Seminario 2. Acceso a dispositivos de E/S en C

Duración: 1 sesión

Objetivos

I. Usar un simulador del sistema operativo MS-DOS para programar llamadas a interrupciones de entrada/salida en este S.O. desde el lenguaje C.

II. Utilizar el IDE de Borland C que permite acceder a bajo nivel a las rutinas de interrupción de MS-DOS desde el lenguaje C.

1. Introducción

El sistema operativo MS-DOS (*MicroSoft Disk Operating System*) es un sistema operativo en modo texto para ordenadores de la arquitectura 80x86. Fue el sistema más usado para PCs compatibles con IBM PC en la década de 1980 y mediados de 1990, hasta que fue sustituido gradualmente por sistemas operativos con GUI (interfaz gráfica), en particular por varias generaciones de Microsoft Windows.

MS-DOS es un sistema operativo monousuario y monotarea, con interfaz en modo texto en la cual la comunicación entre el usuario y el sistema operativo se realiza mediante instrucciones formadas por caracteres introducidos desde el teclado.

2. Sistema Básico de Entrada/Salida (BIOS) de un PC

La BIOS (Sistema Básico de Entrada/Salida, *Basic Input/Output System*) de un IBM-PC es un conjunto de programas alojados en una memoria RAM-CMOS dentro de la placa base. Estas rutinas se utilizan durante el arranque del computador para configurar los diferentes dispositivos y arrancar a su vez el sistema operativo.

En un menú típico de la BIOS, se pueden configurar las siguientes características:

- *Main* o *Standard CMOS Features*. Permite cambiar la hora y la fecha, y configurar varias opciones del disco duro u otras unidades de disco. Muestra informaciones sobre la BIOS, la CPU y la memoria.
- *Advanced* o *Advanced BIOS Features*. Permite activar o desactivar las funciones de red (LAN o inalámbrica), el USB, el teclado numérico. Definir el tipo de controlador del disco duro (SATA, IDE). También opciones de la CPU, la memoria o la propia BIOS. Muchas de ellas orientadas a mejorar el rendimiento.
- *Security*. Definir, cambiar o quitar contraseñas para entrar en la configuración de la BIOS o en el sistema.
- *Power* o *Power Management Setup*. Gestionar las características de ahorro de energía del PC. Por ejemplo, si la pantalla o el disco duro deben o no entrar en suspensión. O cómo "despertar" la computadora cuando entra en ese estado.
- *Boot*. Se define la secuencia de arranque. Es decir, desde qué unidades y en qué orden el PC debe buscar un modo de iniciar.

Los PCs recientes sustituyen la BIOS por la llamada UEFI (*Unified Extensible Firmware Interface*), que extiende las capacidades de la BIOS permitiendo arranque en red, inicio selectivo de componentes y, en general, un arranque de los nuevos sistemas operativos más rápido.

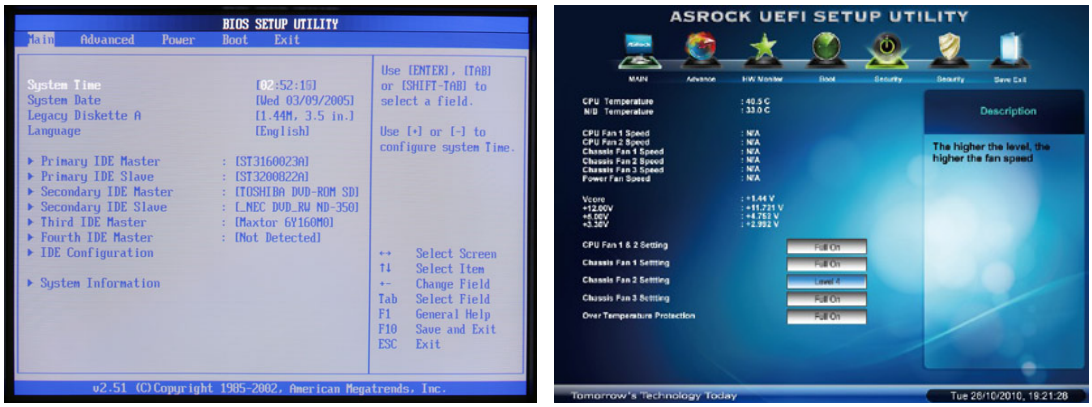

Figura 1. Menú BIOS (izquierda) y UEFI (derecha)

