

SISTEMAS BASADOS EN MICROPROCESADORES

Grado en Ingeniería Informática

ENUNCIADO PROBLEMA 1

A continuación se incluye el código de una aplicación formada por un programa principal escrito en lenguaje C, así como de subrutinas o funciones y de un programa residente escritos en lenguaje ensamblador del 8086. En el código se han distribuido un conjunto de marcadores numéricos que permiten situar al alumno en distintas zonas de dicho programa sobre los cuáles se realizan preguntas. El programa implementa un reloj digital que muestra la hora y los minutos en la pantalla del PC.

Programa Principal

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

extern int far DetectarDriver ();
extern void far DesinstalarDriver ();
extern void far ActivarReloj ();
extern void far PararReloj ();
extern void far ActualizarReloj (int hora, int minutos, int segundos);

void main()
{
    char c;
    int salir;
    int hora, minutos, segundos; (P9)

    if (DetectarDriver()==1) (P6)
    {
        printf("Driver no instalado.\n");
        exit(-1);
    }

    salir = 0;
    while( !salir )
    {
        c = getch();
        if (c=='a' || c=='A')
        {
            printf("Introduzca la hora (0-23): ");
            scanf("%d", &hora);
            printf("Introduzca los minutos (0-59): ");
            scanf("%d", &minutos);
            printf("Introduzca los segundos (0-59): ");
            scanf("%d", &segundos);
            PararReloj();
            ActualizarReloj( hora, minutos, segundos ); (P1) (P2)
        }
    }
}
```

```

        ActivarReloj();
    }

    if (c=='s' || c=='S') salir = 1;
}

DesinstalarDriver();
printf("Saliendo del programa de control del reloj.\n");
}

```

Rutinas en Ensamblador

```

_text segment byte public 'code'
    assume cs:_text

_DetectarDriver proc far
    push es
    xor ax, ax
    mov es, ax
    cmp word ptr es:[61h*4], 0
    jne _detectar_int
    cmp word ptr es:[61h*4+2], 0
    je _detectar_nodriver
_detectar_int:
    mov ah, 00h (P6)
    int 61h
    cmp ax, 0F0F0h
    jne _detectar_nodriver
    xor ax, ax
    jmp _detectar_fin
_detectar_nodriver:
    mov ax, 1
_detectar_fin:
    pop es
    ret
_DetectarDriver endp

_DesinstalarDriver proc far
    push ax
    mov ah, 01h
    int 61h
    pop ax
    ret
_DesinstalarDriver endp

_PararReloj proc far
    push ax
    mov ah, 02h
    int 61h
    pop ax
    ret
_PararReloj endp

```

```

_ActivarReloj proc far
    push ax
    mov ah, 03h
    int 61h
    pop ax
    ret
_ActivarReloj endp

_ActualizarReloj proc far (P1) (P2) (P3)
    push bp
    push bx
    push cx
    push dx
    mov bp, sp
    push ax

    mov bx, bp[12] (P2) (P3)
    mov cx, bp[14] (P2) (P3)
    mov dx, bp[16] (P2) (P3)

    mov ah, 04h (P1 <-)
    int 61h

    pop ax
    pop dx
    pop cx
    pop bx
    pop bp
    ret
_ActualizarReloj endp

public _DetectarDriver
public _DesinstalarDriver
public _PararReloj
public _ActivarReloj
public _ActualizarReloj

_text ends
end

```

Driver (archivo .com)

```

code segment
    assume cs:code

    org 100h

driver_start:
    jmp instalar

;Variables del driver
    old_61h      dw 0, 0
    old_1Ch      dw 0, 0
    estado_reloj db 1 (P9)
    contador     dw 0
    hora         dw 0 (P9)

```

```

        minutos        dw 0 (P9)
        segundos       dw 0 (P9)

