USO DE LOS RECURSOS DE UN EQUIPO DE CÓMPUTO MEDIANTE C++

Bruno López Takeyas
Instituto Tecnológico de Nuevo Laredo
Reforma Sur 2007, C.P. 88250, Nuevo Laredo, Tamps. México
http://www.itnuevolaredo.edu.mx/takeyas
E-mail: takeyas@itnuevolaredo.edu.mx

Resumen: *En esta sección se examinarán varias funciones especiales de Turbo C++ que permiten que los programas accedan a los recursos de la computadora mediante las funciones BIOS o DOS.*

Cada procesador, sistema operativo y entorno tiene sus propios métodos de acceder a los recursos del sistema. Para efectos didácticos se asumirá el uso del sistema operativo PC-DOS y la familia de procesadores 8086.

1. Registros de la CPU

La familia de procesadores 8086 tiene 14 registros en los que se pone la información para procesar o el programa de control. Los registros pueden ser de las categorías siguientes:

Registros de propósito general: Son de trabajo de la CPU. En estos registros se colocan los valores para su procesamiento que incluye operaciones aritméticas, comparaciones e instrucciones de bifurcación o saltos.

Registros de base de puntero e índice: Se usan para proporcionar soporte a cosas como direccionamiento relativo, apuntador de pila e instrucciones para mover bloques.

Registros de segmento: Son usados para soportar el esquema de memoria segmentada. El registro CS guarda el segmento de código actual, el DS el segmento actual de datos, el ES el segmento extra y el SS el segmento de pila.

Registros de propósito especial: Guardan el estado de la CPU y el apuntador de instrucciones que indica la siguiente instrucción que ejecutará la CPU.

Registros de Propósito General					
	AH	AL		CH	CL
AX			CX		
	BH	BL		DH	DL
BX			DX		
Registros de puntero e índice					
SP			SI		
	Puntero de pila			Índice fuente	
BP			DI		
	Puntero base			Índice destino	
Registros de segmento					
CS			SS		
	Segmento de código			Segmento de pila	
DS			ES		
	Segmento de datos			Segmento extra	
Registro de propòsito especial					
			IP		
	Registro de indicadores			Puntero de instrucciones	

Fig. 1. Registros de la CPU

2. Interrupciones

Una interrupción es un tipo especial de instrucción que provoca la parada de la ejecución del programa, guarda el estado actual del sistema en la pila y salta a una rutina de manejo de la interrupción que se determina por el número de la interrupción. Después de que acabe la rutina, realiza una vuelta a la interrupción que provoca que se reanude la ejecución del programa anterior.

Hay dos tipos básicos de interrupciones: las generales por hardware y las provocadas por software. La CPU permite a un programa ejecutar una interrupción software por la instrucción **INT**. El número que sigue a la instrucción determina el número de la interrupción. Por ejemplo, **INT 21h** provoca la interrupción **21h**.

Muchas de estas interrupciones son usadas por el BIOS o el DOS como un medio de acceder a diversas funciones que son parte del sistema operativo. Cada interrupción se asocia a una categoría de servicios a las que accede que son determinados por el valor del registro AH. Si se necesita información adicional se pasa en los registros AL, BX, CX y DX.

3. Función int86()

La función ***int86()*** de Turbo C++ se usa para ejecutar una interrupción de software. Se declara como

*int86(int **intnum**, union REGS ***in**, union REGS ***out**)*

El número de la interrupción en esta función es **intnum**, **in** es una unión que contiene los registros que se usarán para pasar la información a los manejadores de la interrupción y **out** es una unión que guardará los valores devueltos por la interrupción (si los hay).

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

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

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

Fig. 2.- Tipo REGS incluido en el archivo de encabezado DOS.H

Como se puede observar en la Fig. 2, REGS es una unión de dos estructuras que contiene el archivo de encabezado DOS.H. Usar la estructura WORDREGS permite acceder a los registros de la CPU como cantidades de 16 bits. El uso de BYTEREGS da acceso a los registros de 8 bits.

