Práctica 1. Introducción a la programación de periféricos

Duración: 2 sesiones

1. Objetivos de la práctica

Los objetivos concretos de esta práctica son:

- Usar un simulador del sistema operativo MS-DOS para poder programar las llamadas a interrupciones de entrada/salida a bajo nivel.
- Acceder a bajo nivel (ensamblador) a los recursos del microprocesador para realizar E/S.

2. Introducción

Un programa ensamblador para MSDOS sigue una estructura en tres segmentos, pila, datos y código, seguidos de una última línea con la directiva al compilador para declarar el punto de entrada al programa (función por la que se empieza a ejecutar el programa).

A modo de ejemplo muy sencillo, se muestra a continuación un programa que muestra una cadena de texto por pantalla. La sintaxis es muy estricta (las líneas en negrita son fijas):

```
pila segment stack 'stack'
       dw 100h dup (?)
pila ends

datos segment 'data'
       msg db 'hola$'
datos ends

codigo segment 'code'
       assume cs:codigo, ds:datos, ss:pila
       main PROC
              mov ax,datos
              mov ds,ax

              mov dx,OFFSET msg  ; mostrar por pantalla una cadena de texto
              mov ah,9
              int 21h

              mov ax,4C00h    ; terminar y salir al S.O.
              int 21h

       main ENDP
codigo ends

END main
```

Una vez hemos introducido estas instrucciones en un archivo de texto plano, con extensión .asm, lo compilaremos en dos pasos (compilación y enlazado). La ejecución se hace llamando al nuevo archivo con extensión .EXE recién creado.

Como vemos, cada tarea en el programa la hemos llevado a cabo haciendo una llamada a una función a interrupción. Las líneas en verde hacen una llamada a la función de la interrupción 21h (MSDOS) que muestra por pantalla (salida) una cadena

de texto que le indiquemos. Por su parte, las líneas en azul hacen una llamada a la función de la interrupción 21h (MSDOS) que terminan el programa y salen al S.O.

De esta forma, si quisiéramos leer de teclado (entrada), sólo tenemos que incluir en el punto concreto del programa las instrucciones para hacer la llamada a interrupción para esperar una pulsación de tecla.

3. Funciones de interrupción para hacer E/S convencional

Para realizar E/S por los dispositivos convencionales (pantalla, teclado y ratón), podemos usar las siguientes funciones de interrupción. No son las únicas, ya que tanto la BIOS como el MSDOS ofrecen diferentes funciones para hacer las mismas tareas sencillas (lectura de una tecla, mostrar un carácter, etc).

terminar el programa y salir al MSDOS

```
mov ah, 4Ch ; función de terminar el programa y salir
mov al, 00h  ; código de salida al S.O.
int 21h
```

escribir una cadena (terminada en $) por pantalla

```
mov dx, OFFSET cadena
mov ah, 9
int 21h
```

escribir el carácter CARACTER en pantalla (int 21h)

```
mov dl,CARACTER
mov ah,2
int 21h
```

esperar la pulsación de una tecla sin mostrarla por pantalla

```
mov ah,08h   ;función para leer una tecla
int 21h    ;interrupción DOS para teclado
;en AL devuelve el carácter tecleado
```

esperar la pulsación de una tecla mostrándola por pantalla

```
mov ah,01h   ;función para leer una tecla
int 21h    ;interrupción DOS para teclado
;en AL devuelve el carácter tecleado
```

colocar el cursor en modo texto en la coordenada x,y (int 10h)

```
mov dl,columna   ; dl=columna
mov dh,fila      ; dl=fila
mov bh,0
mov ah,2    ;función para posicionar el cursor
int 10h     ;interrupción BIOS para pantalla
```

poner el MODO gráfico y pintar un punto en X,Y de COLOR específico:

MODO = 0 – texto 40x25 b/n
1 - texto 40x25 color
2 – texto 80x25 b/n
3 – texto 80x25 color
4 – gráfico 320x200 con cuatro colores (CGA)
5 – gráfico 320x200 b/n
6 – gráfico 640x200 b/n

```
mov al,MODO
mov ah,0  ;función para poner modo texto o gráfico deseado
int 10h
```

dibujar un pixel de un color determinado en la coordenada x,y indicada en la pantalla gráfica (es necesario haber establecido un modo gráfico de pantalla):

```
mov cx,X      ;columna
mov dx,Y      ;fila
mov al,COLOR
mov ah,0Ch
int 10h
```

Como ejemplo, veamos cómo hacer una pausa esperando una pulsación de tecla:

```
pila segment stack 'stack'
        dw 100h dup (?)
pila ends

datos segment 'data'
        cadena db 13,10,'pulsa una tecla...',13,10,'$'
datos ends

codigo segment 'code'
      assume cs:codigo, ds:datos, ss:pila
      main PROC
            mov ax,datos
            mov ds,ax

            mov dx,OFFSET cadena
            mov ah,9
            int 21h

            mov ah,8
            int 21h

            mov ax,4C00h
            int 21h
      main ENDP

codigo ends
END main
```

Cuestiones a resolver

El objetivo principal de esta práctica es usar un simulador de S.O. para acceder a bajo nivel a los recursos del microprocesador, BIOS y S.O. para realizar E/S.

A partir de los programas realizados en el seminario-1, se desarrollarán los siguientes programas en lenguaje ensamblador:

1. programa que use la función de interrupción de cambio de modo de vídeo (usar el modo texto 40x25) y mostrar una cadena de texto para comprobar el funcionamiento. Antes de terminar, hacer una pausa esperando la pulsación de una tecla, y restaurar el modo de vídeo de 80x25.
2. programa que use la función de interrupción de cambio de modo de vídeo (usar el modo gráfico 320x200 color) y dibujar un rectángulo blanco en pantalla; a continuación, mostrar varios pixels de diferentes colores dentro del rectángulo.

Antes de terminar, hacer una pausa esperando la pulsación de una tecla, y restaurar el modo de vídeo de 80x25.

Como resultado de hacer la práctica 1 **se mostrará** al profesor el funcionamiento de cada programa propuesto.

En el documento a entregar se describirá cómo se ha realizado la configuración del simulador, se detallará cada programa desarrollado, y se mostrarán imágenes mostrando el funcionamiento correcto de cada programa.

Normas de entrega

La práctica/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 "**P1**".

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 prácticas copiadas 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/seminario.

Referencias

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://issuu.com/jorgjls/docs/manual_b__sico_para_ensamblador_f
http://www.nachocabanes.com/tutors/asmperif.htm
http://irlenys.tripod.com/microii/interr.htm
http://ict.udlap.mx/people/oleg/docencia/ASSEMBLER/asm_interrup_21.html

http://www.nachocabanes.com/tutors/asmperif.htm
http://ict.udlap.mx/people/oleg/docencia/ASSEMBLER/asm_interrup_21.html
http://irlenys.tripod.com/microii/interr.htm