Parte 1 de 5 -

Preguntas 1 de 6

2.0 Puntos

Tenemos una interfaz de consola en la dirección 0xFFFF2000 con los siguientes registros:

Registro de órdenes y estado (Lectura/escritura. Dirección = Base).
• R (bit 4, sólo lectura). Indicador de dispositivo preparado: R = 1 cuando la consola está disponible.
• E: (bit 7, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo, línea INT1*).
• C: (bit 0, escritura). Cancela la interrupción al escribir un '1', y R vuelve a valer 0 y se desactiva la línea de interrupción si la hubiera.
Registro de datos (Sólo escritura. Dirección = Base + 4).
Ambos registros son de 8 bits.

El siguiente programa escribe el carácter de la posición de memoria "Valor" en la consola, sincronizando por consulta de estado (la interrupción estará deshabilitada). Completar las instrucciones que hay detrás del bucle de consulta de estado (el cual no se muestra) que primero deben cancelar el bit R y luego escribir el valor en la consola.

```
        .data  0x1000000
Valor: .ascii 'p'

        la  $t0, 0xFFFF2000
... bucle de consulta de estado
        li $t1, 0x01
        sb $t1, 0($t0)
        lbu $t1, valor
        sb  $t1, 4($t0)
```

Parte 2 de 5 -

Preguntas 2 de 6

2.0 Puntos

Tenemos una interfaz de teclado en DB=0xFFFF0000 que tiene un registro de Estado/Ordenes de Lectura/escritura con. Dirección = Base, con los siguientes bits:

• R (bit 2, sólo lectura). Indicador de dispositivo preparado: R = 1 cada vez que se pulsa una tecla. Para hacer R = 0 es necesario realizar un acceso de lectura en el registro de datos.

• E: (bit 0, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo). Línea INT3*.

Y un registro de datos (Sólo lectura. Dirección = Base + 1).

4($t0)

• COD (bits 7...0). Código ASCII de la tecla pulsada. Leer de este registro provoca que R = 0.

Seleccionar las instrucciones que usarías indicando el orden para realizar un bucle de consulta de estado que espere hasta que se haya pulsado una tecla y posteriormente borre el bit R. Asumir que en $t1 hemos cargado la dirección base.
NOTA: el programa sólo tiene 4 instrucciones, hay 3 que no son correctas (un branch, un and y una load o store)

A.

```
andi $t0, $t0, 0x01
```

B.

```
lbu $t0, 1($t1)
```

```
andi $t0, $t0, 0x01
```

B.

```
lbu $t0, 1($t1)
```

C.

```
lbu $t0, 0($t1)
```

D.

```
beqz $t0, espera
```

E.

```
sb $zero, 0($t1)
```

F.

```
andi $t0, $t0, 0x04
```

G.

```
bnez $t0, espera
```

C	seleccionar	1. (1) espera:
F	seleccionar	2. (2)
D	seleccionar	3. (3)
E	seleccionar	4. (4)
G	seleccionar	5. (branch incorrecto)
A	seleccionar	6. (and incorrecto)
B	seleccionar	7. (load o store incorrecto)

Parte 3 de 5 -

Recibidos (1) - cem2100 | https://poliformat.upv.es/

https://poliformat.upv.es/portal/site/GRA_11552_2014/page/62d60405-3c94-4f73-9fa1-c98a1d2e1b74

Preguntas 3 de 6

2.0 Puntos

Completad las siguientes instrucciones de un fragmento de código de inicialización para que habilite las interrupciones generales y las línea INT4 y active el modo usuario:

li $t0, 0x 1003

MTC0 $t0, $12

Completad las siguientes instrucciones de un fragmento de código para que habilite las interrupciones generales, habilite la línea INT2, deshabilite la INT1 e INT3 y deje el resto del registro de estado sin modificar:

mfc0 $t0, $12

ori $t0, $t0, 0x 0801 0401

andi $t0, $t0, 0x F5FF F5FF

mtc0 $t0, $12

Parte 4 de 5 -

Preguntas 4 de 6

2.0 Puntos

La interfaz del reloj de PCSpim es la siguiente:

Dirección Base=FFFF0010, con un único registro de órdenes y estado. Interrumpe por la línea Int2

Registro de órdenes y estado (Lectura/escritura)

- R (bit 1, lectura/escritura). Indicador de dispositivo preparado: R = 1 a cada segundo.
  0 Cancelación (R = 0): se escribirá en bit R un 0. mtc0
- E (bit 0, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo)

Considere el siguiente código de inicio del sistema:

```
la $t0,0xFFFF0010
li $t1,3
sb $t1,0($t0)
li $t1,0x403
mtc0 $t1,$12
```

Indique qué afirmaciones de las siguientes son correctas:

- [ ] A. Este código no puede ser de inicio. Necesariamente formará parte del manejador de interrupciones
- [x] B. El código habilita completamente la interrupción del reloj
- [ ] C. El código cancela el bit R del reloj
- [ ] D. Este código prepara el sistema para sincronizarse con el reloj mediante consulta de estado

ES 15:12 28/05/2015

```
li $t1,0x403

mtc0 $t1,$12
```

Indique qué afirmaciones de las siguientes son correctas:

- [ ] A. Este código no puede ser de inicio. Necesariamente formará parte del manejador de interrupciones
- [ ] B. El código habilita completamente la interrupción del reloj
- [ ] C. El código cancela el bit R del reloj
- [ ] D. Este código prepara el sistema para sincronizarse con el reloj mediante consulta de estado

Parte 5 de 5 -

Preguntas 5 de 6 — 1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué punto se utilizarían las siguientes acciones o instrucciones:

```
mfc0 $k0, $14 # EPC
addi $k0,$k0,4
sw $k0, dirret
```

- ○ A.Se hace en el código de la llamada al sistema (después de saltar a la etiqueta print_char)
- ○ B.Se utilizan en el código del proceso usuario cuando hace una llamada al sistema
- ○ C.Se hace en la inicialización del sistema operativo
- ● D.Se hace en el manejador de excepciones después de comprobar que se trata de una llamada al sistema, pero antes de analizar $v0
- ○ E.Se hace en el manejador de excepciones al inicio.

Borra selección

Preguntas 6 de 6 — 1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué punto se utilizarían las siguientes acciones o instrucciones:

```
li $v0,0x12
li $a0,0x41 # código del carácter A
syscall
```

Exámenes

- ○ A.Se hace en el manejador de execepciones al inicio.
- ○ B.Se hace en el manejador de excepciones después de comprobar que se trata de una llamada al sistema, pero antes de analizar $v0
- ○ C.Se hace en la inicialización del sistema operativo
- ○ D.Se hace en el código de la llamada al sistema (después de saltar a la etiqueta print_char)
- ● E.Se utilizan en el código del proceso usuario cuando hace una llamada al sistema

Borra selección

Salvar | Enviar para calificar

Preguntas 1 de 6

2.0 Puntos

Tenemos un interfaz de consola en la dirección 0xFFFF0100 con los siguientes registros:

Registro de órdenes y estado (Lectura/escritura. Dirección = Base).

- R (bit 2, lectura/escritura). Indicador de dispositivo preparado: el hardware hace R = 1 cuando la consola está disponible. Para cancelar, el programa ha de escribir R=0 .
- E: (bit 7, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo, línea INT1*).

Registro de datos (Sólo escritura. Dirección = Base + 1).
Ambos registros son de 8 bits.

El siguiente programa escribe el carácter de la posición de memoria "Letra" en la consola, sincronizando por consulta de estado (la interrupción estará deshabilitada). Completad las instrucciones que hay detrás del bucle de consulta de estado (que no se muestra) que primero deben cancelar el bit R y luego escribir el carácter en la consola.

```
        .data  0x1000000
Letra: .ascii 'p'

        la  $t0, 0xFFFF0100

... bucle de consulta de estado

        sb  $zero , 0($t0)

        lbu $t1,  letra
                  4($t0)
```

2.0 Puntos

Tenemos una interfaz de reloj en DB=0xFFFF0010 que tiene un único registro de Estado/Ordenes accesible para lectura y escritura. Su contenido es :

- R (bit 2, lectura/escritura). Indicador de dispositivo preparado: R = 1 a cada segundo.
  - Cancelación: se escribirá un 0 el en bit R .
- E: (bit 0, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo)

El dispositivo está conectado a la línea de interrupción INT2*.

Ordena las instrucciones para que el código realice un bucle de consulta de estado que espere hasta que haya pasado un segundo para (1) cancelar el bit R y (2) incrementar la variable *tiempo* de tipo word. Asumir que en $t0 está cargada la dirección base de la interfaz.

A.

```
la $t0,tiempo
```

B.