4. Desapareciendo el cursor

Se puede desaparecer el cursor si se coloca un 1 en el 5º bit del registro CH (Fig. 3). Para colocar el 5º bit con el valor de 1, se coloca el número hexadecimal 20 en el registro CH, lo cual equivale a 00100000 en binario, lo cual indica el valor que se necesita.

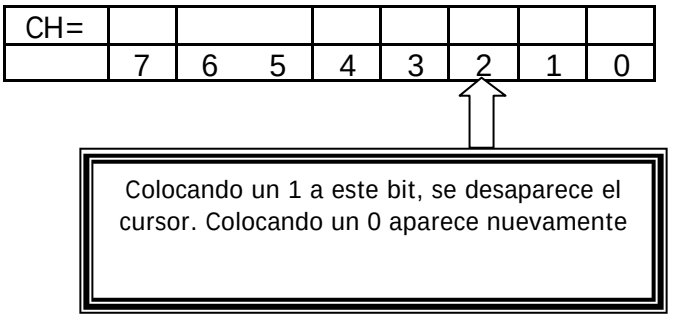

Fig. 3.- Bit del cursor

```
/*
 Programa para desaparecer el cursor
usando la funcion int86()


 Instructor: M.C. Bruno Lopez Takeyas
*/

#include <dos.h>  // Encabezado con la
definicion de int86()
#include <conio.h>
#include <iostream.h>
```

```
void main(void)
{
union REGS regs;  // Declaracion de la
union regs de tipo REGS para el uso de
              // los registros de la CPU

  clrscr();

  cout << "\n\rDesapareciendo el cursor
...";

  regs.h.ch=0x20;  // Inicializar el
registro CH con el valor 20(hexadecimal)
               // equivalente a
00100000; es decir colocar un 1 en el
               // 5o. bit del registro
CH para desaparecer el cursor

  regs.h.ah=1;   // Servicio 1 de la INT
10h (video) que se refiere al tama¤o
             // del cursor

  int86(0x10,&regs,&regs); // Invoca la
INT 10h (video)

  cout << "\n\n\n\n\r<<< Oprima
cualquier tecla para aparecer el cursor
>>>";
  getch();
  cout << "\n\n\n\n\n\rApareciendo el
cursor ...";

  regs.h.ch=0;  // Inicializar el
registro CH con el valor 0(hexadecimal)
               // equivalente a
00000000; es decir colocar un 0 en el
               // 5o. bit del registro
CH para aparecer el cursor

  regs.h.ah=1;   // Servicio 1 de la INT
10h (video) que se refiere al tama¤o
             // del cursor

  int86(0x10,&regs,&regs); // Invoca la
INT 10h (video)

  cout << "\n\n\n\n\r<<< Oprima
cualquier tecla para aparecer el cursor
>>>";
  getch();

  return;
}
```

Fig. 4.- Desapareciendo el cursor usando la función int86().

