Tema 8

Ejercicios de clase

EJERCICIO 1. Considere un periférico TEMP_SENSE, que consiste en un sensor de temperatura ambiente y cuya interfaz tiene tres registros: registro de control, registro de estado y registro de datos, que se muestran en la figura 8.1 y cuya descripción se incluye más adelante. Esta interfaz está conectada a un sistema basado en MIPS R2000, siendo su dirección base $DB = 0x0700FFF0$. La interfaz también dispone de una conexión a línea de interrupción $\overline{Int2}$ del MIPS.

El funcionamiento del sensor es el siguiente: Cuando cambia la temperatura al menos un grado la almacena en el registro de datos, y activa los bits RDY y ERROR. Si el bit IE está activo solicita interrupción al procesador. La interfaz se considera atendida e inactiva el bit RDY cuando se escribe el bit CL del registro de control.

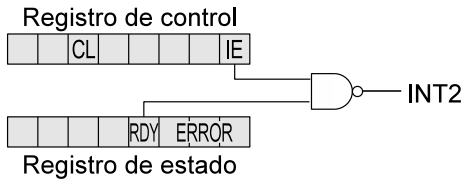

Figura 8.1: Registros de control y estado (ejercicio 1).

REGISTRO DE CONTROL (Dir: DB - 8 bits - escritura):

- **IE** (bit 0): '1' Interrupciones habilitadas, '0' inhabilitadas.
- **CL** (bit 5): '1' Bit Clear: borra el bit RDY. Cuando las interrupciones están activas tiene como efecto que la línea de interrupción se desactiva.

REGISTRO DE ESTADO (Dir: DB - 8 bits - lectura):

- **RDY** (bit 3): '1' Interfaz preparada (hay nueva temperatura), '0' No preparada (no hay nueva temperatura). Además, si está preparado y el bit IE del registro de control está activo, se emite petición de interrupción $\overline{Int_2}$.
- **ERROR** (bit 2, 1 y 0): '000' Cuando no hay error en la medida, valor entre '001' y '111' cuando ha ocurrido un error en la medida, tipificando el error de acuerdo a unas tablas internas.

REGISTRO DE DATOS (Dir: DB+4 - 8 bits - lectura):

- **TEMP** (bits 0..7): Valor de temperatura registrado por el sensor (0 - 255 grados)

Se pretende escribir una rutina de atención (*leer_temp*) a esta interfaz que realice la sincronización mediante la técnica de consulta del estado. La rutina de atención debe leer el contenido

de los bits de ERROR del registro de estado y, si son cero, leer el registro de datos y almacenar su contenido en la variable del sistema TEMP. Si los bits de ERROR son distintos de cero, la rutina debe almacenarlos en la variable del sistema ERROR, y no actualizar TEMP. No olvide que para que el bit RDY cambie a cero y pueda la interfaz mostrar otra medida, se deberá escribir un '1' en el bit CL.

Las variables del sistema asociadas, asi como las primeras instrucciones del programa se muestran a continuación:

```
          .kdata
TEMP:     .byte 0
ERROR:    .byte 0

           .ktext
__start:   mfc0 $t0, $12
            andi $t0, $t0, 0xFBFF
            mtc0 $t0, $12
            li $t0, 0x0700FFF0
            sb $zero, 0($t0)
            .....
            jal leer_temp    # Llama a la subrutina 'leer_temp'
            ....
            ....
            .end
```

1. Indique cuál es el propósito de las primeras líneas del programa.
2. Escriba el código de la subrutina *leer_temp*.

■

EJERCICIO 2. Considere el periférico TEMP_SENSE del ejercicio 1. Se pretende ahora gestionar dicho periférico mediante el mecanismo de sincronización por interrupción. Parte del código de inicialización del sistema de excepciones se muestra a continuación:

```
          .kdata
TEMP:     .byte 0
ERROR:    .byte 0

           .ktext
__start:   mfc0 $t0, $12
            andi $t0, $t0, 0xFBFF
            mtc0 $t0, $12
            li $t0, 0x0700FFF0
            sb $zero, 0($t0)
            ....
            ....
            .end
```

1. Modifique el código de inicialización para habilitar la atención a la interrupción $\overline{Int_2}$, tanto en el procesador como en el periférico.
2. Escriba el código de la rutina de servicio de la interrupción $\overline{Int_2}$.

■

EJERCICIO 3. Considere de nuevo el periférico *TEMP_SENSOR* del ejercicio 1. Suponga ahora que se conecta dos de estos sensores a un sistema MIPS. La dirección base del sensor_0 es `0xFFFFFF00` y la del sensor_1 es `0xFFFFFF10`. Ambos sensores se conectan a la misma línea de interrupción $\overline{Int_2}$. Esta línea se activará cuando alguno o ambos sensores activen la interrupción. La rutina de servicio de dicha interrupción deberá averiguar cuál se ellos ha interrumpido y almacenar la temperatura actual de dicho sensor en las variables del sistema `Temperatura_0` o `Temperatura_1`. Se asume que el sensor_0 es el más prioritario.

```
                .kdata
Temperatura_0:  .byte 0
Temperatura_1:  .byte 0
```

1. Escriba el código de inicio del sistema de excepciones que debe habilitar la interrupción $\overline{Int_2}$, dejando el resto de interrupciones igual que estaban.

2. Escriba el código de servicio de la interrupción $\overline{Int_2}$.

   (Nota: Se pueden usar los registros $t0, $t1 y $t2).