3. Arquitectura 80x86

La arquitectura 80x86 abarca la familia de procesadores Intel de 8, 16 y 32 bits. Le sucedieron las arquitecturas de 64 bits (IA-64). En los procesadores de 16 bits (8086, 80186 y 80286), éste dispone de 14 registros de 16 bits, 4 de ellos de propósito general:

1. AX: Registro de acumulador. Es el único que puede ser usado como multiplicando en la multiplicación y como dividendo en la división. Es fundamental en la programación de interrupciones, ya que selecciona los distintos servicios del Sistema Operativo.
2. BX: Registro de Base.
3. CX: Registro de Contador
4. DX: Registro de Datos

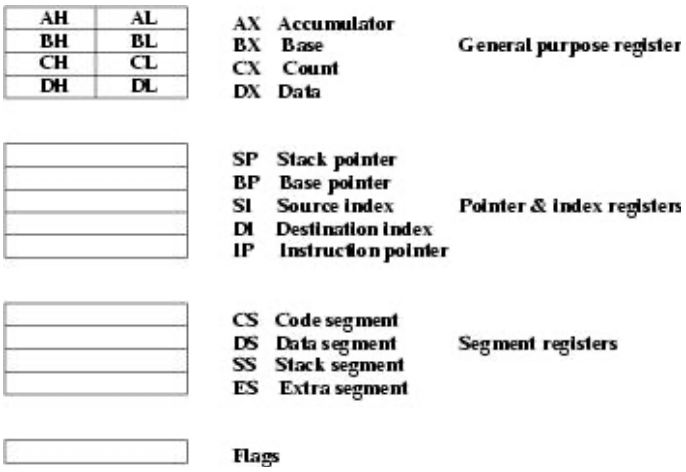

Figura 2. Registros de la arquitectura del procesador 8086

4. Programación de interrupciones para acceder a los dispositivos de E/S

La memoria BIOS de cualquier PC contiene un conjunto de rutinas software para gestionar distintos dispositivos a nivel básico, tales como el teclado, el video, el ratón o las unidades de disco magnético. Estas rutinas son, de hecho, independientes del SO y se ejecutan a partir de interrupciones. Una interrupción es un evento que provoca que la CPU detenga su tarea actual y pase a ejecutar inmediatamente una determinada rutina. La comunicación entre la unidad básica y los periféricos se realiza frecuentemente por medio de estas interrupciones.

Las rutinas de interrupción de la BIOS pueden ser activadas desde un programa a través de la API de MS-DOS (Figura 3). Para ello, es preciso especificar qué rutina debe ejecutarse y los parámetros que ésta precise. Cada interrupción tiene asociado un número de rutina, bajo el cual se aglutinan a su vez a diversas subrutinas o subfunciones. Así pues, para ejecutar una rutina de interrupción específica hay que indicar, por un lado, un número de interrupción y, por otro, el número de subfunción. Además, una subfunción suele necesitar unos determinados parámetros de entrada y quizás devuelva una serie de valores al terminar de ejecutarse. Para pasar los parámetros de entrada se utilizan algunos de los registros internos de la CPU, a los que es necesario asignarle los distintos parámetros de entrada antes de invocar a la rutina de servicio de interrupción (ISR). Si la subfunción devuelve valores, éstos se encuentran en los registros internos justo al acabar de ejecutarse la subfunción.