5. Aplicaciones usando el mouse

Se pueden desarrollar aplicaciones en las que se incluya el manejo mediante el uso del mouse a través de la INT 33h con sus servicios correspondientes. El programa de la Fig. 5 muestra un ejemplo de la forma de implementar el mouse.

```
/*
 Programa para usar el mouse

Instructor: M.C. Bruno Lopez Takeyas
*/

#include <dos.h>  // Encabezado con la
definicion de int86()
#include <conio.h>
#include <iostream.h>


void main(void)
{
union REGS regs;  // Declaracion de la
union regs de tipo REGS para el uso de
              // los registros de la CPU

  clrscr();
  gotoxy(1,24);  cout << "Oprima
cualquier tecla para salir";

  regs.h.al=0x1;  // Inicializar el
registro AL con el valor 1(hexadecimal)
               // es decir, invoca el
servicio 1 de la INT 33h (mouse)
               // para habilitar el
mouse

  int86(0x33,&regs,&regs); // Invoca la
INT 33h (mouse)

  while(!kbhit())  // Ciclo mientras NO
se oprima cualquier tecla
```

```
  {

    regs.h.al=0x3;  // Inicializar el
registro AL con el valor 3(hexadecimal)
                // es decir, invoca el
servicio 3 de la INT 33h (mouse)
                // para detectar el
status del mouse, o sea, si se
                // oprime algun boton y
las coordenadas del cursor

    int86(0x33,&regs,&regs);  // Invoca
la INT 33h (mouse)

// El servicio 3 de la INT 33h devuelve
el status del mouse en el registro BX
// Si BX=1 entonces se oprimio el boton
izquierdo
// Si BX=2 entonces se oprimio el boton
derecho

    if(regs.x.bx==1)
    {
      gotoxy(20,12); cout << "Boton
izquierdo";
      gotoxy(20,12); cout << "
";
    }

    if(regs.x.bx==2)
    {
      gotoxy(40,12); cout << "Boton
derecho";
      gotoxy(40,12); cout << "
";
    }

  }
  return;
}
```

Fig. 5.- Manejo del mouse

El programa de la Fig. 5 inicializa primero el mouse invocando la INT 33h con el servicio AL=1. Contiene un ciclo que mientras no se oprima cualquier tecla, monitorea constantemente el status del mouse invocando la INT 33h con el servicio AL=3, la cual devuelve en BX=1 si se oprimió el botón izquierdo o en BX=2 si se oprimió el botón derecho.

En la Fig. 5 se muestra un encabezado (LIBMOUSE.h) que contiene las declaraciones necesarias y la mayoría de los servicios necesarios para utilizar el mouse en un programa. Sólo basta incluirlo como mediante la siguiente línea de código:

```
#include “LIBMOUSE.h”
```