■

EJERCICIO 4. Tienes que diseñar el sistema de acceso al aparcamiento la universidad. Cada acceso incluye tres dispositivos: un punto de inserción de tarjetas magnéticas, una barrera motorizada y un sensor de paso de coches. Las tarjetas magnéticas son de tres categorías: de estudiante, de plantilla y de VIP. Los tres dispositivos van conectados a un computador con MIPS R2000 mediante las interfaces que se describen a continuación (figura 8.2):

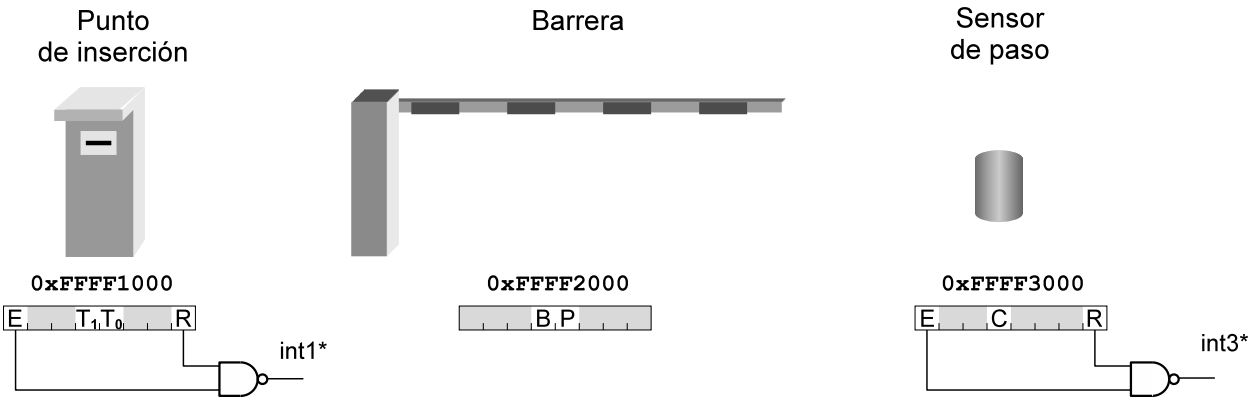

Figura 8.2: Dispositivos para la entrada del aparcamiento (ejercicio 4).

Punto de inserción: Contiene un único registro que combina estado y órdenes en la dirección `0xFFFF1000`. Los bits relevantes que contiene son:

	Campo	Bits	Acceso	Función
R	Preparado	0	Lectura	Se hace R=1 al insertar una tarjeta
T	Tipo	4-3	Lectura	Tipo de tarjeta : 00 (estudiantes), 01 (plantilla), 10 (VIP)
E	Habilitación	7	Escritura	E = 1 habilita las interrupciones

Cuando se inserta una tarjeta, R pasa a valer 1 y el par de bits T indica el tipo de tarjeta. Además, si E = 1 se produce la interrupción $\overline{Int_1}$. R cambia a 0 al leer o escribir el registro.

Barrera: Contiene un registro de órdenes en la dirección `0xFFFF2000`.

	Campo	Bits	Acceso	Función
P	Abrir	3	Escritura	P=1 sube la barrera
B	Cerrar	4	Escritura	B=1 baja la barrera

Para levantar la barrera hay que escribir un 1 en P y para bajarla hay que escribir un 1 en B. Escribir 1 en los dos bits no produce ningún efecto. Igualmente, escribir 0 en los dos bits tampoco produce ningún efecto. Las órdenes inician el movimento de levantar o bajar; la barrera se detiene al final del recorrido y así queda hasta que se da la orden contraria.

Sensor de paso: Contiene un único registro que combina estado y órdenes en la dirección 0xFFFF3000.

	Campo	Bits	Acceso	Función
R	Preparado	0	Lectura	Se hace R=1 a pasar un coche
C	Cancelación	4	Escritura	Escribir C = 1 hace R = 0
E	Habilitación	7	Escritura	E = 1 habilita las interrupciones

El bit R del sensor de paso cambia a 1 cuando pasa el coche. Además, si E = 1 se produce la interrupción $\overline{Int_3}$. R vuelve a 0 cuando se escribe un 1 en el bit C.

El manejador incluye este tratamiento para la interrupción del punto de inserción:

```
int1:   la $t0,0xFFFF1000
        lb $t1,0x0($t0) # cancelo interrupción
        la $t0,0xFFFF2000
        li $t1,0x08
        sb $t1,0($t0)
        j retexc # salta al final del manejador
```

1. Explica lo que hace este fragmento de programa que se ejecuta en modo supervisor.

```
la $t0,0xFFFF1000
sb $zero,0($t0)
la $t0,0xFFFF2000
li $t1,0x10
sb $t1,0($t0)
la $t0,0XFFFF3000
sb $zero,0($t0)
```

2. El vicerrector de Fomento de la universidad quiere que sólo se levante la barrera cuando la tarjeta sea del tipo VIP. Modifica el tratamiento de la interrupción $\overline{Int_1}$ para seguir estas directrices.

3. Escribe el tratamiento de la interrupción $\overline{Int_3}$ para que se baje la barrera al pasar un coche. Además, el tratamiento ha de contar los coches que entran incrementando la variable definida en el segmento de memoria `.ktext`:

```
coches: .word 0
```

4. Escribe el tratamiento de dos funciones de sistema con la especificación siguiente:

Servicio	Código	Parámetros de entrada	Parámetros de salida
`clear_counter`	$v0 = 900	—	—
`read_counter`	$v0 = 901	—	$v0 = número de coches

Estas funciones dan acceso a la variable `coches` definida en el apartado 3. La función *clear_counter* inicia la variable a 0 y la función *read_counter* devuelve su valor.