etsinf

Escola Tècnica
Superior d'Enginyeria
Informàtica

ESTRUCTURA DE COMPUTADORS (GII)

Exercicis del tema 8

Mecanismes de Sincronització de la Entrada/Eixida

Profesors:

Ana PONT
Antonio ROBLES
José M. VALIENTE
José FLICH
Xavi MOLERO
Jorge REAL
Álvaro DOMENECH
Milagros MARTÍNEZ

Curs 2012 - 2013

Tema 8

Mecanismes de Sincronització de la E/E

9.1. Sincronització per consulta d'estat

PROBLEMA 1. La figura 9.1 mostra les connexions de la interfície d'un perifèric que es connecta a una versió reduïda del MIPS R2000 amb només 16 bits d'adreça i 8 de dades. Els registres d'estat i control són de 8 bits. El circuit de selecció consisteix en una única porta tipus NAND de 15 entrades a què es connecten els bits d'adreça des de A_1 fins a A_{15}.

Realitzeu un programa que espere, per consulta de l'estat, que el bit de menor pes del registre d'estat siga igual a 1, per a llavors escriure un 0 en el registre de control.

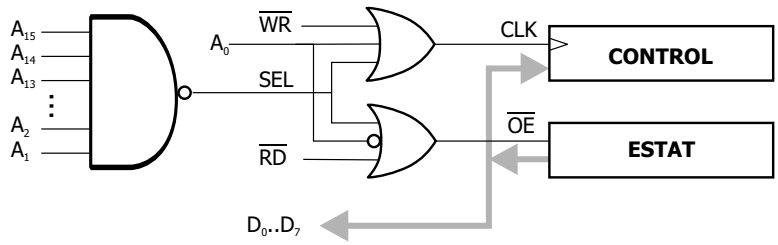

Figura 9.1: Connexions de la interfície del perifèric (problema 1).

SOLUCIÓ: Atenent al circuit de selecció (la porta NAND), l'adreça base d'aquest dispositiu és la que té els bits d'adreça des de A_1 fins a A_{15} a 1 i un zero en A_0. Es tracta doncs de l'adreça $0xFFFE$.

Quan A_0 val 0, llavors s'habilita la lectura del registre de l'estat, aleshores la seua adreça és $0xFFFE$. Per la seua banda, el registre de control es pot escriure només si $A_0 = 1$, amb la qual cosa la seua adreça és la $0xFFFF$, cosa que equival a l'adreça base més 1.

```
            .data
adr_base:   .word 0xFFFE

            .text
            la $t0, adr_base
espera:     lb $t1, 1($t0)
            andi $t1, $t1, 0x01
            beq $zero, $t1, espera
            sb $zero, 0($t0)
```

■

PROBLEMA 2. Un MIPS R2000 té connectats dos perifèrics senzills: un polsador i una perilla (vegeu la Figura 9.2). L'estat del polsador pot ser observat en el bit P que ocupa la posició 0

d'un registre de 8 bits que es pot llegir en l'adreça $0xFFFF0000$. Mentre el polsador està premut $P = 1$ i quan està lliure $P = 0$. La perilla pot ser controlada mitjançant el bit L que ocupa la posició 1 d'altre registre ubicat en l'adreça $0xFFFF1000$. Quan $L = 1$ la perilla s'encén i quan $L = 0$ s'apaga.

Figura 9.2: Els perifèrics del problema 2 i les seues interfícies.

1. Expliqueu què fa aquest programa:

```
        la $t0,0xFFFF0000
        la $t1,0xFFFF1000
        sb $zero,0($t1)
b1:     lb $t2,0($t0)
        andi $t2,$t2,1
        bnez $t2,b1
        li $t2,2
        sb $t2,0($t1)
b2:     lb $t2,0($t0)
        andi $t2,$t2,1
        beqz $t2,b2
        sb $zero,0($t1)
        j b1
```

2. Escriviu un programa que permeta invertir l'estat de la perilla en prémer i alliberar el polsador. En aquest cas, invertir vol dir que si està encesa s'ha d'apagar i si està apagada s'ha d'encendre. Un esquema del programa podria ser aquest:

   apagar perilla
   repetir
   esperar polsador premut
   esperar polsador lliure
   invertir l'estat de la perilla
   per sempre

SOLUCIÓ:

1. El programa encén la perilla mentre el polsador està premut i la deixa apagada mentre el polsador està lliure. Vegem-ho amb més detall:

   Primerament, el programa inicialitza el sistema, preparant les adreces base de tots dos perifèrics i apagant la perilla:

`la $t0,0xFFFF0000`	$t0 = adreça base del botó
`la $t1,0xFFFF1000`	$t1 = adreça base de la perilla
`sb $zero,0($t1)`	apaga la perilla

   Després, es queda en un bucle d'espera del què només eixirà quan es prema el polsador

`b1:`	`lb $t2,0($t0)`	llig el registre amb l'estat del botó
	`andi $t2,$t2,1`	aïlla el bit P

En eixir del bucle d'espera emet l'ordre d'encendre la perilla. Primer prepara l'ordre en el registre $t2 i després l'escriu en la interfície de la perilla.

```
bnez $t2,b1          si P ≠ 0 torna a llegir l'estat
li $t2,2             $t2 = ordre d'encendre perilla
sb $t2,0($t1)        escriu l'ordre en el registre de la perilla
```

Tot seguit, espera que s'allibere el polsador per a apagar la perilla. Noteu que no cal preparar l'ordre d'apagar en el registre $t2 perquè el registre $zero ja la té.

```
b2:   lb $t2,0($t0)      llig el registre amb l'estat del botó
      andi $t2,$t2,1     aïlla el bit P
      beqz $t2,b2        si P = 0 torna a llegir l'estat
      sb $zero,0($t1)    escriu ordre d'apagar en el registre de la perilla
```

I torna al principi:

```
j b1
```

2. Traduint l'esquema de programa a l'assemblador, tenim:

   Inicialització: triem $t2 per a construir l'ordre que cal donar a la perilla. El seu valor inicial serà 0, i l'escrivim en la interfície. Així, la llum comença apagada.

```
la $t0,0xFFFF0000
la $t1,0xFFFF1000
li $t2,0
sb $t2,0($t1)
```

   Esperem a que es prema el polsador:

```
b1:   lb $t3,0($t0)
      andi $t3,$t3,1
      bnez $t3,b1
```

   i ara esperem que s'allibere:

```
b2:   lb $t3,0($t0)
      andi $t3,$t3,1
      beqz $t3,b2
```

   Finalment, el programa construeix l'ordre. Si $t2 val 0, és a dir, si la llum estava apagada, ha d'encendre-la fent $t2 = 2. Igualment, si $t2 val 2, és a dir, si la llum estava encesa, ha d'apagar-la fent $t2 = 0. Després d'escriure l'ordre, el programa torna a començar.

```
xori $t2,$t2,2
sb $t2,0($t1)
j b1
```

   Altra solució suposaria invertir l'estat de la perilla en el moment que es prema el botó. L'esquema de programa seria aquest:

   apagar perilla
   repetir
   esperar polsador premut
   invertir l'estat de la perilla
   esperar polsador lliure
   per sempre

En assemblador:

```
        la $t0,0xFFFF0000
        la $t1,0xFFFF1000
        li $t2,0
        sb $t2,0($t1)
b1:     lb $t3,0($t0)
        andi $t3,$t3,1
        bnez $t3,b1
        xori $t2,$t2,2
        sb $t2,0($t1)
b2:     lb $t3,0($t0)
        andi $t3,$t3,1
        beqz $t3,b2
        j b1
```

■

PROBLEMA 3. La interfície d'un sensor de temperatura ambient, anomenat TEMP_SENSE, té tres registres: registre de control, registre de estat i registre de dades, que es mostren en la figura 9.3 i la seua descripció es trova mes endavant. Esta interfície es troba connectada a un sistema basat en MIPS R2000 per la línia d'atenció a interrupció $\overline{Int_2}$. Els registres d'estat i de control són accessibles a través de l'adreça base $0x0700FFF0$, mitjançant operacions de lectura i d'escriptura, respectivament, i el registre de dades només es pot llegir en l'adreça base+4.

El funcionament del sensor és el següent:

Quan canvia la temperatura almenys un grau, s'emmagatzema en el registre de dades, i activa els bits RDY i ERROR. Si el bit IE està actiu sol·licita interrupció al processador. La interfície es considera atesa i inactiva el bit RDY quan s'escriu el bit CL del registre de control.

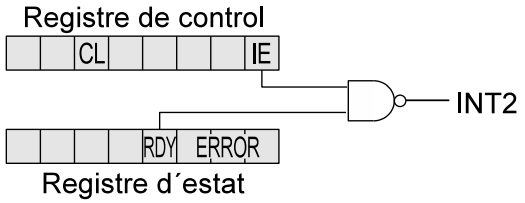

Figura 9.3: Registres de control i d'estat del termòmetre (problema 3).

REGISTRE DE CONTROL (Dir: DB - 8 bits - escriptura):

IE (bit 0): '1' Interrupcions habilitades, '0' inhabilitades.

CL (bit 5): '1' *Bit Clear*. El bit RDY del registre d'estat canvia a zero. Quan les interrupcions estan actives té com a efecte que la línia d'interrupció s'inactiva. El bit RDY només passa a zero quan s'escriu un u en aquest bit.

