NOMBRE Y APELLIDOS	GRUPO
Fabián, Scherle Carboneres	2C2

Práctica 13: Sincronización por interrupciones

Paso 1: Habilitación de interrupciones y modo de funcionamiento

► **Cuestión 1.** Código de inicio correcto.

```
li $t0, 0xffff0000 # registro de estado/órdenes del teclado
li $t1, 0x2
sw $t1, 0($t0) # Habilita las interrupciones en el registro de estado/órdenes del teclado
mfc0 $t0, $12
ori $t0, $t0, 0x103
mtc0 $t0, $12
```

► **Cuestión 2**. ¿Por qué se producen los mensajes de error?

No es correcta ya que al ejecutar "jr $k0" no regresa realmente al programa, porque para regresar al programa se debe cargar la dirección de la instrucción en la que se produjo la excepción, y esta dirección se encuentra en el registro $14 del coprocesador.

Paso 2: Obtención de la dirección de retorno al programa de usuario

► **Cuestión 3.** Copia aquí la línea de código que escribe en *$k0* la dirección de retorno

.

```
mfc0 $k0, $14
```

Paso 3: Tratamiento provisional de la interrupción de teclado

► **Cuestión 4.** ¿Por qué se escriben tantos asteriscos al pulsar una tecla?

Esto se debe porque una vez se escribe un asterisco debido a que se ha detectado una interrupción, se vuelve al programa, el procesador comprueba las peticiones pendientes tras la ejecución de cada instrucción y como la petición de interrupción del teclado no se cancela, se trata la interrupción sin parar.

Paso 4: Cancelación de la interrupción

► **Cuestión 5.** Líneas de código que cancelan la interrupción.

```
li $t0, 0xffff0000
lw $a0, 4($t0)
```

► **Cuestión 6.** ¿Por qué acaba el programa de usuario antes de lo esperado?

Esto se debe a que el manejador modifica el contenido del registro $t0 por 0xffff0000, y al ejecutarse la instrucción "bgtz $t0, bucleE" del programa bucles.asm, se interpreta el contenido de este registro como un número negativo, por lo que no se salta a la etiqueta bucleE, por lo que sale del bucle y el programa finaliza ejecutando sus últimas instrucciones.

Paso 5: Gestión del contexto

► **Cuestión 7.** Modificaciones:

- En el segmento de datos del manejador:

```
.kdata
contexto: .word 0,0,0,0 # para salvar $at,$t0,$a0 y $v0
```

- En el código de inicio del sistema:

```
la $k1, contexto
```

- Al principio del código del manejador:

```
## Preserva el contexto del programa de usuario
.set noat
sw $at, 0($k1) # Salvo $at
.set at
sw $t0, 4($k1) # Salvo $t0
sw $a0, 8($k1) # Salvo $a0
sw $v0, 12($k1) # Salvo $v0
```

- Al final del código del manejador:

```
## Restaura el contexto
.set noat
lw $at, 0($k1) # Resturo $at
.set at
lw $t0, 4($k1) # Restauro $t0
lw $a0, 8($k1) # Restauro $a0
lw $v0, 12($k1) # Restauro $v0
```

Paso 6: Habilitación de las interrupciones del reloj

► **Cuestión 8.** Archivo *teclado-y-reloj.handler*.

- En el código de inicio: instrucciones que habilitan la interrupción del reloj.

```
## Habilitar interrupciones para el reloj
li $t0, 0xffff0010 # registro de estado/órdenes del reloj
li $t1, 0x1
sw $t1, 0($t0) # Habilita las interrupciones en el registro de estado/órdenes del reloj
```

- En el código de inicio: desenmascara la línea de interrupción *int2**

```
mfc0 $t0, $12
ori $t0, $t0, 0x503
mtc0 $t0, $12
De esta manera queda desenmascarada int2* e int0* (reloj y teclado)
```

► **Cuestión 9.** Explica por qué la consola mostrará asteriscos sin parar.

Ocurre algo similar a lo que sucedía con el teclado, cuando se produce una interrupción por el reloj, se escribe un asterisco, luego se vuelve al programa, el procesador comprueba las peticiones pendientes tras la ejecución de cada instrucción y como la petición de interrupción del reloj no se cancela, se trata la interrupción sin parar, mostrando asteriscos sin parar.

Paso 7: Análisis de la causa de excepción

► **Cuestión 10**. Archivo *teclado-y-reloj.handler*.

- Código a partir de `retexc`:

```
retexc:                                   El resto de codigo se encuentra en
## Restaura el contexto                   la cuestión 7 apartado 4
.set noat .....
```

- Tratamiento de la interrupción del teclado.

```
int0:   li $v0,11
        li $a0, '*'
        syscall
        li $t0, 0xffff0000
        lw $a0, 4($t0)
        b retexc # Salto para finalizar
```

- Tratamiento del reloj.

```
int2:
  li $t0, 0xffff0010
  li $t1, 0x1
  sw $t1, 0($t0)
  # Cancela la interrupción pero deja habilitadas las interrupciones del reloj
  b retexc
```

- Instrucciones que leen y aíslan el código de causa de excepción.

```
## Identifica y trata excepciones
mfc0 $k0, $13 # Lee registro de Causa
andi $t0, $k0, 0x003c # Aísla los bits 2,3,4,5
bne $t0, $zero, retexc # Compara con 0 y salta si no son iguales
```

EC = 0, el motivo de excepción es una interrupción

- Instrucciones que analizan los bits *IP0* e *IP2* de la palabra de estado

```
## Causa por interrupción
andi $t0, $k0, 0x400 # miro int0
bne $t0, $zero, int0
andi $t0, $k0, 0x1000 # miro int2
bne $t0, $zero, int2
b retexc
```