H,B,D,G,A,F,C,E,

```
andi $t1,$t1,4
```

C.

```
addiu $t1,$t1,1
```

D.

```
beqz $t1,bucle
```

E.

```
sw $t1,0($t0)
```

F.

```
lw $t1,0($t0)
```

G.

```
sb $zero,0($t0)
```

H.

```
lb $t1,0($t0)
```

Parte 3 de 5 -

Preguntas 3 de 6

2.0 Puntos

Completar las siguientes instrucciones de un fragmento de código de inicialización que habilite las interrupciones generales, la INT3 y activa el modo usuario:

```
li $t0, 0x [1003] 0803
   mtc0   $t0, $12
```

Completar las siguientes instrucciones de un fragmento de código que habilita las interrupciones generales, la INT1,INT3 y deshabilita la INT2 dejando el resto sin modificar:

```
   MFC0   $t0, $12
ori $t0, $t0, 0x 0801
andi $t0, $t0, 0x f9ff
   mtc0   $t0, $12
```

Parte 4 de 5 -

Preguntas 4 de 6

2.0 Puntos

La interfaz del reloj de PCSpim es la siguiente:

Dirección Base=FFFF0010, con un único registro de órdenes y estado. Interrumpe por la línea Int2

Registro de órdenes y estado (Lectura/escritura)

- R (bit 1, lectura/escritura). Indicador de dispositivo preparado: R = 1 a cada segundo.
  - Cancelación (R = 0): se escribirá en bit R un 0.
- E: (bit 0, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo)

Considere el siguiente código de inicio del sistema:

```
la $t0,0xFFFF0010
li $t1,3
sb $t1,0($t0)
li $t1,0x403
mtc0 $t1,$12
```

Indique qué afirmaciones de las siguientes son correctas:

- [ ] A. Este código no puede ser de inicio. Necesariamente formará parte del manejador de interrupciones
- [ ] B. Este código prepara el sistema para sincronizarse con el reloj mediante consulta de estado
- [ ] C. El código cancela el bit R del reloj
- [x] D. El código habilita completamente la interrupción del reloj

Parte 5 de 5 -

Preguntas 5 de 6

1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué punto se utilizarían las siguientes acciones o instrucciones:

```
.set noat
sw $at,0($k1)
.set at
```

- ( ) A.Se hace en la inicialización del sistema operativo
- ( ) B.Se hace en el manejador de excepciones después de comprobar que se trata de una llamada al sistema, pero antes de analizar $v0
- ( ) C.Se utilizan en el código del proceso usuario cuando hace una llamada al sistema
- (x) D.Se hace en el manejador de excepciones al inicio.
- (x) E.Se hace en el código de la llamada al sistema (después de saltar a la etiqueta print_char)

Borra selección

Preguntas 6 de 6

1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué punto se utilizarían las siguientes acciones o instrucciones:

```
li $v0,0x12
li $a0,0x41 # código del carácter A
syscall
```

- A.Se hace en el manejador de excepciones después de comprobar que se trata de una llamada al sistema, pero antes de analizar $v0
- B.Se hace en el manejador de execepciones al inicio.
- C.Se hace en la inicialización del sistema operativo
- D.Se hace en el código de la llamada al sistema (después de saltar a la etiqueta print_char)
- E.Se utilizan en el código del proceso usuario cuando hace una llamada al sistema

Borra selección

Salvar | Enviar para calificar

Inicio
Guía Docente
Recursos
Espacio compartido
Tareas
Exámenes
Calificaciones
Sondeos
Calendario
Anuncios
Grupos
Correo interno
Foros
Chat
Contenidos
Videoapuntes
Corrector ALCE
Wiki

Etc: Exámenes

Examen Laboratorio 04

Tabla de Contenidos

Tiempo restante: 0:39:44

Ocultar/Mostrar el tiempo restante

Parte 1 de 5 -

Preguntas 1 de 6 — 2.0 Puntos

Tenemos una interfaz de consola en la dirección 0xFFFF2000 con los siguientes registros:

Registro de órdenes y estado (Lectura/escritura. Dirección = Base).
• R (bit 4, sólo lectura). Indicador de dispositivo preparado: R = 1 cuando la consola está disponible.
• E: (bit 7, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo, línea INT1*).
• C: (bit 2, escritura). Cancela la interrupción al escribir un '1', y R vuelve a valer 0 y se desactiva la línea de interrupción si la hubiera.
Registro de datos (Sólo escritura. Dirección = Base + 4).
Ambos registros son de 8 bits.

El siguiente programa escribe el carácter de la posición de memoria "Valor" en la consola, sincronizando por consulta de estado (la interrupción estará deshabilitada). Completar las instrucciones que hay detrás del bucle de consulta de estado (el cual no se muestra) que primero deben cancelar el bit R y luego escribir el valor en la consola.

```
        .data  0x1000000
Valor: .ascii 'p'

        la  $t0, 0xFFFF2000
... bucle de consulta de estado
        li $t1, 0x 10      04
        sb $t1,  0($t0)
        lbu $t1, valor
        sb  $t1, 4($t0)
```

Parte 2 de 5 -

Preguntas 2 de 6 — 2.0 Puntos

Tenemos una interfaz de teclado en DB=0xFFFF0000 que tiene un registro de 8 bits de Estado/Ordenes, de Lectura/escritura con Dirección = DB, con los siguientes bits:

• R (bit 3, sólo lectura). Indicador de dispositivo preparado: R = 1 cada vez que se pulsa una tecla. Para hacer R = 0 es necesario realizar un acceso de lectura en el

```
sb $t1,
lbu $t1,
sb  $t1,
```

Parte 2 de 5 -

Preguntas 2 de 6

2.0 Puntos

Tenemos una interfaz de teclado en DB=0xFFFF0000 que tiene un registro de 8 bits de Estado/Ordenes, de Lectura/escritura con Dirección = DB, con los siguientes bits:

• R (bit 3, sólo lectura). Indicador de dispositivo preparado: R = 1 cada vez que se pulsa una tecla. Para hacer R = 0 es necesario realizar un acceso de lectura en el registro de datos.

• E: (bit 0, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo). Línea INT3*.

Y un registro de datos (Sólo lectura. Dirección = DB + 1).

• COD (bits 7...0). Código ASCII de la tecla pulsada. Leer de este registro provoca que R = 0.

Seleccionar las instrucciones que usarías indicando el orden para realizar un bucle de consulta de estado que espere hasta que se haya pulsado una tecla y posteriormente borre el bit R. Asumir que en $t1 hemos cargado la dirección base.
NOTA: el programa sólo tiene 4 instrucciones, hay 3 que no son correctas (un branch, un and y una load o store)

A.

```
andi $t0, $t0, 0x08
```

B.

```
beqz $t0, espera
```

C.

```
sb $zero, 0($t1)
```

D.

```
lbu $t0, 0($t1)
```

E.

```
andi $t0, $t0, 0x01
```

F.

```
bnez $t0, espera
```

G.

```
lbu $t0, 1($t1)
```

1.

B.

```
beqz $t0, espera
```

C.

```
sb $zero, 0($t1)
```

D.

```
lbu $t0, 0($t1)
```

E.

```
andi $t0, $t0, 0x01
```

F.

```
bnez $t0, espera
```

G.

```
lbu $t0, 1($t1)
```

D seleccionar — 1. (1) `espera:`

A seleccionar — 2. (2)

B seleccionar — 3. (3)

G seleccionar — 4. (4)

F seleccionar — 5. (branch incorrecto)

E sel — 6.

C sel — o)

Parte 3 de 5 -

F

4. seleccionar (4)

5. seleccionar (branch incorrecto)

6. seleccionar (and incorrecto)

7. seleccionar (load o store incorrecto)

Parte 3 de 5 -

Preguntas 3 de 6 — 2.0 Puntos

Completar las siguientes instrucciones de un fragmento de código de inicialización que habilite las interrupciones generales, la INT5 y activa el modo usuario:

```
li $t0, 0x 2003
mtc0       $t0, $12
```

Completar las siguientes instrucciones de un fragmento de código que habilita las interrupciones generales, la INT0,INT3 y deshabilita la INT1 dejando el resto sin modificar:

```
mfc0       $t0, $12
ori $t0, $t0, 0x 0501   0901
andi $t0, $t0, 0x fdff
mtc0       $t0, $12
```

Parte 4 de 5 -

Preguntas 4 de 6 — 2.0 Puntos

La interfaz del reloj de PCSpim es la siguiente:

Dirección Base=0xFFFF0010, con un único registro de órdenes y estado. Interrumpe por la línea Int2

```
andi $t0, $t0, 0x
                  $t0, $12
```

Parte 4 de 5 -

Preguntas 4 de 6

2.0 Puntos

La interfaz del reloj de PCSpim es la siguiente:

Dirección Base=0xFFFF0010, con un único registro de órdenes y estado. Interrumpe por la línea Int2

Registro de órdenes y estado (Lectura/escritura)

- R (bit 1, lectura/escritura). Indicador de dispositivo preparado: R = 1 a cada segundo.
  - Cancelación (R = 0): se escribirá en bit R un 0.
- E: (bit 0, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo)

Considere que el inicio del sistema ha habilitado completamente la interrupción del reloj y que el manejador contiene el siguiente tratamiento:

```
int2:   la $t0,0xFFFF0010
        li $t1,3
        sb $t1,0($t0)
        j retexc
```

Indique qué afirmaciones de las siguientes son correctas:

- [ ] A. El código cancela el bit R
- [ ] B. El manejador sólo atenderá la primera interrupción del reloj porque el tratamiento la inhabilita
- [ ] C.