Las rutinas contenidas en este encabezado son:

```
void zMouseDetect(struct mdata *Mptr);
/* INICIALIZAR SI SE ENCUENTRA   */
void zMouseInit(struct mdata *Mptr);
/* INICIALIZA EL DRIVER          */
void zMouseShow(struct mdata *Mptr);
/* MUESTRA EL CURSOR             */
void zMouseHide(struct mdata *Mptr);
/* ESCONDE EL CURSOR             */
void zMousePos(struct mdata *Mptr);
/* COLOCA LAS COORDENADAS x,y    */
void zMouseInfo(struct mdata *Mptr);
/* CHECA LA POSICION Y EL BOTON  */
void zMouseHLimit(struct mdata *Mptr);
/* COLOCA EL RANGO HORIZONTAL    */
void zMouseVLimit(struct mdata *Mptr);
/* COLOCA EL RANGO VERTICAL      */
```

```
/* LIBRERIA
.....................................
........ LIBMOUSE.H
.
   Instituto Tecnol¢gico de Nuevo Laredo
.                              */

#define zCOMPILER zTURBOC
#define zMOUSE 0x33
#include <dos.h>
#if zCOMPILER==zQUICKC
    static union REGS inregs,outregs;
#elif zCOMPILER==zTURBOC
    static union REGS regs;
/* PARA EL CONTENIDO DE REGISTROS*/
#endif
```

```
#ifdef SII
struct mdata                        /*
ESTRUCTURA DE DATOS DE MOUSE  */
{
  int MouseFlag;
  int MouseButton;
  int MouseX;
  int MouseY;
  int MouseMinX;
  int MouseMaxX;
  int MouseMinY;
  int MouseMaxY;
}MouseParams;
#endif SII

void zMouseDetect(struct mdata *Mptr);
/* INICIALIZAR SI SE ENCUENTRA   */
void zMouseInit(struct mdata *Mptr);
/* INICIALIZA EL DRIVER          */
void zMouseShow(struct mdata *Mptr);
/* MUESTRA EL CURSOR             */
void zMouseHide(struct mdata *Mptr);
/* ESCONDE EL CURSOR             */
void zMousePos(struct mdata *Mptr);
/* COLOCA LAS COORDENADAS x,y    */
void zMouseInfo(struct mdata *Mptr);
/* CHECA LA POSICION Y EL BOTON  */
void zMouseHLimit(struct mdata *Mptr);
/* COLOCA EL RANGO HORIZONTAL    */
void zMouseVLimit(struct mdata *Mptr);
/* COLOCA EL RANGO VERTICAL      */


/* DECLARACION DE ALGUNAS VARIABLES :
*/

/* FUNCION DE ALTO NIVEL PARA
INICIALIZAR EL MOUSE SI ESTA PRESENTE :
*/
void zMouseDetect(struct mdata *Mptr)
{
  zMouseInit(Mptr);
/* INTENTA INICIALIZAR EL MOUSE  */
  if(Mptr->MouseFlag==0) return;
/* ABORTAR SI NO SE ENCUENTRA    */
  zMouseHLimit(Mptr);
  zMouseVLimit(Mptr);
  zMousePos(Mptr);
}

/* PRE-INICIALIZACION DE MOUSE A BAJO
NIVEL :                             */
void zMouseInit(struct mdata *Mptr)
{
  #if zCOMPILER==zQUICKC
    inregs.x.ax=0;
    int86(zMOUSE,&inregs,&outregs);
    Mptr->MouseFlag=outregs.x.ax;
  #elif zCOMPILER==zTURBOC
    regs.x.ax=0;
/* FUNCION #0 DEL MOUSE          */
    int86(zMOUSE,&regs,&regs);
/* LLAMA LA INTERRUPCION 33 HEX  */
    Mptr->MouseFlag=regs.x.ax;
/* IGUALA A 0 SI NO ESTA PRESENTE*/
  #endif
  return;
}

/* FUNCION PARA MOSTRAR EL CURSOR DEL
MOUSE :                             */
void zMouseShow(struct mdata *Mptr)
{
  #if zCOMPILER==zQUICKC
    inregs.x.ax=1;
    int86(zMOUSE,&inregs,&outregs);
  #elif zCOMPILER==zTURBOC

    regs.x.ax=1;
/* FUNCION #1 DEL MOUSE          */
    int86(zMOUSE,&regs,&regs);
/* LLAMA LA INTERRUPCION 33 HEX  */
  #endif
  return;
}

/* FUNCION PARA NO MOSTRAR EL CURSOR DEL
MOUSE :                             */
void zMouseHide(struct mdata *Mptr)
{
  #if zCOMPILER==zQUICKC
    inregs.x.ax=2;
    int86(zMOUSE,&inregs,&outregs);
  #elif zCOMPILER==zTURBOC