;Reloj del driver
reloj_driver proc far
    sti
    cmp estado_reloj, 1
    jne reloj_driver_fin

    inc contador
    cmp contador, 18 (P7)
    jne pintar_reloj

    mov contador, 0
    inc segundos
    cmp segundos, 60
    jne pintar_reloj

    mov segundos, 0
    inc minutos
    cmp minutos, 60
    jne pintar_reloj

    mov minutos, 0
    inc hora
    cmp hora, 24
    jne pintar_reloj

    mov hora, 0

pintar_reloj:
    call mostrar_reloj

reloj_driver_fin:
    iret
reloj_driver endp

;Interfaz de comunicación con el driver
interfaz_driver proc far (P3)
    sti

    cmp ah, 00h
    jne driver_desinstalar

    mov ax, 0F0F0h
    jmp driver_fin

driver_desinstalar:
    cmp ah, 01h
    jne driver_parar_reloj

    call desinstalar
    jmp driver_fin

driver_parar_reloj:
    cmp ah, 02h

```

```

        jne driver_activar_reloj

        mov estado_reloj, 0
        jmp driver_fin

driver_activar_reloj:
        cmp ah, 03h
        jne driver_actualizar_reloj

        mov estado_reloj, 1
        jmp driver_fin

driver_actualizar_reloj: (P3)
        cmp ah, 04h
        jne driver_fin

        mov hora, XX (P4)
        mov minutos, XX (P4)
        mov segundos, XX (P4)
        mov contador, 0

driver_fin:
        iret
interfaz_driver endp

;Rutinas auxiliares del driver

;Rutina para mostrar el reloj por pantalla (hh:mm:ss)
mostrar_reloj proc near
        .....
        ret
mostrar_reloj endp

;Rutinas de instalación / desinstalación del driver

;Funcion que desinstala el driver
desinstalar proc near
        push ax
        push es
        xor ax, ax
        mov es, ax

        cli

        ;Vector 1Ch
        mov ax, old_1Ch
        mov es:[1Ch*4], ax
        mov ax, old_1Ch+2
        mov es:[1Ch*4+2], ax

        ;Vector 61h
        mov ax, old_61h
        mov es:[61h*4], ax
        mov ax, old_61h+2
        mov es:[61h*4+2], ax

        sti

```

```

        mov es, cs:[2Ch]
        mov ah, 49h
        int 21h

        mov ax, cs
        mov es, ax
        mov ah, 49h
        int 21h

        pop es
        pop ax
        ret
desinstalar endp

;Funcion que instala el driver
instalar proc near
        xor ax, ax
        mov es, ax

        cli

        mov ax, es:[1Ch*4]
        mov old_1Ch, ax
        mov ax, es:[1Ch*4+2]
        mov old_1Ch+2, ax

        mov ax, es:[61h*4]
        mov old_61h, ax
        mov ax, es:[61h*4+2]
        mov old_61h+2, ax