Este tratamiento de la interrupción permite ofrecer servicios de espera a los procesos

- [x] D. Al terminar el manejador, la interrupción del reloj continuará activa y el manejador volverá a ejecutarse

Parte 5 de 5 -

Preguntas 5 de 6

1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué punto se utilizarían las siguientes acciones o instrucciones:

```
li $v0,0x12
li $a0,0x41 # código del carácter A
```

Parte 5 de 5 -

Preguntas 5 de 6 — 1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué punto se utilizarían las siguientes acciones o instrucciones:

```
li $v0,0x12
li $a0,0x41 # código del carácter A
syscall
```

- (●) A.Se utilizan en el código del proceso usuario cuando hace una llamada al sistema
- ( ) B.Se hace en el manejador de excepciones después de comprobar que se trata de una llamada al sistema, pero antes de analizar $v0
- ( ) C.Se hace en la inicialización del sistema operativo
- ( ) D.Se hace en el código de la llamada al sistema (después de saltar a la etiqueta print_char)
- ( ) E.Se hace en el manejador de execepciones al inicio.

Borra selección

Preguntas 6 de 6 — 1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué punto se utilizarían las siguientes acciones o instrucciones:

```
acceder al adaptador del dispositivo (consola en el caso de print_char)
terminar ejecutando: b retexc
```

- ( ) A.Se hace en la inicialización del sistema operativo
- ( ) B.Se utilizan en el código del proceso usuario cuando hace una llamada al sistema
- ( ) C.Se hace en el manejador de excepciones después de comprobar que se trata de una llamada al sistema, pero antes de analizar $v0
- ( ) D.Se hace en el manejador de excepciones al inicio.
- (●) E.Se hace en el código de la llamada al sistema (después de saltar a la etiqueta print_char)

Borra selección

Salvar | Enviar para calificar

Tiempo restante: 0:39:28

Ocultar/Mostrar el tiempo restante

Parte 1 de 5 -

Preguntas 1 de 6 — 2.0 Puntos

Tenemos una interfaz de consola en la dirección 0xFFFF2000 con los siguientes registros:

Registro de órdenes y estado (Lectura/escritura. Dirección = Base).
• R (bit 4, sólo lectura). Indicador de dispositivo preparado: R = 1 cuando la consola está disponible.
• E: (bit 7, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo, línea INT1*).
• C: (bit 2, escritura). Cancela la interrupción al escribir un '1', y R vuelve a valer 0 y se desactiva la línea de interrupción si la hubiera.
Registro de datos (Sólo escritura. Dirección = Base + 4).
Ambos registros son de 8 bits.

El siguiente programa escribe el carácter de la posición de memoria "Valor" en la consola, sincronizando por consulta de estado (la interrupción estará deshabilitada). Completar las instrucciones que hay detrás del bucle de consulta de estado (el cual no se muestra) que primero deben cancelar el bit R y luego escribir el valor en la consola.

```
        .data  0x1000000
Valor: .ascii 'p'

        la  $t0, 0xFFFF2000
... bucle de consulta de estado
        li $t1, 0x 04
        sb $t1,  0($t0)
        lbu $t1, valor
        sb  $t1, 4($t0)
```

Parte 2 de 5 -

Preguntas 2 de 6 — 2.0 Puntos

Tenemos una interfaz de teclado en DB=0xFFFF0000 que tiene un registro de 8 bits de Estado/Ordenes, de Lectura/escritura con Dirección = DB, con los siguientes bits:

• R (bit 3, sólo lectura). Indicador de dispositivo preparado: R = 1 cada vez que se pulsa una tecla. Para hacer R = 0 es necesario realizar un acceso de lectura en el registro de datos.

• E: (bit 0, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo). Línea INT3*.

Y un registro de datos (Sólo lectura. Dirección = DB + 1).

• COD (bits 7...0). Código ASCII de la tecla pulsada. Leer de este registro provoca que R = 0.

Parte 2 de 5 -

Preguntas 2 de 6

2.0 Puntos

Tenemos una interfaz de teclado en DB=0xFFFF0000 que tiene un registro de 8 bits de Estado/Ordenes, de Lectura/escritura con Dirección = DB, con los siguientes bits:

• R (bit 3, sólo lectura). Indicador de dispositivo preparado: R = 1 cada vez que se pulsa una tecla. Para hacer R = 0 es necesario realizar un acceso de lectura en el registro de datos.

• E: (bit 0, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo). Línea INT3*.

Y un registro de datos (Sólo lectura. Dirección = DB + 1).

• COD (bits 7...0). Código ASCII de la tecla pulsada. Leer de este registro provoca que R = 0.

Seleccionar las instrucciones que usarías indicando el orden para realizar un bucle de consulta de estado que espere hasta que se haya pulsado una tecla y posteriormente borre el bit R. Asumir que en $t1 hemos cargado la dirección base.
NOTA: el programa sólo tiene 4 instrucciones, hay 3 que no son correctas (un branch, un and y una load o store)

A.

```
sb $zero, 0($t1)
```

B.

```
andi $t0, $t0, 0x01
```

C.

```
andi $t0, $t0, 0x08
```

D.

```
lbu $t0, 1($t1)
```

E.

```
bnez $t0, espera
```

F.

```
beqz $t0, espera
```

G.

```
lbu $t0, 0($t1)
```

---

1. seleccionar

```
(1) espera:
```

2.