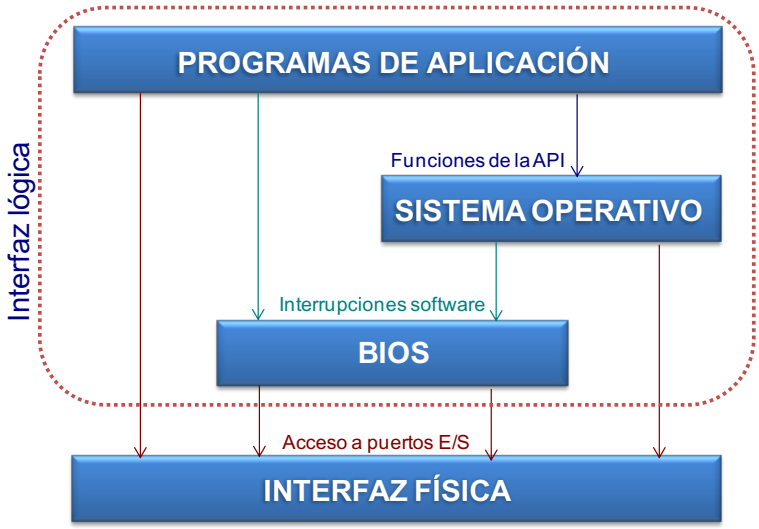

Figura 3. Niveles de acceso a los dispositivos de E/S. El acceso directo a la interfaz física a través de los programas no suele estar permitido por el sistema operativo, ni tampoco a las interrupciones de la BIOS. De esta forma, el sistema operativo se asegura que sólo se puede acceder a los dispositivos a través de su API (*Application Programming Interface*).

5. Emulando MS-DOS con DosBox

Como vimos en el seminario 1, podemos emular el sistema MSDOS bajo cualquier S.O. actual y utilizar programas para aquel sistema. Además, esto nos permite acceder a bajo nivel a cualquier recurso software/hardware, como interrupciones, memoria, puertos de E/S, etc, que de otra forma no podríamos usar, ya que un S.O. actual “se protege” de cierto tipo de accesos que pueden resultar críticos.

6. La librería dos.h

En el seminario 1 vimos cómo hacer uso de las las interrupciones por software desde ensamblador.

Sin embargo, también se pueden realizar en un lenguaje de alto nivel mediante funciones ya implementadas para ello, sin necesidad de recurrir forzosamente al lenguaje ensamblador. En este caso vamos a utilizar la función `int86()`, que viene definida en el fichero `dos.h` de varios compiladores de C.

La sintaxis de dicha función (usando Turbo C o Borland C) es la siguiente:

```
#include <dos.h>
int int86(int intno, union REGS *inregs, union REGS *outregs);
```

El parámetro de entrada `intno` indica el número de interrupción que se desea ejecutar; en `inregs` se especifican los valores de los registros antes de la llamada, y en `outregs` se obtienen los valores de los registros tras ser ejecutada la rutina correspondiente. Tanto `inregs` como `outregs` se declaran como una unión de tipo REGS que está definida como sigue:

```
union REGS {
struct WORDREGS x;
struct BYTEREGS h;
};
```

De esta forma se puede acceder a los registros internos bien como registros de 16 bits, bien como registros de 8 bits. Las estructuras BYTEREGS y WORDREGS están también definidas:

```
struct BYTEREGS  {
    unsigned char  al, ah, bl, bh;
    unsigned char  cl, ch, dl, dh;
};
struct WORDREGS  {
    unsigned int  ax, bx, cx, dx;
    unsigned int  si, di, cflag, flags;
};
```

7. Entorno de desarrollo Borland C

La librería `dos.h` no forma parte del estándar ANSI C. Esto implica que no todos los compiladores de C van a poder trabajar con dicha librería. En este caso, utilizaremos el entorno de programación Borland C 3.1, que sí incluye dicha librería. Este entorno funciona en modo texto y, debido a su antigüedad, carece de las múltiples comodidades que nos facilita un entorno de programación actual como CodeBlocks, Eclipse, Kdevelop, etc.