        mov es:[1Ch*4], offset reloj_driver (P7)
        mov es:[1Ch*4+2], cs

        mov es:[61h*4], offset interfaz_driver
        mov es:[61h*4+2], cs

        sti

        mov dx, offset instalar
        int 27h
instalar endp

code ends
end driver_start

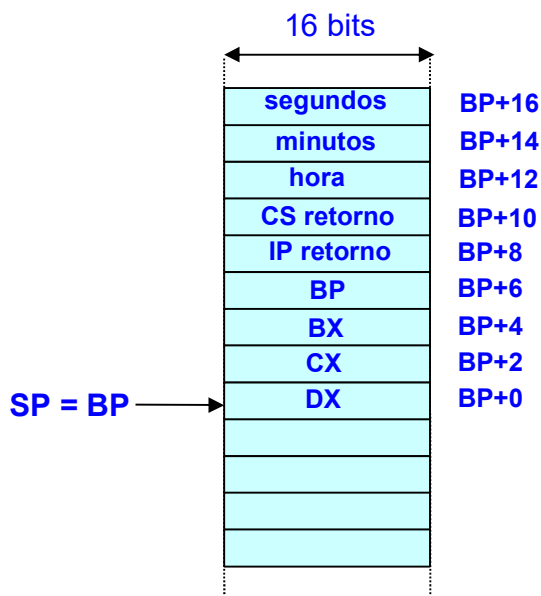
```

SISTEMAS BASADOS EN MICROPROCESADORES

Grado en Ingeniería Informática

PREGUNTAS PROBLEMA 1

P1. A la vista del código fuente y, particularmente, analizando con detalle los fragmentos del mismo marcados con (P1), dibuje la pila del sistema cuando el programa principal se encuentra en el punto (P1 <-). Justifique su repuesta. (0,5 puntos)



La llamada a “ActualizarReloj” desde C apila sus tres parámetros en orden inverso a su declaración, ocupando cada uno de ellos dos *bytes*. A continuación apila la dirección de retorno de la siguiente instrucción de CM del programa en C (llamada a “ActivarReloj”). Como todas las rutinas de ensamblador son lejanas (*far*), dicha dirección de retorno consta de dos palabras (dos *bytes* por palabra): el valor del registro de segmento CS y el del registro de *offset* IP, apilados en este orden.

Seguidamente se carga el CS y el IP con la dirección de la primera instrucción de “ActualizarReloj” y se inicia la ejecución de ésta, que apila los valores de los registros de 16 bits BP, BX, CX y DX en este orden. El puntero de pila y el registro BP quedan apuntando al valor de DX.

P2. De acuerdo con el código fuente del programa principal y de la rutina *ActualizarReloj*, especialmente las líneas marcadas con (P2), indique la información que se almacena en cada uno de los registros de la CPU: BX, CX, DX. Tenga en cuenta la funcionalidad de la aplicación. Justifique su repuesta. (0.5 puntos)

Según el esquema de la pila indicado en el ejercicio anterior y los valores almacenados en los correspondientes desplazamientos de BP, **BX** se carga con el parámetro “hora”, almacenado en la dirección BP+12; **CX** se carga con el parámetro “minutos”, almacenado en BP+14; **DX** se carga con el parámetro “segundos”, almacenado en BP+16.

P3. De acuerdo con la rutina *ActualizarReloj* (*int, int, int*) del programa principal y de la rutina *interfaz_driver* del *driver*, indique cuál es el mecanismo utilizado para pasar la información actualizada del reloj desde el programa principal al *driver* y dónde es almacenada dicha información. Ayúdese de la información de los puntos marcados como (P3). Justifique su respuesta. (1 punto)

Como se ha visto en el ejercicio anterior, la rutina "ActualizarReloj" carga los registros BX, CX y DX con los parámetros "hora", "minutos" y "segundos" respectivamente, que constituyen la información actualizada del reloj. Por tanto, la rutina "interfaz_driver" recibe dicha información en esos tres registros (**mecanismo de paso de parámetros por registros**) y la almacena en las variables correspondientes del *driver* (ver índice P9).

P4. Complete las instrucciones marcadas con (P4). Ayúdese de los puntos marcados con P3. Justifique su respuesta. (1 punto)

De acuerdo a lo indicado en los anteriores problemas P2 y P3, las instrucciones marcadas con (P4) se completarían del siguiente modo:

```
mov hora, bx  
mov minutos, cx  
mov segundos, dx
```

P5. En las rutinas de servicio incluidas en el *driver*, ¿sería necesario enviar el comando EOI a los PIC que corresponda en cada caso?. Justifique su respuesta. (1 punto)

El *driver* incluye rutinas de servicio de las interrupciones 1Ch y 61h. Ambas interrupciones son generadas desde *software* mediante la instrucción **INT** (la primera es generada por la rutina de servicio del reloj del sistema y la segunda por las rutinas de ensamblador indicadas en el ejercicio). Por tanto, al no ser estas interrupciones generadas directamente por un dispositivo conectado a los controladores de interrupción (PIC) 8259, **no hay que enviar a los PIC ningún comando de final de interrupción EOI**.

P6. ¿Cuál es el mecanismo que utiliza el *driver* para indicar que está residiendo en memoria cuando el programa principal lo comprueba? Ayúdese del análisis de la zona de código alrededor de las líneas marcadas con (P6). Justifique su respuesta. (1 punto)

El programa principal interroga al *driver* mediante una llamada a una función del mismo (**función 00h**). Si el *driver* está instalado y operativo, devuelve en el registro **AX** un valor característico **0F0F0h**. El programa principal comprueba después de la llamada a la función si el valor de AX es éste. Si no lo es, determina que el *driver* no está instalado y operativo.

P7. ¿Qué mecanismo está utilizando el *driver* para calcular la hora en cada momento, es decir, en base a qué se ha conseguido implementar un reloj en tiempo real? Ayúdese del análisis de las líneas de código marcadas con (P7). Justifique su respuesta. (1 punto)

El *driver* hace uso de las interrupciones periódicas del **Timer** del PC (**PIT**) que es el encargado de mantener la hora y la fecha cuando el PC está funcionando. La rutina asociada a las interrupciones del Timer (**vector 08h**) hace una llamada a una **interrupción software** asociada al **vector 1Ch** cuya rutina de servicio no hace nada por defecto. Dicha rutina de servicio del **vector 1Ch**, *que se ejecuta con cada interrupción del Timer, es sustituida por nuestra propia rutina de servicio*, que será la encargada de contar las interrupciones y de actualizar los contadores asociados a la hora, minutos y segundos. No se usa el RTC para generar una interrupción periódica que sea la base de tiempos del reloj en tiempo real que necesitamos. Tenga en cuenta que nuestra rutina de servicio, asociada al vector de interrupción 1Ch, es llamada mediante el mecanismo de interrupción software y que, por tanto, no pasa por los PICs del PC como ocurre con las interrupciones hardware externas.

P8. De acuerdo con el funcionamiento del código de la aplicación, tanto del programa ejecutable (.EXE) como del *driver* (.COM), ¿funcionaría el reloj si salimos del programa principal escrito en C? ¿Qué repercusiones tendría salir del mismo?. Justifique su respuesta. (0.5 puntos)

Existen dos posibilidades, si **salimos bien** del programa principal, éste desinstala el driver y por tanto, **el reloj dejaría de funcionar**. Si **salimos mal**, el driver sigue instalado y por tanto, **el reloj sigue funcionando**. El **problema** es que **no podremos actualizar la hora** ya que esto se hace desde el programa principal.

P9. En el programa principal escrito en C y en el driver existen variables con el mismo nombre. Ayúdese del análisis de las líneas de código marcadas con (P9). De acuerdo con la implementación de la aplicación, razone las siguientes preguntas:

P9.1 ¿Daría algún error la compilación del programa en C y el ensamblado del *driver* debido a la utilización de variables con el mismo nombre? Justifique su respuesta. (0.5 puntos)

No, son **programas independientes** en ficheros independientes. El programa en C es compilado para generar su objeto, y lo mismo se hace con sus rutinas en ensamblador, que serán ensambladas. Los objetos del programa principal en C y de las rutinas en ensamblador serán montados para generar un ejecutable. El otro programa, el *driver*, será ensamblado y montado como un programa independiente, generando así otro ejecutable.

P9.2 ¿Podríamos utilizar el hecho de tener variables con el mismo nombre para pasar la información de la hora, los minutos y los segundos desde el programa principal al *driver* cada vez que queramos actualizar el reloj? Justifique su respuesta. (0.5 puntos)

No, al ser **programas independientes**, es imposible conocer las direcciones asignadas a las variables en un programa desde el otro y viceversa. Son variables diferentes y no existe forma de conocer sus direcciones.

P9.3 Cuando ejecutamos el *driver*, ¿el reloj empezará a funcionar o permanecerá parado mientras no ejecutemos el programa principal? ¿cuál será la hora, los minutos y los segundos que mostrará el reloj en la pantalla del PC si el formato es *hh:mm:ss* suponiendo que el reloj comience a funcionar sin haber ejecutado el programa principal? Justifique su respuesta en cada caso. (0.5 puntos)

Al ejecutar el *driver* y quedarse instalado, **el reloj empieza a funcionar** ya que el código que lo implementa y las interrupciones que lo controlan dependen en exclusiva del *driver* y no del programa principal. **El problema es que las variables asignadas a las horas, minutos y segundos están inicializadas a 0, con lo que se mostraría una hora inicial 00:00:00.** Tenga en cuenta que la *puesta en hora* del reloj se realiza desde el programa principal en C.