de lápiz óptico.
0Eh	Desactiva emulación de lápiz óptico.
0Fh	Determina la relación entre Mickeys y pixels.
10h	Permite definir un área de la pantalla donde desaparece el cursor.
11h	No documentada.
12h	No documentada.
13h	Define el umbral de la velocidad del ratón a la que se duplica la velocidad del cursor.
14h	Cambiar el manejador de eventos por otro.
15h	Averiguar el tamaño del bufer de estado del ratón.
16h	Copiar la información de estado actual en un bufer de usuario.
17h	Restaurar el estado del ratón.
18h	Instalar varios manejadores de eventos.
19h	Averiguar la dirección de un manejador alternativo.
1Ah	Graduar la sensibilidad del ratón.
1Bh	Averiguar la sensibilidad del ratón.
1Ch	Graduar la frecuencia de interrupción del ratón.(Frecuencia con la que el ratón activa la interrupcion del puerto.).
1Dh	Activar la página de pantalla para el cursor del ratón.
1Eh	Averiguar la página actual (de pantalla) del cursor de ratón.
1Fh	Desactivar el driver de ratón.
20h	Activar el driver de ratón.
21h	Reset del driver de ratón.
22h	Elegir idioma para los mensajes del driver del ratón.
23h	Averiguar idioma para mensajes
24h	Averiguar el tipo de ratón y versión del driver.
25h	Obtener información diversa: Tipo driver, estado cursor, y frecuencia de interrupción.
26h	Leer la extensión de la pantalla gráfica virtual para el ratón, dice tambien si el driver está o no activo.
27h	Leer mascaras de bits para el cursor software.
28h	Activar un modo de video determinado.
29h	Averiguar los modos de video disponibles.
2Ah	Informacion diversa sobre el cursor del ratón.
2Bh	Ajustar las cuatro curvas de aceleración.
2Ch	Leer la curva de aceleración actual.
2Dh	Ajustar/leer la curva de aceleración actúal.
2Eh	No documentada.
2Fh	Reset del hardware del ratón.
30h	Leer/ajustar la graduación del ball-point mouse (es un ratón que se puede rotar)
31h	Leer la extensión de la pantalla virtual, no confundir con la funcion 2h
32h	Averiguar las funciones soportadas a partir de la 25h
33h	Copiar parametros del driver en un buffer.
34h	Averiguar la ubicacion del fichero MOUSE.INI
35h	Soporte de cursor agrandado para pantallas LCD.

Direccionamiento de memoria de los x86

Cuando Intel diseñó los procesadores 8086/88 introdujo un concepto bastante curioso : la dirección de acceso a memoria es normalmente lineal, es decir, con un número se accede a la memoria en... Pero el 8086/88 direcciona la memoria mediante dos valores entre 0 y 65535, un segmento y un offset (o desplazamiento) formando la dirección física según la fórmula **segmento*16+offset.** Es decir el segmento es la base y el offset es el desplazamiento sobre esa base, con un segmento determinado nos podemos desplazar hasta un máximo de 64 kbytes .

La razón del porqué es que cuando se diseñaron estos procesadores, los procesadores solían trabajar con 64 kbytes de memoria como máximo. Intel quiso dar un paso gigantesco en aquellos tiempos (1980) al ampliar la capacidad de direccionamiento a 1Mbyte, pero para direccionar una memoria de este tamaño se necesita un registro de 20 bits y eso era difícil con la tecnología de entonces. Asi que Intel ideó el truco de combinar dos registros de 16 bits (pueden direccionar 64KB) de la manera descrita para formar la dirección real.

Para poder acceder a toda la memoria hay que usar entonces por lo menos 16 segmentos distintos (64 Kbytes * 16 = 1 Mbyte) a través de los que nos desplazamos. Aún sabiendo poco del funcionamiento de un microprocesador se puede intuir la complicación que esto implica. Para aliviar esta situación, se han diseñado cuatro registros de segmento utilizables a la vez para el acceso a memoria con propósitos distintos :

CS	Acceder a código.
DS	Acceso a datos.
SS	Para la pila.
ES	Como segmento auxiliar

De manera que un programa se divide en segmentos con código, datos y la pila a los que se accede con los distintos registros. Se usan de manera implícita, es decir, la lectura de una instrucción se hace usando el segmento CS y una variable el DS, aunque se pueden indicar explícitamente los registros a usar. Los registros de segmento se pueden cargar con los valores adecuados y acceder de esa manera a la memoria entera. Así cuando se quieren acceder a direcciones de memoria por encima de los 64Kbytes hay que cambiar el valor de CS, esto se conoce como salto FAR, para acceder a direcciones inferiores no es necesario, ya que sólo con el offset nos bastaría para llegar a esa dirección con un salto que llamaríamos NEAR (cercano).