NOTA: el programa solo tiene 4 instrucciones, hay 3 que no son correctas (un branch, un and y una load o store)

A.

```
sb $zero, 0($t1)
```

B.

```
andi $t0, $t0, 0x01
```

C.

```
andi $t0, $t0, 0x08
```

D.

```
lbu $t0, 1($t1)
```

E.

```
bnez $t0, espera
```

F.

```
beqz $t0, espera
```

G.

```
lbu $t0, 0($t1)
```

G	seleccionar	1. `(1) espera:`
C	seleccionar	2. `(2)`
F	seleccionar	3. `(3)`
D	seleccionar	4. `(4)`
E	seleccionar	5. `(branch incorrecto)`
B	seleccionar	6. `(and incorrecto)`
A	seleccionar	7. `(load o store incorrecto)`

7.
seleccionar (load o store incorrecto)

Parte 3 de 5 -

Preguntas 3 de 6 — 2.0 Puntos

Completad las siguientes instrucciones de un fragmento de código del manejador de excepciones para que cuando la causa de la excepción sea la interrupción 3 salte a la etiqueta int3, que cuando la causa de la excepción sea una instrucción syscall (código 8) salte a la etiqueta `sysfun` y que en cualquier otro caso salte a la etiqueta `retexc`.

```
mfc0 $k0, $13
andi $t0, $k0, 0x 003c
beq $t0, $zero, int
li $t1, 32
beq $t0, $t1, sysfun
int: andi $t0, $k0, 0x 32(en duda)
bne $t0,$zero,int3
j retexc
```

200C

Parte 4 de 5 -

Preguntas 4 de 6 — 2.0 Puntos

La interfaz del reloj de PCSpim es la siguiente:

Dirección Base=0xFFFF0010, con un único registro de órdenes y estado. Interrumpe por la línea Int2

Registro de órdenes y estado (Lectura/escritura)

- R (bit 1, lectura/escritura). Indicador de dispositivo preparado: R = 1 a cada segundo.
  - Cancelación (R = 0): se escribirá en bit R un 0.
- E: (bit 0, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo)

Considere que el inicio del sistema ha habilitado completamente la interrupción del reloj y que el manejador contiene el siguiente tratamiento:

```
                $t0,$zero,int3

j retexc
```

Parte 4 de 5 -

Preguntas 4 de 6

2.0 Puntos

La interfaz del reloj de PCSpim es la siguiente:

Dirección Base=0xFFFF0010, con un único registro de órdenes y estado. Interrumpe por la línea Int2

Registro de órdenes y estado (Lectura/escritura)

- R (bit 1, lectura/escritura). Indicador de dispositivo preparado: R = 1 a cada segundo.
  - Cancelación (R = 0): se escribirá en bit R un 0.
- E: (bit 0, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo)

Considere que el inicio del sistema ha habilitado completamente la interrupción del reloj y que el manejador contiene el siguiente tratamiento:

```
int2:   la $t0,0xFFFF0010
        li $t1,3
        sb $t1,0($t0)
        j retexc
```

Indique qué afirmaciones de las siguientes son correctas:

- [ ] A. El código cancela el bit R
- [ ] B. El manejador sólo atenderá la primera interrupción del reloj porque el tratamiento la inhabilita
- [x] C. Al terminar el manejador, la interrupción del reloj continuará activa y el manejador volverá a ejecutarse
- [ ] D.

Este tratamiento de la interrupción permite ofrecer servicios de espera a los procesos

Parte 5 de 5 -

Preguntas 5 de 6

1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué punto se utilizarían las siguientes acciones o instrucciones:

```
li $v0,0x12
li $a0,0x41 # código del carácter A
syscall
```

Parte 5 de 5 -

Preguntas 5 de 6

1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué punto se utilizarían las siguientes acciones o instrucciones:

```
li $v0,0x12
li $a0,0x41 # código del carácter A
syscall
```

- A.Se hace en el manejador de execepciones al inicio.
- B.Se hace en el manejador de excepciones después de comprobar que se trata de una llamada al sistema, pero antes de analizar $v0
- C.Se hace en el código de la llamada al sistema (después de saltar a la etiqueta print_char)
- D.Se hace en la inicialización del sistema operativo
- E.Se utilizan en el código del proceso usuario cuando hace una llamada al sistema

Borra selección

Preguntas 6 de 6

1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué punto se utilizarían las siguientes acciones o instrucciones:

```
acceder al adaptador del dispositivo (consola en el caso de print_char)
terminar ejecutando: b retexc
```

- A.Se hace en el código de la llamada al sistema (después de saltar a la etiqueta print_char)
- B.Se hace en el manejador de excepciones después de comprobar que se trata de una llamada al sistema, pero antes de analizar $v0
- C.Se hace en la inicialización del sistema operativo
- D.Se utilizan en el código del proceso usuario cuando hace una llamada al sistema
- E.Se hace en el manejador de excepciones al inicio.