No obstante, nos permite editar código fuente con resaltado de sintaxis, gestionar proyectos, depurar programas, fijar puntos de ruptura, etc. En cualquier caso, se puede optar por usar el entorno de programación para editar al que se esté acostumbrado y usar Borland C sólo para compilar nuestro proyecto.

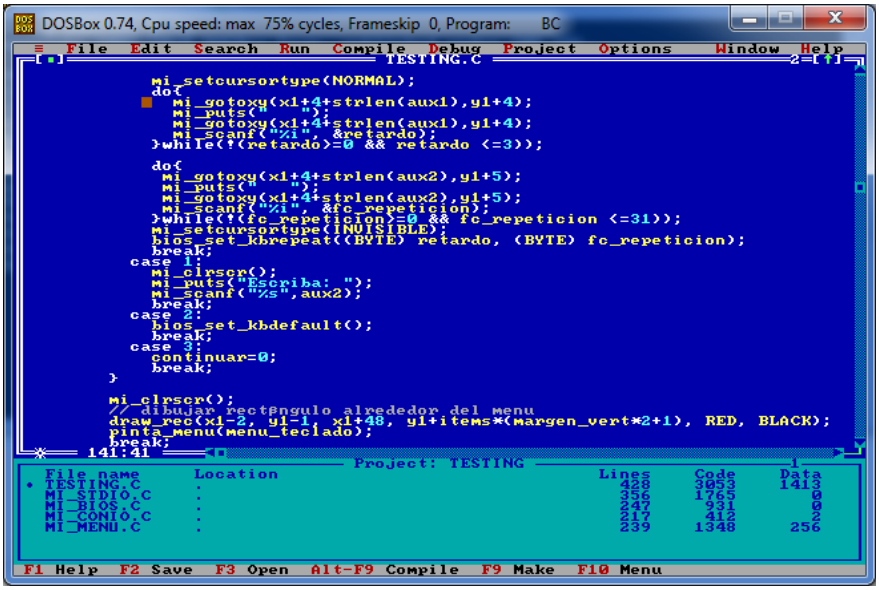

Figura 4. Entorno de programación Borland C

El siguiente ejemplo ilustra el acceso a las interrupciones software desde un programa en C. La comunicación con la tarjeta de vídeo se puede realizar a través de la interrupción número 10h. Asignando distintos valores al registro AH, es posible acceder a diversas subfunciones que afectan a la salida de caracteres por pantalla.

Ejemplo: Seleccionar el modo de vídeo

Número de interrupción: 10h
Número de función: 0
Entrada: AH = 0
AL = modo
Salida: No tiene

Tabla I. Modos de vídeo.

Modo	Tipo	Resolución	Colores
0h	texto	40x25	16 tonos de gris
1h	texto	40x25	16 colores
2h	texto	80x25	16 tonos de gris
3h	texto	80x25	16 colores
4h	gráfico	320x200	4 colores
5h	gráfico	320x200	4 colores
6h	gráfico	640x200	2 colores
7h	texto	80x25	monocromo
dh	gráfico	320x200	16 colores
eh	gráfico	640x200	16 colores
fh	gráfico	640x350	monocromo

10h	gráfico	640x350	16 colores
11h	gráfico	640x480	2 colores
12h	gráfico	640x480	16 colores
13h	gráfico	320x200	256 colores

Función en C que permite modificar el modo de video actual:

```
#include <dos.h>
#define BYTE unsigned char

/* Selecciona el modo de video deseado */
void selecciona_modo_video(BYTE modo){
   union REGS inregs, outregs;
   inregs.h.ah = 0x00;
   inregs.h.al = modo;
   int86(0x10,&inregs,&outregs);
   return;
}
```

Como se puede apreciar en este ejemplo, se llama a la interrupción de video (10h). Se debe asignar al registro *AH* de la unión inregs el valor de la subfunción que se va a ejecutar (0x00). Esta subfunción admite como parámetro de entrada el modo de video que se desea activar y dicho valor se pasa a través del registro *AL*. En este caso, no es preciso devolver ningún valor, así que los valores que tengan los registros de la unión outregs tras ejecutarse esta interrupción no son relevantes.