    regs.x.ax=2;
/* FUNCION #2 DEL MOUSE          */
    int86(zMOUSE,&regs,&regs);
/* LLAMA LA INTERRUPCION 33 HEX  */
  #endif
  return;
}

/* FUNCION PARA DETERMINAR LA UBICACION
DEL MOUSE Y EL ESTADO DE BOTONES :*/
void zMouseInfo(struct mdata *Mptr)
{
  #if zCOMPILER==zQUICKC
    inregs.x.ax=3;
    int86(zMOUSE,&inregs,&outregs);
    Mptr->MouseButton=outregs.x.bx;
    Mptr->MouseX=outregs.x.cx;
    Mptr->MouseY=outregs.x.dx;
  #elif zCOMPILER==zTURBOC
```

```
    regs.x.ax=3;
/* FUNCION #3 DEL MOUSE          */
    int86(zMOUSE,&regs,&regs);
/* LLAMA LA INTERRUPCION 33 HEX  */
    Mptr->MouseButton=regs.x.bx;
/* 1 (IZQUIERDO) ¢ 2 (DERECHO)   */
    Mptr->MouseX=regs.x.cx;
/* OBTIENE LA COORDENADA EN x    */
    Mptr->MouseY=regs.x.dx;
/* OBTIENE LA COORDENADA EN y    */
  #endif
  return;
}

/* FUNCION PARA BORRAR LA UBICACION DEL
CURSOR DEL MOUSE :                */
void zMousePos(struct mdata *Mptr)
{
  #if zCOMPILER==zQUICKC
    inregs.x.ax=4;
    inregs.x.cx=Mptr->MouseX;
    inregs.x.dx=Mptr->MouseY;
    int86(zMOUSE,&inregs,&outregs);
  #elif zCOMPILER==zTURBOC
    regs.x.ax=4;
/* FUNCION #4 DEL MOUSE          */
    regs.x.cx=Mptr->MouseX;
/* COLOCA LA COORDENADA EN x     */
    regs.x.dx=Mptr->MouseY;
/* COLOCA LA COORDENADA EN y     */
    int86(zMOUSE,&regs,&regs);
/* LLAMA LA INTERRUPCION 33 HEX  */
  #endif
  return;
}

/* FUNCION PARA COLOCAR EL RANGO
HORIZONTAL MINIMO Y MAXIMO :
*/
void zMouseHLimit(struct mdata *Mptr)
{

  #if zCOMPILER==zQUICKC
    inregs.x.ax=7;
    inregs.x.cx=Mptr->MouseMinX;
    inregs.x.dx=Mptr->MouseMaxX;
    int86(zMOUSE,&inregs,&outregs);
  #elif zCOMPILER==zTURBOC
    regs.x.ax=7;
/* FUNCION #7 DEL MOUSE          */
    regs.x.cx=Mptr->MouseMinX;
/* COLOCA LA MINIMA COORDENADA x */
    regs.x.dx=Mptr->MouseMaxX;
/* COLOCA LA MAXIMA COORDENADA x */
    int86(zMOUSE,&regs,&regs);
/* LLAMA LA INTERRUPCION 33 HEX  */
  #endif
  return;
}

/* FUNCION PARA COLOCAR EL RANGO
VERTICAL MINIMO Y MAXIMO :
*/
void zMouseVLimit(struct mdata *Mptr)
{
  #if zCOMPILER==zQUICKC
    inregs.x.ax=8;
    inregs.x.cx=Mptr->MouseMinY;
    inregs.x.dx=Mptr->MouseMaxY;
    int86(zMOUSE,&inregs,&outregs);
  #elif zCOMPILER==zTURBOC
    regs.x.ax=8;
/* FUNCION #8 DEL MOUSE          */
    regs.x.cx=Mptr->MouseMinY;
/* COLOCA LA MINIMA COORDENADA y */
    regs.x.dx=Mptr->MouseMaxY;
/* COLOCA LA MAXIMA COORDENADA y */
    int86(zMOUSE,&regs,&regs);
/* LLAMA LA INTERRUPCION 33 HEX  */
  #endif
  return;
}
```

Fig. 6.- Encabezado LIBMOUSE.H

6. Bibliografía

? Barkakati Nabajyoti. **“The Waite Group´s. Turbo C Bible”**. Howard W. Sams & Company. Estados Unidos. 1990.

? García Badell, J. Javier. **"Turbo C. Programación en manejo de archivos"**. Macrobit.

? Deitel y Deitel. **“C++ Cómo programar”**. Segunda edición. Pearson-Prentice Hall. Estados Unidos. 1999.

? Lafore, Robert. **“The Waite Group´s. Turbo C. Programming**

for the PC". Revised Edition. Howard W. Sams & Company. Estados Unidos. 1990.

? Schildt, Herbert. **"Turbo C, Programación avanzada"**. Segunda edición, McGraw Hill. Estados Unidos. 1990.

? Staugaard, Andrew. **"Técnicas estructuradas y orientadas a objetos. Una introducción utilizando C++"**. Segunda edición. Prentice Hall. Estados Unidos. 1998.