Borra selección

Salvar | Enviar para calificar

Inicio
Guía Docente
Recursos
Espacio compartido
Tareas
Exámenes
Calificaciones
Sondeos
Calendario
Anuncios
Grupos
Correo interno
Foros
Chat
Contenidos
Videoapuntes
Corrector ALCE
Wiki

Etc: Exámenes

Examen Laboratorio 04

Tabla de Contenidos

Tiempo restante: 0:24:48

Ocultar/Mostrar el tiempo restante

Parte 1 de 5 -

Preguntas 1 de 6 — 2.0 Puntos

Tenemos una interfaz de consola en la dirección 0xFFFF2000 con los siguientes registros:

Registro de órdenes y estado (Lectura/escritura. Dirección = Base).
• R (bit 4, sólo lectura). Indicador de dispositivo preparado: R = 1 cuando la consola está disponible.
• E: (bit 7, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo, línea INT1*).
• C: (bit 0, escritura). Cancela la interrupción al escribir un '1', y R vuelve a valer 0 y se desactiva la línea de interrupción si la hubiera.
Registro de datos (Sólo escritura. Dirección = Base + 4).
Ambos registros son de 8 bits.

El siguiente programa escribe el carácter de la posición de memoria "Valor" en la consola, sincronizando por consulta de estado (la interrupción estará deshabilitada). Completar las instrucciones que hay detrás del bucle de consulta de estado (el cual no se muestra) que primero deben cancelar el bit R y luego escribir el valor en la consola.

```
        .data  0x1000000
Valor: .ascii 'p'

        la  $t0, 0xFFFF2000
... bucle de consulta de estado
        li $t1, 0x [01]
        sb $t1, [0($t0)]
        lbu $t1, [Valor]
        sb  $t1, [4($t0)]
```

Parte 2 de 5 -

Preguntas 2 de 6 — 2.0 Puntos

Tenemos una interfaz de teclado en DB=0xFFFF0000 que tiene un registro de Estado/Ordenes de Lectura/escritura con. Dirección = Base, con los siguientes bits:

• R (bit 2, sólo lectura). Indicador de dispositivo preparado: R = 1 cada vez que se pulsa una tecla. Para hacer R = 0 es necesario realizar un acceso de lectura en el registro de datos.

• E: (bit 0, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo). Línea INT3*.

Preguntas 2 de 6

2.0 Puntos

Tenemos una interfaz de teclado en DB=0xFFFF0000 que tiene un registro de Estado/Ordenes de Lectura/escritura con. Dirección = Base, con los siguientes bits:

• R (bit 2, sólo lectura). Indicador de dispositivo preparado: R = 1 cada vez que se pulsa una tecla. Para hacer R = 0 es necesario realizar un acceso de lectura en el registro de datos.

• E: (bit 0, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo). Línea INT3*.

Y un registro de datos (Sólo lectura. Dirección = Base + 1).

• COD (bits 7...0). Código ASCII de la tecla pulsada. Leer de este registro provoca que R = 0.

Seleccionar las instrucciones que usarías indicando el orden para realizar un bucle de consulta de estado que espere hasta que se haya pulsado una tecla y posteriormente borre el bit R. Asumir que en $t1 hemos cargado la dirección base.
NOTA: el programa sólo tiene 4 instrucciones, hay 3 que no son correctas (un branch, un and y una load o store)

A.

```
andi $t0, $t0, 0x04
```

B.

```
lbu $t0, 1($t1)
```

C.

```
sb $zero, 0($t1)
```

D.

```
bnez $t0, espera
```

E.

```
andi $t0, $t0, 0x01
```

F.

```
beqz $t0, espera
```

G.

```
lbu $t0, 0($t1)
```

---

1. G ▼ `(1) espera:`
2. A ▼ `(2)`
3. F ▼ `(3)`
4. B ▼ `(4)`
5. D ▼ `(branch incorrecto)`
6. E ▼ `(and incorrecto)`
7. C ▼ `(load o store incorrecto)`