En el caso de querer leer una pulsación de tecla, podemos usar:

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

int mi_getchar(){
  union REGS inregs, outregs;
  int caracter;

  inregs.h.ah = 1;
  int86(0x21, &inregs, &outregs);

  caracter = outregs.h.al;
  return caracter;
}

int main(){
 int tmp;

 printf("\nPulsa una tecla: ");
 tmp = mi_getchar();

 return 0;
}
```

```
mov ah,1              ;función para leer una tecla
int 21h               ;interrupción MSDOS

 ;en AL devuelve el carácter tecleado
```

programando en C usando la librería dos.h

```c
int mi_getchar(){
    union REGS inregs, outregs;
    int caracter;

    inregs.h.ah = 1;
    int86(0x21, &inregs, &outregs);

    caracter = outregs.h.al;

    return caracter;
}

void mi_putchar(char c){
    union REGS inregs, outregs;

    inregs.h.ah = 2;
    inregs.h.dl = c;

    int86(0x21, &inregs, &outregs);
}
```

programando en ensamblador (x86, 16bits)

Esperar la pulsación de una tecla mostrándola por pantalla

```
mov ah,1                ;función para leer una tecla
int 21h                 ;interrupción MSDOS

 ;en AL devuelve el carácter tecleado
```

Escribir un carácter en pantalla

```
mov dl,CARACTER         ;código ASCII del carácter a escribir
mov ah,2                ;función para escribir un carácter
int 21h                 ;interrupción MSDOS
```

8. Ejercicios

Partiendo de la configuración de DOSBox realizada en el seminario anterior, teniendo ya montada en la unidad C: el directorio donde se encuentra el entorno de programación Borland C y probada la configuración para asegurarnos que podemos compilar nuestros programas, se propone hacer los siguientes ejercicios:

1. Realizar un programa en lenguaje C que llame a la función de cambio de modo de vídeo y comprobar su funcionamiento.

2. Realizar un programa en lenguaje C que llame a la función de interrupción correspondiente a la lectura de caracteres desde teclado. Ejecutarlo para leer pulsaciones de teclas y mostrarlas por pantalla usando otra función de interrupción.

Forma de Entrega:

La práctica o seminario podrá realizarse de manera individual o por grupos de hasta 2 personas.

Se entregará como un archivo de texto en el que se muestre la información requerida. También se puede utilizar la sintaxis de Markdown para conseguir una mejor presentación e incluso integrar imágenes o capturas de pantalla. La entrega se realizará subiendo los archivos necesarios al repositorio "**PDIH"** en la cuenta de GitHub del estudiante, a una carpeta llamada "**S2**".

Toda la documentación y material exigidos se entregarán en la fecha indicada por el profesor. No se recogerá ni admitirá la entrega posterior de las prácticas/seminarios ni de parte de los mismos.

La detección de copias implicará el suspenso inmediato de todos los implicados en la copia (tanto de quien realizó el trabajo como de quien lo copió).

Las faltas de ortografía se penalizarán con hasta 1 punto de la nota de la práctica o seminario.

9. Enlaces y recursos

https://www.dosbox.com/DOSBoxManual.html

https://www.linuxadictos.com/dosbox-en-linux.html

https://es.wikihow.com/usar-DOSBox

http://ubuntudriver.blogspot.com/2011/09/instalacion-basica-de-dosbox-en-ubuntu.html

https://www.dosgamers.com/es/dos/dosbox-dos-emulator/screen-resolution

https://www.enmimaquinafunciona.com/pregunta/168127/aumentar-el-tamano-de-la-ventana-de-dosbox