Hay que tener mucho cuidado en las llamadas (CALL) sobre todo, hay procedimientos FAR (retorno FAR) y procedimientos NEAR (retorno con NEAR) y hay que llamarlos con su CALL FAR o NEAR correspondiente. Si no concuerdan el tipo de retorno con el tipo de llamada provocaremos inconsistencias en el sistema.

Aunque las arquitecturas a partir del 80286 ya son diseños nuevos, mantienen un modo de compatibilidad con el 8086 y sus problemas.

Sugerencias para la parte opcional de la práctica (manejador)

Para las llamadas a las funciones 0009h, 000Ch y 0018h se necesita pasarle a la interrupción 33h un puntero en los registros ES:DX. Para ello, debemos usar la función ***int86x*** que viene definida en el fichero cabecera *dos.h* .

La sintaxis de la función es la siguiente :

```
#include <dos.h>
int int86x(int intno, union REGS *registrosE, union REGS *registrosS, struct SREGS s_regs);
```

donde intno es el número de interrupción que queremos llamar (aquí 0x33). En registrosE se especifican los valores de los registros antes de la llamada y en registros obtenemos los valores de los mismos tras ser ejecutada la rutina correspondiente (ver práctica anterior). En s_regs podemos especificar los valores de los registros de segmento ES y DS (los demás segmentos son ignorados por ***int86x***, int86x además restablece DS al valor previo tras la llamada). La estructura SREGS está definida de la siguiente forma:

```
struct SREGS{
    unsigned int es;
    unsigned int cs;
    unsigned int ss;
    unsigned int ds;
};
```

Para poder obtener la dirección de segmento y de offset de una rutina o una variable podemos utilizar las macros definidas en dos.h : **FP_SEG** y **FP_OFF** . Estas macros en turboC están definidas de la siguiente forma :

```
#define FP_OFF(fp) ((unsigned)(fp))

#define FP_SEG(fp) ((unsigned)((unsigned long)(fp) >> 16)) ...
```

Para definir la función que vaya a controlar los eventos del ratón mediante las funciones 000Ch y 0018h, recordemos que éstas deben cumplir los siguientes requisitos :

1.- Estar definidas como FAR (ya que las rutinas del driver del ratón se hayan en otro segmento).
2.- Han de guardar los diferentes registros del procesador y volverlos a restaurar luego (*_saveregs*)
3.- No se debe comprobar el desbordamiento de la pila.
4.- Si quieren acceder a rutinas o variables NEAR del propio programa deben cargar la dirección del segmento de datos (DS) donde se encuentren dichas rutinas o variables (ver *_loadds*).

Nota: Cuando usamos un controlador de eventos debemos tener en cuenta que el ratón puede interrumpir al programa en cualquier instante. Existe un problema de re-entrada en la in 21h que ademas es una interrupción software que modifica el segmento de pila (SS) durante su ejecución por lo que no conviene usar rutinas de la int 21h durante la ejecución del bucle principal del programa. En particular, si en el bucle principal estamos esperando a que se pulse una tecla por el usuario NO se deben usar las funciones kbhit(), getch(), etc... de C sino que debemos usar las rutinas del teclado ya implementadas en la práctica anterior (int 16h) – Esto es solo necesario para programas que implementen manejadores de eventos de ratón.

En definitiva, la definición del controlador en C debe tener la siguiente forma :

```
#pragma option –N-   //deshabilita el chequeo de la pila.

void far _saveregs _loadds ManejadorEventosRaton(void)
{
   …..
   …..
}
#pragma option –N  //habilitar el chequeo de la pila.
```

REALIZACION PRACTICA

Primera parte de la práctica : (Obligatoria)

En esta parte han de implementar las funciones básicas de programación del ratón.

1.- Implementar la función de reset.
2.- Obtener información del ratón, si es PS/2 inPort, si está en el IRQ3, IRQ4 .... .
3.- Obtener información acerca del tipo de driver instalado para el ratón.
4.- Mostrar y ocultar el puntero del ratón.
5.- Indicar en cada momento donde esta el cursor del ratón (x,y)
6.- Mover el cursor del ratón a una posición determinada sin tocar el periférico.
7.- Definir una ventana en la que se le restrinja el movimiento al ratón, es decir , que fuera de esa ventana no se pueda mover el ratón.
8.- Alterar los parámetros de sensibilidad y de umbral de aceleración.

Segunda parte de la práctica : (Subir nota)

Diseñar e instalar un manejador de ratón que obligue a que cada vez que se pulse un botón muestre impreso en pantalla que botón ha sido presionado.

Esta parte de la practica se puede dividir en dos tareas principales , una el desarrollo de la función que actuara como manejador y otro el de la instalación de dicho manejador.