REGISTRE D'ESTAT (Dir: DB - 8 bits - lectura):

RDY (bit 3): '1' Interfície preparada (hi ha nova temperatura), '0' No preparada (no hi ha nova temperatura). A més a més, si està preparada i el bit IE del registro de control està actiu, s'emet petició d'interrupció.

ERROR (bits 2, 1 i 0): '000' Quan no hi ha error en la mesura, valor entre '001' i '111' quan ha hagut un error en la mesura, tipificant l'error d'acord amb unes taules internes.

REGISTRE DE DADES (Dir: DB+4 - 8 bits - lectura):

TEMP (bits 0..7): Valor de temperatura registrat per el sensor (0 - 255 graus)

Es desitja ecriure el codi de la rutina d'atenció a aquesta interfície (*leer_temp*) mitjançant la tècnica de consulta de l'estat per a la sincronització, tot assumint que les interrupcions en la interfície estan deshabilitades (bit IE = 0). La rutina d'atenció ha de llegir el contingut del registre d'estat, bits de ERROR. Si són zero, cal llegir el registre de dades i emmagatzemar el seu contingut en la variable del sistema TEMP. Si els bits d'ERROR són distints de zero, cal emmagatzemar-los en la variable del sistema ERROR, i no actualitzar TEMP. No oblideu que perquè el bit RDY canvie a zero i la interfície puga mostrar altra mesura caldrà escriure un '1' en el bit CL.

Les variables del sistema associades, aixi com les primeres instruccions del programa es mostren a continuació:

```
          .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. Indiqueu quin es el propósit de les primeres línies del programa.
2. Escriviu el codi de la rutina *leer_temp*.

SOLUCIÓ:

1. Les primeres instruccions s'han executat per a emmascarar la línia $\overline{Int_2}$ en el MIPS R2000, aixi com per deshabilitar les interrupcions en la interfície del sensor (bit IE $=$ 0).
2. Codi de rutina d'atenció per consulta de l'estat:

```
              .kdata
    COMPTADOR:  .byte 0
    ERROR:      .byte 0

              .ktext
        rutina: li $t0, 0x0700FFF0     # Adreça base
         bucle: lb $t1, 0($t0)         # Accés a ESTAT
                andi $t2, $t1, 0x08    # Filtre bit RDY (bit 3)
                beq $t2, $zero, bucle

                andi $t1, $t1, 0x07    # Filtre bits ERROR (bits 2,1,0)
                bne $t1, $zero, error

     comptador: lb $t1, 4($t0)         # Accés a DADES
```

```
            sb $t1, TEMP          # Emmagatzema en TEMP
            j fi

      error: sb $t1, ERROR        # Emmagatzema ERROR

         fi: li $t1, 0x20         # bit CL = 1, IE = 0
            sb $t1, 0($t0)
            jr $31                # Retorn de subrutina
```

■

9.2. Sincronizatció per interrupcions

PROBLEMA 4. Considere el periférico TEMP_SENSE del ejercicio 3. Se pretende ahora gestionar dicho periférico mediante el mecanismo de sincronización por interrupción. Parte del código de inicialización del manejador 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 del manejador 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 $\overline{Int_2}$.

SOLUCIÓ:

1. Las primeras instrucciones del manejador deben desenmascarar la interrupción $\overline{Int_2}$ en el MIPS R2000, así como para poner el bit IE=1 (Interrupt Enable) en la interfaz del sensor.

```
            .ktext
__start:   mfc0 $t0, $12
            ori $t0, $t0, 0x0400 #  Tan solo posiciono el bit 10,
            mtc0 $t0, $12         #  que es el bit asociado a INT2
            li $t0, 0x0700FFF0
            li $t1, 0x01
            sb $t1, 0($t0)
            ....
            ....
            .end
```

2. Código de rutina de servicio de la interrupción $\overline{Int_2}$:

```
int2:       li $t0, 0x0700FFF0      # Puntero a BASE
            lb $t1, 0($t0)          # Acceso a ESTADO
            andi $t1, $t1, 0x07     # Filtro bits ERROR (bits 2,1,0)
            bne $t1, $zero, error
            lb $t1, 4($t0)          # Acceso a DATOS
            sb $t1, TEMP            # Almaceno en TEMP
            j fin
error:      sb $t1, ERROR           # Almaceno ERROR
fin:        li $t1, 0x21            # bit CL = 1, IE = 1
            sb $t1, 0($t0)          # cancela la interrupción
            b retexec               # Retorno de excepción
```

■

PROBLEMA 5. Considere de nuevo el periférico *TEMP_SENSOR* del ejercicio 3. 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`. Ámbos 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 manejador 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).

SOLUCIÓ:

1. Código inicialización en el manejador de excepciones:

```
            .text
            .globl __start

__start:
                                    ## Código de inicialización
             ......
             la $t0,0xFFFFFF00      # $t0 = dirección base sensor_0
             lb $t1,0($t0)
             ori $t1, $t1, 0x01
             sb $t1, 0($t0)         # IE = 1
             la $t0,0xFFFFFF10      # $t0 = dirección base sensor_1
             lb $t1,0($t0)
             ori $t1, $t1, 0x01
             sb $t1, 0($t0)         # IE = 1
             mfc0 $t0, $12
             ori $t0, $t0, 0x2003   # desenmarcarar int_5, modo USER
             mtc0 $t0, $12          # e Interrupt Enable = 1
             .....
```

2. El código de la rutina de servicio de la interrupción 2 comprueba cuál de los dos sensores ha interrumpido, comenzando por el sensor 0 que es el más prioritario.

```
int_2:      la $t0,0xFFFFFF00       # $t0 = dirección base sensor_0
            lb $t1, 0($t0)          # registro de estado
            andi $t1, $t1, 0x08     # bit R ready
            beq $t1, $0, leer_s_1
            lb $t1, 4($t0)
            sb $t1, temperatura_0   # Guarda temp del sensor_0
            li $t1, 0x81
            sb $t1, 0($t0)          # Clear int_5
            j retexc
leer_s_1:   la $t0,0xFFFFFF10       # $t0 = dirección base sensor_1
            lb $t1, 4($t0)
            sb $t1, temperatura_1   # Guarda temp del sensor_1
            li $t1, 0x81
            sb $t1, 0($t0)          # Clear int_5
            j retexc
```

■

9.3. Interrupcions i funcions de sistema

PROBLEMA 6. Un computador amb processador MIPS R2000 té connectats dos perifèrics *A* i *B* amb les interfícies següents:

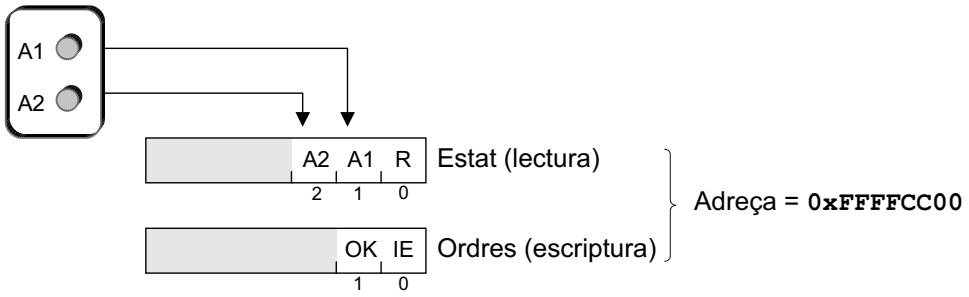

Figura 9.4: Interfície del perifèric *A* (problema 6).

El perifèric *A* només conté dos botons *A1* i *A2* independents. La seua interfície consta dels dos registres de 8 bits, *Estat* i *Ordres*, que es mostren a la figura 9.5. El registre d'estat només admet lectura i el d'ordres només escriptura. Tots dos registres tenen la mateixa adreça `0xFFFFCC00`. En tot moment, els bits *A1* i *A2* del registre d'estat indiquen a 1 si el botó corresponent es troba pitxat i a 0 el cas contrari. El bit *R* passa a valdre 1 quan algun dels bits *A1* o *A2* canvia de 0 a 1 i torna a valdre 0 quan s'escriu 1 en el bit *OK* del registre d'ordres. Finalment, si s'escriu 1 en el bit *IE*, l'adaptador del perifèric activa la línia d'interrupció $\overline{Int_3}$ quan $R = 1$.

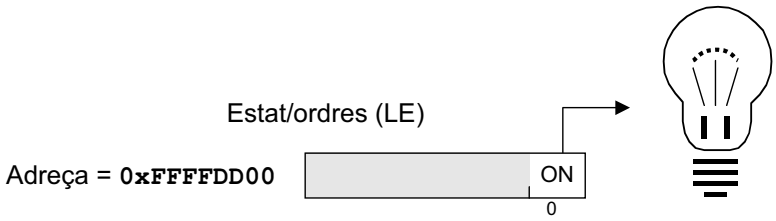

Figura 9.5: Interfície del perifèric *B* (problema 6).

El perifèric *B* és un llum controlat mitjançant la interfície de la figura 9.5. L'únic bit practicable de la interfície (*ON*) pot ser llegit i escrit mitjançant el registre d'estat i ordres ubicat en `0xFFFFDD00`. Si s'escriu $ON = 1$, el llum s'encén, en cas contrari s'apaga. La lectura d'aquest registre proporciona l'últim valor que s'hi ha escrit.

1. Observeu el programa següent i descriviu el comportament del sistema:

```
        la $t0,0xFFFFCC00
        la $t2,0xFFFFDD00

bucle1: lb $t1,0($t0)
        andi $t1,$t1,4
        beq $t1,$zero,bucle1

        li $t1,1
        sb $t1,0($t2)

buble2: lb $t1,0($t0)
        andi $t1,$t1,4
```

```
        bne $t1,$zero,bucle2

        sb $zero,0($t2)

        j bucle1
```

2. Escriviu un programa que, per consulta d'estat del bit *R*, controle els perifèrics perquè el sistema tinga el comportament següent:

   En prèmer el botó *A1*, el llum s'ha d'encendre (estiga com estiga prèviament) i ha de mantenir-se encés fins que, en prèmer el botó *A2*, el llum s'apague. Prèmer tots dos botons al temps no ha de tenir cap efecte.

3. Considereu ara que les interrupcions estan habilitades en la interfície d'*A*. Escriviu el tractament que, dins del manejador d'excepcions, ha d'aplicar-se a la interrupció $\overline{Int_3}$. El comportament ha de ser el descrit a l'apartat 2. Podeu utilitzar els registres *$t0* a *$t3*.

4. S'ha inclòs en el manejador les funcions 3297 i 3298 que poden fer servir els programes d'usuari mitjançant la crida al sistema. La funció 3297 permet als programes encendre i apagar el llum *B*. La funció 3298 permet als programes consultar la situació del llum. La seua especificació és la següent:

índex ($v0)	argument(s)	resultat
3297	$a0 = 1 (encés) o 0 (apagat)	—
3298	—	$v0 = 1 (encés) o 0 (apagat)

   Escriviu un programa d'usuari que cride aquestes funcions per tal d'invertir l'estat del llum: si està apagat cal encendre'l; si està encés cal apagar-lo.

SOLUCIÓ:

1. Comportament del sistema: El computador, per consulta d'estat, fa que s'encenga el llum *B* mentres es prem el botó corresponent al bit *A2*.

2. Exercici de consulta d'estat:

   Una versió amb dos bucles de sincronització.

```
        la $t0,0xFFFFCC00
        la $t3,0xFFFFDD00

        # el llum està apagat
bucle1: lb $t1,0($t0)   # espera R = 1
        andi $t2,$t1,1
        beq $t2,$zero,bucle1

        li $t2,1        # fa OK = 1
        sb $t2,0($t0)

        andi $t2,$t1,4  # si A2 = 1 continua esperant
        bne $t2,$zero,bucle1

encen:  li $t2,1        # fa ON = 1
        sb $t2,0($t3)
```

```
        # el llum està encés
bucle2: lb $t1,0($t0)   # espera R = 1
        andi $t2,$t1,1
        beq $t2,$zero,bucle2

        li $t2,1        # fa OK = 1
        sb $t2,0($t0)

        andi $t2,$t1,2  # si A1 = 1 continua esperant
        beq $t2,$zero,bucle2

apaga:  sb $zero,0($t3) # fa ON = 0
j bucle
```

Altra versió, amb només un bucle de sincronització:

```
        la $t0,0xFFFFCC00
        la $t3,0xFFFFDD00

bucle:  lb $t1,0($t0)   # espera R = 1
        andi $t2,$t1,1
        beq $t2,$zero,bucle

        li $t2,1        # fa OK = 1
        sb $t2,0($t0)

        li $t2,4        # analitza A2 i A1
        beq $t2,$t1,apaga
        li $t2,2
        beq $t2,$1,encen
        j bucle

encen:  li $t2,1        # fa ON = 1
        sb $t2,0($t3)
j bucle

apaga:  sb $zero,0($t3) # fa ON = 0
j bucle
```

3. El tractament d'interrupcions es pot fer fàcilment a partir de la versió de consulta d'estat que se sincronitza amb un bucle d'espera. Només hi caldrà llevar el bucle i sustituir els salts dirigits a l'etiqueta `bucle` per salts al l'etiqueta `retexc` de final del manejador

```
int3:   la $t0,0xFFFFCC00
        la $t2,0xFFFFDD00

        li $t2,1        # fa OK = 1
        sb $t2,0($t0)

        li $t2,4        # analitza A2 i A1
        beq $t2,$t1,apaga
        li $t2,2
        beq $t2,$1,encen
        j retexc

encen:  li $t2,1        # fa ON = 1
```

```
        sb $t2,0($t3)
j retexc

apaga:  sb $zero,0($t3) # fa ON = 0
j retexc
```

4. Crides al sistema per a invertir l'estat del llum:

```
# cride la funció que consulta l'estat
  li $v0,3298
  syscall
# ara tinc en $v0 l'estat del llum
# calcule el paràmetre per a invertir l'estat
  xori $a0,$v0,1
# cride la funció que fixa nou estat
  li $v0,3297
  syscall
```

■

PROBLEMA 7. Un sistema empotrat controla la temperatura d'una planta comercial. Aquest sistema està compost per un computador basat en un processador MIPS R2000 connectat a una interfície que consta d'un termòmetre, un calefactor (bomba de calor) i un refrigerador (bomba de fred). El funcionament que volem aconseguir és mantenir la temperatura entorn a un valor que es programarà inicialment. Els registres que té la interfície són quatre (tots de 8 bits):

CONTROL_ESTAT (escriptura/lectura, adreça base `0xF9000010`)

R: (bit 0) s'activa a 1 quan la temperatura canvia un grau.

CL: (bit 6) a 1 té l'efecte de posar R a 0 (tot cancel·lant la interrupció si n'hi ha).

IE: (bit 7) a 1 habilita la interrupció. Si $IE = 1$, la interrupció s'emetrà quan $R = 1$.

CONTROL_MOTOR (només escriptura, adreça base + 4)

CA: (bit 0) a 1 activa el calefactor (bomba de calor), a 0 l'apaga.

FI: (bit 7) a 1 activa el refrigerador (bomba de fred), a 0 l'apaga.

TEMPERATURA (lectura/escriptura, adreça base + 8) Conté el valor de la temperatura en graus centígrads, codificada com a enter amb signe (complement a dos).

1. Escriviu el codi d'inici del manejador d'excepcions en el MIPS R2000 que ha d'habilitar les interrupcions generals, posar el processador en mode usuari, habilitar la línia d'interrupció $\overline{Int_0}$ deixant la resta d'interrupcions igual que estaven, i accedir a la interfície per tal d'habilitar les interrupcions. El contingut del registre d'estat està descrit en la figura A.1.

2. Escriviu el codi de la rutina de tractament de la línia d'interrupció $\overline{Int_0}$. El seu objectiu és llegir el registre de temperatura (valor amb signe) i emmagatzemar-lo en la variable del sistema `temperatura` que s'ha definit segons es mostra a continuació. No oblideu cancel·lar la petició d'interrupció. Podeu emprar els registres temporals `$t0` i `$t1`.

```
              .kdata
temperatura:  .byte 0
```

3. Escriviu el tractament de les funcions del sistema que tenen el perfil següent:

Funció	Índex ($v0)	Paràmetres d'eixida
`llegir_temp`	33	$a0 = temperatura
`activar_calor`	34	
`activar_fred`	35	
`parar`	36	

Les funcions anteriors tenen el comportament següent:

- `llegir_temp` carrega en el registre $a0 el contingut de la variable del sistema `temperatura`.
- `activar_calor` activa el calefactor de la interfície (bomba de calor) i apaga el de fred.
- `activar_fred` activa el refrigerador de la interfície (bomba de fred) i apaga el de calor.
- `parar` apaga el calefactor i el refrigerador; si ja estaven apagats no té cap efecte sobre la interfície.

4. Cofidiqueu una rutina que, emprant les crides al sistema anteriors, reproduïsca les acciones indicades pel pseudocodi següent:

si $(temp > temp\_max)$
    activar fred
si no
    si $(temp < temp\_min)$
        activar calor
    si no
        parar fred i calor
    fsi
fsi

La variable `temp` indica la temperatura mesurada per la interfície i s'haurà d'accedir al seu valor mitjançant la crida al sistema adient. Les temperatures de referència `temp_màx` i `temp_mín` són variables definides de la manera següent:

```
            .data
temp_màx:   .byte 25
temp_mín:   .byte 17
```

SOLUCIÓ:

1. Inicialización general de interrupciones.

```
inicializa:     la $t0, 0xF9000010  # Base de la interfaz
                li $t1, 0x80        # bit IE (bit 7) a "1"
                sb $t1, 0($t0)      # activa interrupciones en la interfaz

                mfc0 $t0, $13       # lectura del registro estado coprocesador 0
                ori $t0, $t0,0x103  # activar int0 + int generales + modo usuario
                mtc0 $t0, $13       # escritura del registro estado
```

2. Rutina de tratamiento de la interrupción $\overline{Int_0}$.

```
int0:           la $t0, 0xF9000010  # Base de la interfaz
                li $t1, 0xC0        # bit IE (bit 7) a "1", y CL (bit 6) a "1"
                sb $t1, 0($t0)      # cancelar interrupción, manteniendo activación

                lb $t1, 8($t0)      # leer temperatura
                sb $t1, temperatura # actualizar variable del sistema
                j retexc
```

3. Llamadas al sistema,

```
leer_temp:      la $t0, temperatura # variable del sistema
                sb $a0, 0($t0)      # $a0 = temperatura
                j retexc

activar_calor:  la $t0, 0xF9000010  # Base de la interfaz
                li $t1, 1           #  bit CA = "1"
                sb $t1, 8($t0)      # activa calor, parar frio
                j retexc

activar_frio:   la $t0, 0xF9000010  # Base de la interfaz
                li $t1, 0x80        #  bit FI = "1"
                sb $t1, 8($t0)      # activa frio, parar calor
                j retexc

parar:          la $t0, 0xF9000010  # Base de la interfaz
                sb $0, 8($t0)       #  bit FI = "0" y CA="0"
                j retexc
```

4. Rutina que implementa la comprobación de la temperatura entre márgenes.

```
                .data
temp_max:   .byte 25   # temperatura referencia maxima
temp_min:   .byte 17   # temperatura referencia minima
                .text
rutina   :      li $v0, 33
                syscall               # $a0 = temperatura

                lb $t0, temp_max      # Margen superior
                bgt $a0, $t0, frio    # Si temperatura > margen: frio

                lb $t0, temp_min      # Margen inferior
                blt $a0, $t0, calor   # Si temperatura < margen: calor

                li $v0, 36            # Dentro margenes: parar
                j fin

frio:           li $v0, 35            # activa el calor, para el frio
                j fin

calor:          li $v0, 34            # activa el frio, para el calor
                j fin

fin:            syscall               # realiza la función apropiada
                jr $ra
```

■

PROBLEMA 8. Heu de dissenyar el sistema d'accés a l'aparcament de certa universitat americana. Cada accés inclou tres dispositius: un punt d'inserció de targetes magnètiques, una barrera motoritzada i un sensor de pas de cotxes. Les targetes magnètiques són de tres categories: d'estudiant, de plantilla i de VIP. Els tres dispositius van connectats a un computador mitjançant les interfícies que es descriuen tot seguit (veieu la figura 9.6):

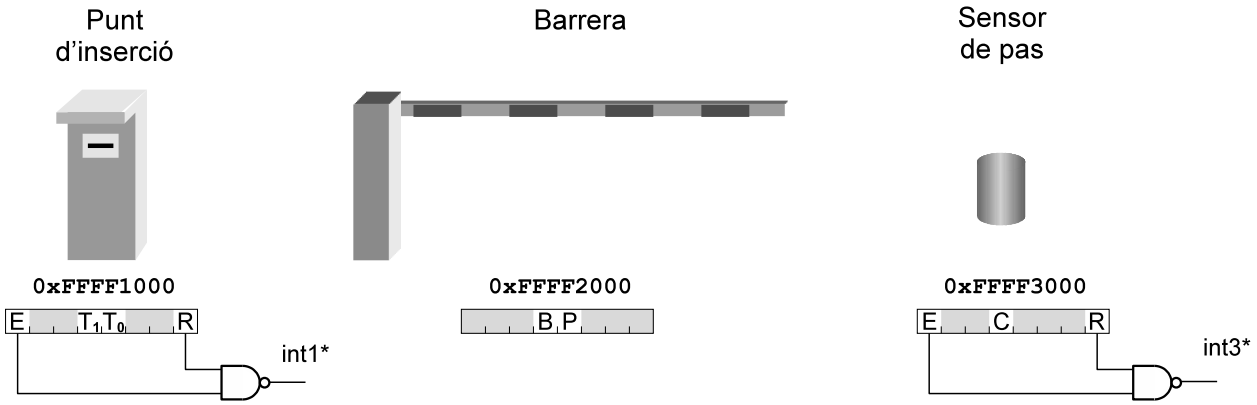

Figura 9.6: Dispositius per a l'entrada de l'aparcament (problema 8).

Punt d'inserció: Conté un únic registre que combina estat i ordres en l'adreça `0xFFFF1000`. Els bits rellevants que conté són:

	Nom	Pos.	Accés	Funció
R	Preparat	0	Lectura	Es fa R=1 en inserir una targeta
T	Tipus	4-3	Lectura	Tipus de targeta : 00 (estudiants) 01 (plantilla), 10 (VIP)
E	Habilitació	7	Escriptura	E = 1 habilita les interrupcions

Quan s'insereix una targeta, R passa a valdre 1 i el parell de bits T indica el tipus de targeta. A més a més, si E = 1 es produeix la interrupció $\overline{Int_1}$. R torna a 0 en llegir o escriure el registre.

Barrera: Conté un registre d'ordres en l'adreça `0xFFFF2000`.

	Nom	Pos.	Accés	Funció
P	Obrir	3	Escriptura	P=1 puja la barrera
B	Tancar	4	Escriptura	B=1 abaixa la barrera

Per alçar la barrera cal escriure un 1 en P i per baixar-la cal escriure un 1 en B. Escriure 1 en ambdós bits no produeix cap efecte. Igualment, escriure 0 en ambdós bits tampoc no produeix cap efecte. Les ordres inicien el moviment d'alçar o abaixar; la barrera es deté al final del recorregut i així queda fins que es dóna l'ordre contrària.

Sensor de pas: Conté un únic registre que combina estat i ordres en l'adreça `0xFFFF3000`.

	Nom	Pos.	Accés	Funció
R	Preparat	0	Lectura	Es fa R=1 en passar un cotxe
C	Cancel·lació	4	Escriptura	Escriure C = 1 fa R = 0
E	Habilitació	7	Escriptura	E = 1 habilita les interrupcions

El bit R del sensor de pas canvia a 1 quan passa el cotxe. A més a més, si E = 1 es produeix la interrupció $\overline{Int_3}$. R torna a 0 quan s'escriu un 1 en el bit C.

El manejador inclou aquest tractament per a la interrupció del punt d'inserció:

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

1. Expliqueu què fa aquest fragment de programa que s'executa en mode 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 vicerector de Foment d'aquella universitat vol que només s'alce la barrera quan la targeta siga del tipus VIP. Modifiqueu el tractament de la interrupció $\overline{Int_1}$ per a seguir aquestes directrius.

3. Escriviu el tractament de la interrupció $\overline{Int_3}$ perquè s'abaixe la barrera en passar un cotxe. A més a més, el tractament ha de comptar els cotxes que entren incrementant la variable definida en el segment de memòria `.ktext`:

```
cotxes: .word 0
```

4. Escriviu el tractament de dues funcions de sistema amb l'especificació següent:

Servei	Codi	Paràmetres d'entrada	Paràmetres d'eixida
`clear_counter`	$v0 = 900	—	—
`read_counter`	$v0 = 901	—	$v0 = nombre de cotxes

Aquestes funcions donen accés a la variable `cotxes` definida en l'apartat 3. La funció *clear_counter* inicia la variable a 0 i la funció *read_counter* retorna el seu valor.

SOLUCIÓ:

1. El programa cancel·la les interrupcions i baixa la barrera

2. Tractament de $\overline{Int_1}$ perquè només entren VIP:

```
int1:   la $t0,0xFFFF1000
        lb $t1,0($t0)
        andi $t1,$t1,0x18   # aïlla bits tipus targeta
        li $t0,0x10         # t0 = tipus VIP
        bne $t0,$t1,retexc  # si tipus targeta != VIP, salta a fi
        la $t0,0xFFFF2000
        li $t1,0x08
        sb $t1,0($t0)
        j retexc # salta al final del manejador
```

3. Tractament de $\overline{Int_3}$:

```
int3:   la $t0,0xFFFF3000
        li $t1,0x90
        sb $t1,0x0($t0)
        la $t0,0xFFFF2000
        li $t1,0x10
        sb $t1,0($t0)
        lw $t1,cotxes
        addi $t1,$t1,1
        sw $t1,cotxes
        j retexc         # salta al final del manejador
```

4. Funcions 900 i 901:

```
fun900: sw $zero,cotxes
        j retexc         # salta al final del manejador

fun901: lw $v0,cotxes
        j retexc         # salta al final del manejador
```

■

PROBLEMA 9. Considereu el perifèric *KeyPad*, que consisteix en un teclat numèric amb tecles del 0 al 9 i un visualitzador de 7 segments, com es mostra a la figura 9.7.

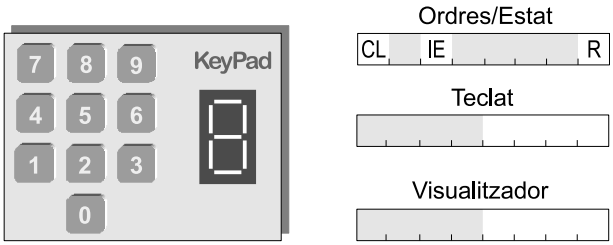

Figura 9.7: Aspecte de *KeyPad* (problema 9) i de la seua interfície.

Aquest dispositiu té la interfície següent:

REGISTRE D'ORDRES I ESTAT (8 bits, lectura/escriptura, Adreça base + 0)

R: (Ready), bit de només escriptura. A 1 indica que s'ha premut una tecla i que el registre de dades del teclat conté un valor nou.

IE: (Interrupt Enable), bit de lectura/escriptura. Fer $IE = 1$ habilita les interrupcions. Mentre $IE = 1$ i $R = 1$, el perifèric activa la línia de petició d'interrupció.

CL: (Clear), bit de només escriptura. Fer $CL = 1$ força el bit $R = 0$.

REGISTRE DE DADES DEL *TECLAT* (8 bits, lectura, Adreça Base + 4)

bits 3...0: Codi de tecla (del 0 al 9).

REGISTRE DE DADES DEL *VISUALITZADOR* (8 bits, escriptura, Adreça Base + 4)

bits 3...0: Si s'hi escriu un valor del 0 al 9, el visualitzador mostra el dígit corresponent. Un valor -1 (*0xF*) encén només el segment central. La resta de valors no estan definits.

Si el bit *IE* està a 1, el dispositiu activa una línia de petició d'interrupció quan algú prem una tecla de *KeyPad* i la desactiva si el processador escriu un 1 en el bit *CL* del registre d'ordres. El visualitzador de segments està sempre preparat i, per tant, no té cap relació amb les interrupcions. El controlador es connecta a un computador basat en el MIPS R2000. L'adreça base escollida és `0xFFFFFF00`. La petició d'interrupció està connectada a la línia $\overline{Int_2}$ del processador.

1. Completeu el disseny del circuit de selecció de la interfície del *KeyPad* que es mostra en la figura 9.8. La part que falta (selecció de registre) té quatre eixides: $OEst_{out}$, $OEst_{clk}$, Tec_{out} i Vis_{clk}. El senyal d'entrada *BE** que hi apareix és la línia d'habilitació (a nivell baix) del byte menys significatiu de les paraules.

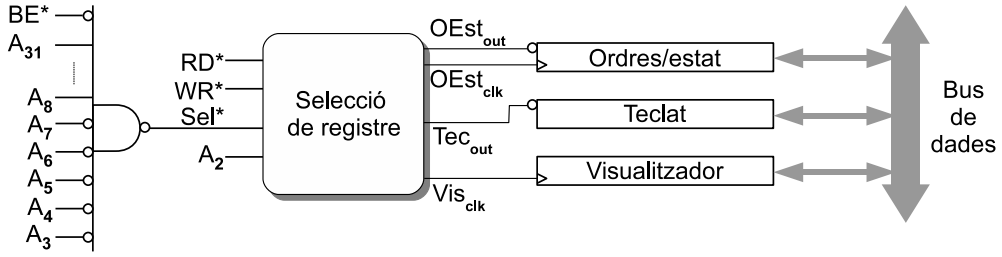

Figura 9.8: Circuit de selecció de *KeyPad* (problema 9).

2. Considereu que, inicialment, el teclat té les interrupcions inhibides. Què fa el programa següent quan s'executa en mode supervisor? Afegiu els comentaris adients a les instruccions i les explicacions necessàries.

```
        la t0,0xFFFFFF00
bucle:  lb $t1,0($t0)
        andi $t1,$t1,1
        beqz $t1,bucle
        li $t1,0x80
        sb $t1,0($t0)
        lb $t1,4($t0)
        andi $t1,t1,0xF
        li $t2,5
        bne $t1,$t2,bucle
        li $t1,0xA0
        sb $t1,0($t0)
```

3. Escriviu el codi de tractament del *KeyPad* que responga a una petició d'interrupció hardware. Aquest tractament ha d'emmagatzemar el codi de la tecla premuda en la variable del manejador *KP_Key* (de tipus *byte*), però no ha d'operar sobre el visualitzador. *Expliqueu el vostre codi amb els comentaris adients.*

4. Escriviu el codi d'inicialització del *KeyPad* perquè el tractament de l'apartat 3 siga correcte. Aquest codi ha d'inicialitzar la variable *KP_Key* a -1, encendre el segment central del visualitzador, habilitar la interrupció en la interfície del perifèric i preparar la màscara adient en el registre d'estat del MIPS R2000. Expliqueu el vostre codi amb els comentaris adients.

5. Escriviu el codi d'un servei del sistema amb el perfil següent:

Servei	Codi	Paràmetres d'entrada	Paràmetres d'eixida
`read_KP`	$v0 = 11	—	$v0 = últim número teclejat

Aquest servei ha de tenir dos efectes: ha de lliurar al programa el valor emmagatzemat en la variable *KP_Key* pel tractament de l'apartat 3 i ha de mostrar-lo en el visualitzador de *KeyPad*. Expliqueu el vostre codi amb els comentaris adients.

6. Tot seguit, hi ha el tractament de la funció 513 i un tractament de la interrupció $\overline{Int_2}$ distint del descrit en l'apartat 3:

```
fun513: sb $a0,KP_Key
        jal suspen_aquest_proces
        j retexc

int2:   la $t0,0xFFFFFF00
        li $t1,0xA0
        sb $t1,0($t0)
        lb $t1,4($t0)
        lb $t2,KP_Key
        bne $t1,$t2,retexc
        jal activa_proc_en_espera
        j retexc
```

Expliqueu breument què fa el servei 513 i escriviu-ne l'especificació amb l'estil amb què s'ha definit el servei 11 en l'apartat 5.

SOLUCIÓ:

1. Disseny del circuit de selecció dels registres: En la figura 9.9 (esquerra) apareix la solució convencional amb un descodificador d'una entrada. També es pot dissenyar el circuit directament amb portes. En funció de l'adreça de cada registre i del tipus d'accés (lectura o escriptura) amb què han d'operar, podem determinar quins valors de les quatre variables d'entrada activen cadascun dels quatre senyals d'operació:

Sel*	A_2	RD*	WR*	Eixida activa
0	0	0	X	$OEst_{out}$
0	0	X	0	$OEst_{clk}$
0	1	0	X	Tec_{out}
0	1	X	0	Vis_{clk}

Tenint en compte que els senyals d'eixida són actius per nivell baix, s'hi poden obtenir les expressions algebraiques de cadascuna (per exemple, $OEst_{out} = Sel^* + A_2 + RD^*$ i $Vis_{clk} = Sel^* + \overline{A_2} + WR^*$). Vegeu-ne el circuit resultant en la figura 9.9 (dreta).

2. El codi espera per consulta d'estat fins que algú prema la tecla "5". Aleshores, habilita les interrupcions en la interfície de *KeyPad*. Vegem-ho amb detall:

```
        la $t0,0xFFFFFF00
bucle:  lb $t1,0($t0)     # Bucle de consulta d'estat:
        andi $t1,$t1,1    #   espere fins teclat preparat
        beqz $t1,bucle
        li $t1,0x80       # Reinicie el teclat
        sb $t1,0($t0)
        lb $t1,4($t0)     # Llig codi de tecla
        andi $t1,t1,0xF
        li $t2,5          # Compare codi amb "5"
        bne $t1,$t2,bucle # Si codi != "5", torne a començar
        li $t1,0xA0       # Si codi == "5", reinicie teclat i
        sb $t1,0($t0)     #     habilite les interrupcions
```

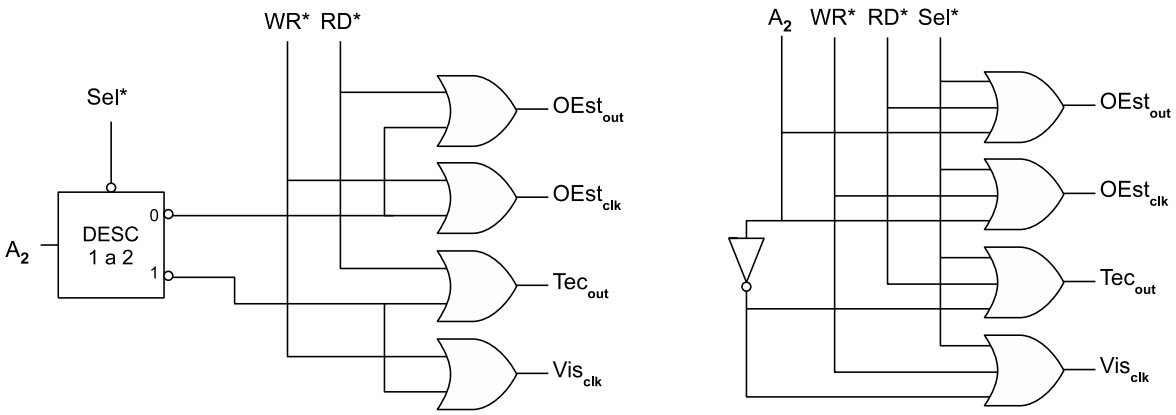

Figura 9.9: Dues versions del circuit de selecció dels registres de la interfície (problema 9, apartat 1).

3. Tractament de la interrupció de la línia $\overline{Int_2}$:

```
int2:   la $t0,0xFFFFFF00
        lb $t1,4($t0)     # llisc codi de tecla
        sb $t1,KP_Key     # actualitze variable
        li $t1,0xA0
        sb $t1,0($t0)     # conserve IE=1 i cancel·le interrupció
        j retexc
```

4. Inicialització del sistema complet:

```
    # inicialitze KeyPad:
        la $t0,0xFFFFFF00
        li $t1,0x20       # faig IE = 0
        sb $t1,0($t0)
    # inicialitze variables:
        li $t1,-1         # assigne KP_Key = -1
        sb $t1,KP_Key
    # inicialitze coprocessador d'interrupcions:
        li $t1,0x0403     # habilite int2 i IEc
        mtc0 $t1,$13
```

5. Codi del servei 11 del sistema:

```
fun11:  lb $v0,KP_Key     # $v0 = últim codi de tecla
        la $t0,0xFFFFFF00
        sb $v0,4($t0)     # faig còpia en visualitzador
        j retexc
```

6. La funció 513 permet als programes esperar fins que algú prema la tecla especificada com argument. Quan un programa d'usuari crida la funció 513, el tractament fa dues coses: emmagatzema l'argument (*$a0*) en la variable *KP_Key* i suspén el procés. La interrupció de *KeyPad* el reactivarà si el codi de la tecla coincideix amb la variable. Mirem-ho amb detall:

```
int2:   la $t0,0xFFFFFF00
        li $t1,0xA0       # Cancel·le interrupció
        sb $t1,0($t0)
        lb $t1,4($t0)     # Llig tecla
        lb $t2,KP_Key     # Llig variable
        bne $t1,$t2,retexc # si no són iguals, hem acabat
     ## Si tecla == KP_Key, reactiva el procés en espera:
```

```
jal activa_proc_en_espera
j retexc
```

L'especificació en podria ser la següent:

Servei	Codi	Paràmetres d'entrada	Paràmetres d'eixida
`wait_KP`	\$v0 = 513	—	\$a0 = tecla que cal esperar

■

PROBLEMA 10. Considereu un lloc de venda on els clients demanen torn quan hi arriben i els depenents atenen per ordre. El sistema anirà controlat per un computador amb tres perifèrics (vegeu la figura 9.10):

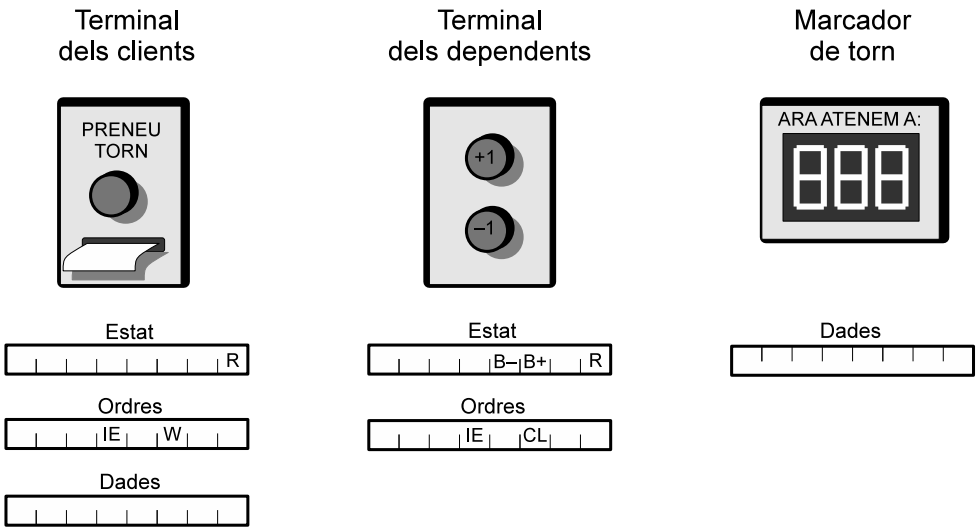

Figura 9.10: Els tres perifèrics del problema 10 i les seues interfícies.

- Un expendedor de torn format per un botó i una impressora que escriu un número en un tiquet. La interfície comprén tres registres d'1 byte.

  ESTAT (només lectura, adreça base + 0)

  R: (bit 0) s'activa cada vegada que un client prem el botó i la impressora està preparada.

  ORDRES (només escriptura, adreça base + 4), només escriptura

  IE: (bit 4) a 1 habilita la interrupció. Si $IE = 1$, la interrupció s'emetrà quan $R = 1$.

  W: (bit 2) té dos efectes: fa $R = 0$ (tot cancel·lant la interrupció si n'hi ha) i activa la impressora.

  DADES (només escriptura, adreça base + 8): el valor numèric (entre 0 i 255) que s'imprimeix en donar l'ordre W.

- Una botonera per als depenents amb dos botons retolats "+1" i "–1".

  ESTAT (només lectura, adreça base + 0):

  R: (bit 0) val 1 quan algú prem un botó qualsevol.

  B+: (bit 2) val 1 mentre el botó retolat "+1" està premut.

B–: (bit 3) val 1 mentre el botó retolat "–1" està premut.

ORDRES (només escriptura, adreça base + 4):

IE: (bit 4) a 1 habilita la interrupció. Si $IE = 1$, la interrupció s'emetrà quan $R = 1$.

CL: (bit 2) fa el bit $R = 0$ (tot cancel·lant la interrupció si està activa).

- Un marcador que està sempre preparat. La seua interfície només conté un registre de dades de 8 bits on s'escriu el valor numèric que ha d'aparéixer en el visualitzador de tres dígits.

El hardware ignora el valor dels bits no descrits en les interfícies anteriors. Els tres perifèrics estan connectats a un computador amb MIPS R2000. En la taula següent en teniu els detalls:

Perifèric	Adreça base	Línia d'interrupció
Expenedor per als clients	`0xFFFF0010`	$\overline{Int_1}$
Botonera dels depenents	`0xFFFF8040`	$\overline{Int_4}$
Marcador	`0xFFFFB000`	—

El manejador d'excepcions del sistema conté dues variables rellevants: `clients` indica quants clients han demanat torn i `servint` a quin número s'està atenent.

1. Escriviu un programa de prova de l'expenedor que es limite a esperar per consulta d'estat que es prema el botó de demanar torn i que imprimisca un tiquet amb el número "255". Aquest programa, útil per al diagnòstic del sistema, s'executarà en mode supervisor i podrà accedir sense restriccions a la interfície dels perifèrics.

2. Observeu aquest tractament de la interrupció $\overline{Int_4}$ dins del manejador. Expliqueu el que fa.

```
int4:   la $t0,0xFFFF8040
        li $t1,0x14
        sb $t1,4($t0)
        lb $t1,0($t0)
        li $t2,0xD
        bne $t1,$t2,L0
        sb $zero,clients
        sb $zero,servint
        la $t0,0xFFFFB000
        sb $zero,0($t0)
        j retexc
L0:     # ací ampliareu més endavant
        j retexc
```

3. Completeu el tractament de l'apartat 2 a partir de l'etiqueta `L0` per tal que, en prémer un dels dos botons del terminal, s'incremente o decremente la variable `servint` i el seu valor resultant es mostre en el marcador. Ignoreu la possibilitat de desbordament, perquè mai no vindran més de 50 clients en un dia qualsevol, i per la nit s'apaga el sistema.

4. Escriviu el tractament corresponent a la interrupció $\overline{Int_1}$. Quan un client prema el botó de l'expenedor, cal incrementar la variable `clients` i imprimir en el tiquet el seu valor.

5. Escriviu el tractament de la funció de sistema `get_clients` que té aquest perfil:

Funció	Índex (en `$v0`)	Resultats (en `$v0`)
`get_clients`	666	Nombre de torns donats

Suposeu que el manejador salta a l'etiqueta `fun666` quan la causa d'excepció és la instrucció `syscall` i `$v0 = 666`. Només heu d'escriure el codi adient a partir d'aquesta etiqueta.

SOLUCIÓ:

1. Programa de prova de l'expenedor. Abans de sincronitzar-se, el programa ha d'inhibir les interrupcions.

```
          la $t0,0xFFFF0010
          li $t1,0x04
          sb $t1,4($t0)
consulta: lb $t2,0($t0)  # bucle de consulta d'estat
          andi $t2,$t2,1
          beqz $t2,consulta
          li $t2,0xFF    # desa 255 en el registre de dades
          sb $t2,8($t0)
          sb $t1,4($t0)  # ordre d'escriptura i cancel·lació
```

2. Tractament inicial de la interrupció $\overline{Int_4}$. El tractament només atén el cas que estiguen premuts els dos botons, i inicialitza les variables `clients` i `servint` a 0.

3. Tractament complet de la interrupció $\overline{Int_4}$.

```
L0:       li $t2,0x8
          bne $t1,$t2,L1
          # tractament de B+
          lb $t1,servint
          addi $t1,$t1,+1
          sb $t1,servint
          la $t0,0xffffb000
          sb $t1,0$(t0)
          j retexc
L1:      # tractament de B-
          lb $t1,servint
          addi $t1,$t1,-1
          sb $t1,servint
          la $t0,0xffffb000
          sb $t1,0$(t0)
          j retexc
```

4. Tractament de la interrupció $\overline{Int_1}$. En donar l'ordre d'escriptura del tiquet, el tractament ha de deixar habilitades les interrupcions

```
int1:     la $t0,0xFFFF0010
          lb $t1,clients
          addi $t1,$t1,1
          sb $t1,clients
          sb $t1,8($t0)
          li $t1,0x14
          sb $t1,4($t0)
          j retxec
```

5. Tractament de la funció de sistema `get_clients`.

```
fun666:   lb $v0,clients
          j retxec
```

■

PROBLEMA 11. Un forn disposa de dos espais A i B separats per una paret (vegeu la figura 9.11, part esquerra). Els dos espais es comuniquen per un ventilador que fa circular l'aire. Per a control de la temperatura, s'han instal·lat dos termòmetres T_A i T_B, un en cada espai.

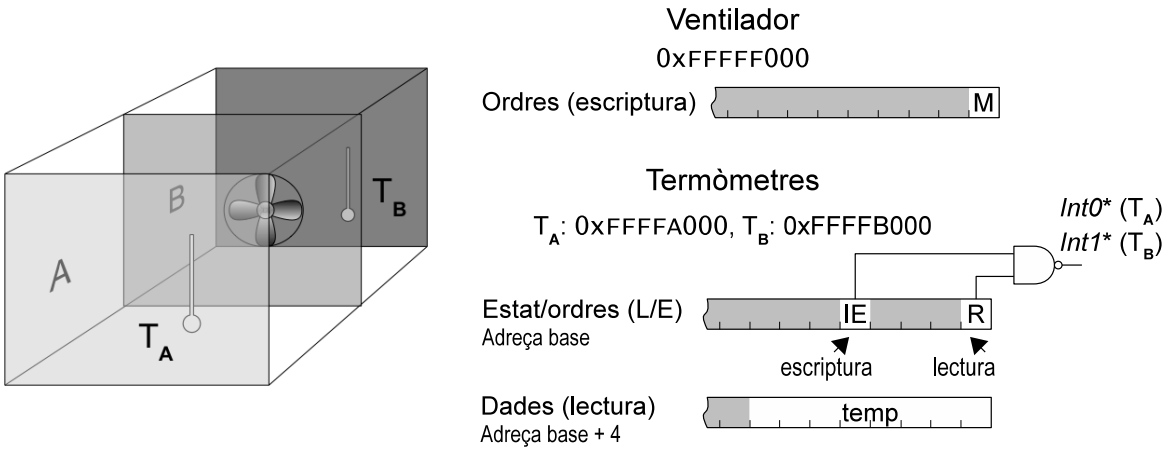

Figura 9.11: Esquema del forn i de les interfícies del ventilador i dels termòmetres (problema 11).

El ventilador i els termòmetres estan controlats per un computador mitjançant adaptadors amb les interfícies mostrades en la figura 9.11 (dreta) que detallem tot seguit:

Ventilador: El registre d'ordres (només escriptura, ubicat en l'adreça base de la interfície) conté el bit M per a controlar-lo. Escriure $M = 1$ activa el ventilador i $M = 0$ l'atura.

Termòmetres: Les interfícies de T_A i T_B són idèntiques i contenen dos registres: estat/ordres i dades:

- El registre d'estat/ordres (ubicat en l'adreça base) conté dos bits significatius: R (només lectura) en la posició 0 i el bit IE (només escriptura) en la posició 4
- El bit R, de perifèric preparat, pren el valor $R = 1$ quan canvia la temperatura i el valor $R = 0$ quan hi ha un accés de lectura al registre de dades
- El bit IE habilita les interrupcions. Si $IE = 1$, la interfície activa la línia d'interrupció a què està connectada mentre $R = 1$.
- El registre de dades (lectura) conté el valor de la temperatura en els 8 bits menys significatius . Es tracta d'un valor sense signe, de manera que el rang de representació és de $[0 \ldots 255]$ ^{o}C.

Els adaptadors estan connectats al computador amb les adreces base de la taula 9.1; a més a més, els adaptadors dels termòmetres estan connectats a les línies d'interrupció $\overline{Int_0}$ i $\overline{Int_1}$.

Perifèric	Adreça base	Línia d'interrupció
Termòmetre T_A	*0xFFFFA000*	$\overline{Int_0}$
Termòmetre T_B	*0xFFFFB000*	$\overline{Int_1}$
Ventilador	*0xFFFFF000*	—

Tabla 9.1: Connexió dels perifèrics del problema 11.

Tot suposant que la UCP del computador és un MIPS R2000:

1. Escriviu un programa (que s'executarà en mode supervisor) que espere, per consulta de l'estat, que $T_A > 100ºC$ per activar el motor.

2. Escriviu el tractament de les interrupcions dels termòmetres adient per a posar en marxa el ventilador si les temperatures en els espais A i en B són diferents i l'ature si les temperatures són iguals. Escriviu només el codi que cal incloure en el manejador a partir de les etiquetes `int0` i `int1` fins a la instrucció `j retexc`. Si us cal definir alguna variable del manejador, escriviu també la seua declaració en el segment de dades `kdata`.

3. Escriviu el codi de la crida al sistema `stop_fan` que atura el ventilador i deshabilita les interrupcions dels termòmetres.

4. Expliqueu el que fa la crida al sistema `functionxxx` si el seu codi i el tractament de la interrupció $\overline{Int_0}$ són aquests:

```
functionxxx:
            jal suspen_aquest_proces
            j retexc
int0:       la $t0, 0xFFFFA000
            li $t1, 0x10
            sb $t1, 0($t0)
            lbu $t1, 4($t0)
            li $t0, 200
            bne $t0, $t1, retexc
            jal activa_proc_en_espera
            j retexc
```

SOLUCIÓ:

1. Programa que espera, per consulta de l'estat, que $T_A > 100ºC$ per activar el ventilador

```
# Carregue adreça base i deshabilite interrupcions
            la $t0, 0xFFFFA000
            sb $0, 0($t0)
# Consulta de l'estat
espera:     lb $t1, 0($t0)
            andi $t1, $t1, 1
            beqz $t1, espera
# Comprova valor de temperatura
            lbu $t1, 4($t0)
            li $t2, 100
            sub $t1, $t1, $t2
            blez $t1, espera
# Activa el ventilador
            la $t0, 0xFFFFF000
            li $t1, 1
            sb $t1, 0($t0)
# Final
            li $v0, 10
            syscall
```

2. Codi dins del manejador:

   Variables:

```
tempA:      .byte 0
tempB:      .byte 0
```

Tractament de les interrupcions.

```
int0:       la $t0, 0xFFFFA000    # adreça base del termòmetre
            lbu $t1, 4($t0)       # llig temperatura (i cancel·la)
            sb $t1, tempA         # emmagatzema valor en la variable
            lbu $t0, tempB        # compara amb última temperatura
            beq $t0, $t1, parar   # de l'altre termòmetre
            j ventilar

int1:       la $t0, 0xFFFFB000
            lbu $t1, 4($t0)
            sb $t1, tempB
            lbu $t0, tempA
            beq $t0, $t1, parar

ventilar:   la $t0, 0xFFFFF000    # adreça base del ventilador
            li $t1, 1
            sw $t1, 0($t0)        # fa bit M=1
            j retexc

parar:      la $t0, 0xFFFFF000
            sw $0, 0($t0)         # fa bit M=0
            j retexc
```

3. Codi de la crida al sistema `stop_fan`

```
stop_fan:   la $t0, 0xFFFFA000
            sw $0, 0($t0)
            la $t0, 0xFFFFB000
            sw $0, 0($t0)
            la $t0, 0xFFFFF000
            sw $0, 0($t0)
            j retexc
```

4. La crida al sistema `functionxxx` permet als programes esperar fins que la temperatura en el recinte A siga $T_A = 200ºC$

   El codi que tracta la crida `stop_fan` es limita a deixar el programa en estat d'espera.

   El tratamiento de la interrupción $\overline{Int_0}$ lee la temperatura del termómetro y sólo si $T_A = 200$ cambia el estado del programa a preparado.

■

PROBLEMA 12. Es disposa d'un termòmetre a partir del qual s'ha d'implementar la funcionalitat d'un termògraf sobre un sistema basat en el MIPS R2000. Un termògraf és un dispositiu que permet emmagatzemar valors de temperatura registrats al llarg del temps per tal de processarlos posteriorment. El termòmetre presenta la interfície de la figura 9.12.
El termòmetre s'ubica en l'adreça base $\texttt{FFFFFF00}_{16}$. El registre de dades conté la temperatura actual, representada mitjançant un valor de 16 bits (*half word*). El registre de control és de només escriptura i conté dos bits d'interés (la resta de bits són ignorats pel hardware):

- **IE**: *Interrupt Enable*. Quan està a 1, el termòmetre provoca una interrupció cada 10 minuts per la línia int4*.
- **CLI**: *CLear Interrupt*. Quan s'escriu un 1 en aquest bit, el termòmetre desactiva la línia de petició d'interrupció. Aquesta acció s'ha de fer cada vegada que s'atén la interrupció a fi que el termòmetre puga produir una nova interrupció posteriorment.

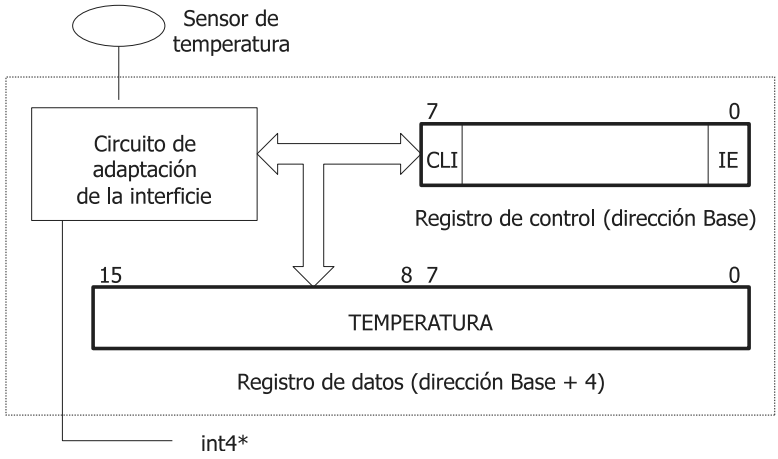

Figura 9.12: El termòmetre i la seua interfície (problema 12.

El termògraf s'implementa per mitjà d'una sèrie de funcions del sistema. Cada vegada que es produeix una interrupció del termòmetre (int4*) el sistema ha d'obtenir el valor de temperatura actual i copiar-lo a la primera posició lliure d'un vector de valors de temperatura. Les estructures de dades associades a la gestió d'aquest vector en el sistema són:

```
                .kdata
                ...
temperatures:   .half 0,0,0,...0 # Un nombre suficient de valors
nre_temps:      .word 0          # Nombre de temperatures registrades
                ...
```

1. Implementeu la rutina d'atenció a la interrupció int4*. A més de les accions necessàries per a la gestió del hardware, s'ha de llegir el nou valor de temperatura i emmagatzemar-lo en la primera posició lliure del vector de temperatures. La variable `nre_temps` ha d'incrementar-se adequadament. Com que cada valor de temperatura ocupa dos bytes, la primera posició lliure del vector es pot obtenir com a `nre_temps` multiplicat per dos (o sumat a sí mateix) més el desplaçament `temperatures`. Suposeu que el vector de temperatures és il·limitat, per la qual cosa no s'ha d'implementar la comprovació de límits del vector.

A fi que un programa d'usuari puga accedir a les dades emmagatzemats en el termògraf, s'ha de proporcionar la funció del sistema següent:

Servicio	Índice (`$v0`)	Paràmetres d'entrada	Paràmetres d'eixida
get_temps	31416	$\$a0$ = Adreça vector d'usuari	$\$v0$ = Nombre de mesures tornades

A la funció `get_temps` se li passa com a paràmetre un punter a la zona de memòria del programa d'usuari, on ha de disposar d'espai suficient per tal d'emmamatzemar els valors de temperatura que es puguen haver acumulat des de l'última vegada que la funció es va cridar. La funció copiarà els elements del vector `temperatures` a partir d'eixa adreça. En el registre `$v0` retornarà el nombre de valors de temperatura que s'han copiat a l'espai d'usuari. La funció ha de tenir, a més a més, l'efecte de buidar el vector de temperatures del sistema simplement posant a zero el comptador `nre_temps`.

2. Implementeu la funció del sistema `get_temps`. Només heu d'aportar el codi propi de la funció, ignorant la part d'identificació de funció i retorn. Podeu utilitzar els registrs `$t0`, `$t1`, `$t2` i `$t3`.

3. A banda de `get_temps`, suposeu que hi ha una funció (`wait_n_temps`) que, en invocar-se, posa al procés d'usuari en espera de que el termògraf registre *n* mesures. El seu perfil és el següent:

Funció	Índice (`$v0`)	Paràmetres d'entrada	Eixida
wait_n_temps	31417	$a0 = Nombre de dades per les quals s'esperarà	—

Implementeu un programa d'usuari que, en haver-se registrat 50 temperatures, les emmagatzeme en un vector de la grandària adequada. Per simplicitat, podeu suposar que aquest programa d'usuari és l'únic que accedeix a les funcions del termògraf.

SOLUCIÓ:

Apartat 1

```
int4: li $t0,0xFFFFFF00         # Adreça base del termòmetre
      lh $t1,4($t0)             # Llig temperatura
      lw $t2,nre_temps          # Llig nombre de temperatures
      add $t3,$t2,$t2           # Obtenir 2*num_temps
      sh  $t1,temperaturas($t3) # Afegir nova temperatura al vector
      addi $t2,$t2,1
      sw $t2,nre_temps          # Incrementar num_temps
      li $t1,0x81
      sb $t1,0($t0)             # CLI = 1; IE = 1
      b retexc
```

Apartat 2

```
get_temps:     lw $v0,nre_temps($0) # $v0 = nre_temps
               or $t0,$v0,$0        # $t0 = nombre de mesures acumulades
               la $t1,temperatures  # $t1 apunta a inici vector kernel
               or $t2,$a0,$0        # $t2 apunta a inici vector usuari
        bucle: beq $t0,$0,fi_get_temps
               lh $t3,0($t1)        # Copiar temperatura sobre $t3
               sh $t3,0($t2)        # Copiar temp. a espai d'usuari
               addi $t0,$t0,-1      # Decrementa comptador de mesures
               addi $t1,$t1,2       # Mou punter vector kernel
               addi $t2,$t2,2       # Mou punter vector usuari
               b bucle
fi_get_temps: b retexc
```

Apartat 3

```
            .data
meues_temps:.half 0,0,0...,0  # 50 posiciones

            .text
            ...
            li $v0,31417
            li $a0,50
            syscall           # Llamada a wait_n_temps
            li $v0,31416
            la $a0,meues_temps
            syscall           # Llamada a get_temps
            ...
```

■

Apéndice A

El sistema d'excepcions del MPIS R2000

La figura A.1 mostra el detall del registre d'estat del coprocessador d'excepcions del MIPS R2000. La figura A.2 mostra la relació entre els registres d'estat i de causa.

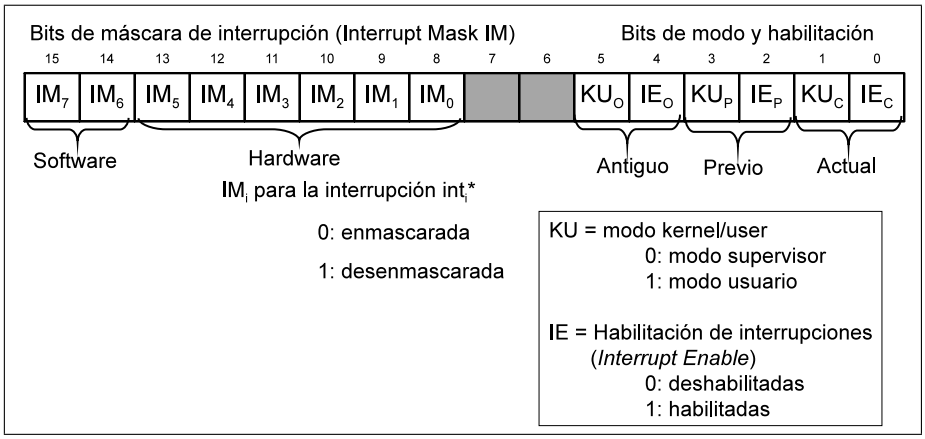

Figura A.1: Registre d'estat del MIPS R2000 (número $13)

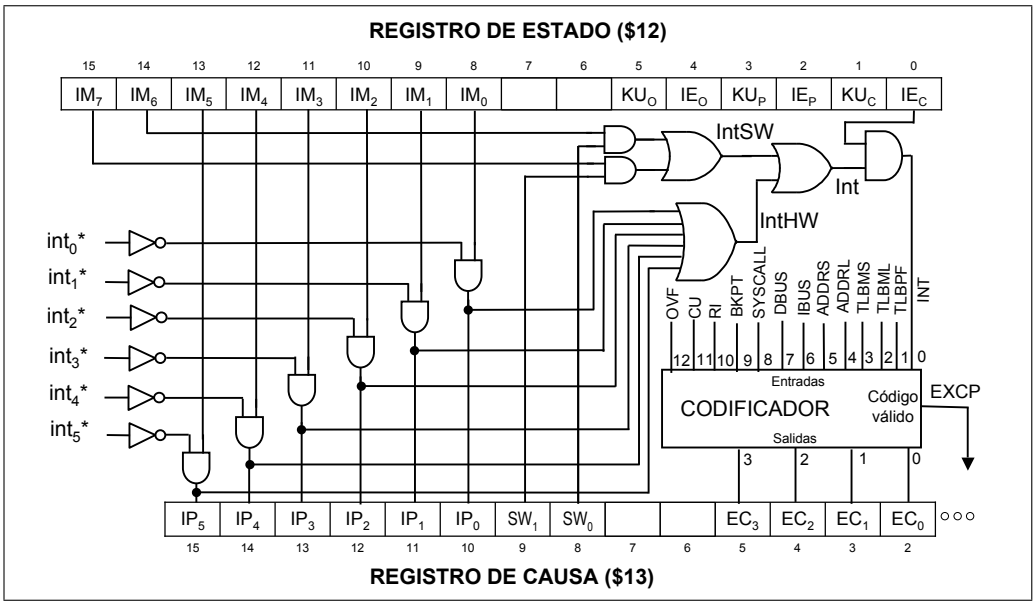

Figura A.2: Relació entre els registres d'excepció i de causa del MIPS R2000