Parte 3 de 5 -

Preguntas 3 de 6 — 2.0 Puntos

Completad las siguientes instrucciones de un fragmento de código del manejador de excepciones para que cuando la causa de la excepción sea la interrupción 3 salte a la etiqueta **int3**, que cuando la causa de la excepción sea una instrucción syscall (código 8) salte a la etiqueta `sysfun` y que en cualquier otro caso salte a la etiqueta `retexc`.

mfc0

```
mtc0        $k0, $13
andi $t0, $k0, 0x 003C   200C
beq $t0, $zero, int
li $t1, 0020   32
beq $t0, $t1, sysfun
int:  andi $t0, $k0, 0x 2000
bne         $t0,$zero,int3
j retexc
```

Els profesors son uns fills de puta i t'ho demanen així en decimal, fixa't que no posa "0x....." com si que ho posa en altres.

en hexadecimal esta be el 20 pero en decimal es 32

com así ho demanen en decimal, pues es 32

Parte 4 de 5 -

Preguntas 4 de 6 — 2.0 Puntos

La interfaz de teclado del simulador PCSPIM contiene dos registros de un byte ubicados a partir de la direción base DB = 0xFFFF0000 y tiene la posibilidad de interrumpir por la linea irq0*. La descripción de los registros es la siguiente:

Registro de Estado/Órdenes (lectura/escritura, dirección = DB)

- R (bit 0, sólo lectura). Indicador de dispositivo preparado. R=1 cada vez que se pulsa una tecla.
  - Cancelación: Para hacer que R=0, es necesario leer el registro de datos.
- E (bit 1, lectura/escritura). Habilita la petición de interrupciones del dispositivo. Mientras E=1, se activará irq0* cada vez que R=1.

Registro de Datos (Sólo lectura, dirección = DB + 4)

- COD (bits 0..7). Código ASCII de la última tecla pulsada. Leer de este registro provoca la puesta a cero del bit R.

Supón que, en el arranque del sistema, se ha habilitado completamente la interrupción del teclado. Considere la siguiente rutina de servicio de la interrupción 0:

```
irq0:  la $t0,0xFFFF0000
       lb $t1,0($t0)
       andi $t1,$t1,2
       sb $t1,0($t0)
       b retexc
```

Indique qué afirmaciones de las siguientes son correctas sobre esta rutina de servicio:

- [ ] A. El carácter leido se almacena en el registro $t1.
- [ ] B. Accede al teclado por consulta de estado.
- [x] C. No cancela correctamente la petición de interrupción.
- [ ] D. Cancela correctamente la petición de interrupción.

Parte 5 de 5 -

Parte 5 de 5 -

Preguntas 5 de 6 — 1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué punto se utilizarían las siguientes acciones o instrucciones:

```
mfc0 $k0, $14 # EPC
addi $k0,$k0,4
sw $k0, dirret
```

- A. Se utilizan en el código del proceso usuario cuando hace una llamada al sistema
- B. Se hace en el código de la llamada al sistema (después de saltar a la etiqueta print_char)
- C. Se hace en la inicialización del sistema operativo
- (•) D. Se hace en el manejador de excepciones después de comprobar que se trata de una llamada al sistema, pero antes de analizar $v0
- E. Se hace en el manejador de excepciones al inicio.

Borra selección

Preguntas 6 de 6 — 1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué punto se utilizarían las siguientes acciones o instrucciones:

```
acceder al adaptador del dispositivo (consola en el caso de print_char)
terminar ejecutando: b retexc
```

- A. Se hace en el manejador de excepciones después de comprobar que se trata de una llamada al sistema, pero antes de analizar $v0
- B. Se hace en la inicialización del sistema operativo
- C. Se utilizan en el código del proceso usuario cuando hace una llamada al sistema
- D. Se hace en el manejador de excepciones al inicio.
- (•) E. Se hace en el código de la llamada al sistema (después de saltar a la etiqueta print_char)

Borra selección

Salvar | Enviar para calificar

(3) li $to, 0x 8003

mtc0 $to, $12

---

mfc0 $to, $12

ori $to, $to, 0x 0801

andi $to, $to, 0x FBFF

mtc0 $to, $12

Habilitan i. generales, I
activa modo usuar

habilita i. generale
deshabilita INT 1
sin modificar

Etc: Exámenes

Examen Laboratorio 04

Volver a la Lista de Exámenes

Parte 1 de 5 - 1.0/ 2.0 Puntos

Preguntas 1 de 6 1.0/ 2.0 Puntos

Tenemos un interfaz de consola en la dirección 0xFFFF0100 con los siguientes registros:

Registro de órdenes y estado (Lectura/escritura. Dirección = Base).
- R (bit 2, lectura/escritura). Indicador de dispositivo preparado: el hardware hace R = 1 cuando la consola está disponible. Para cancelar, el programa ha de escribir R=0 .
- E: (bit 7, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo, línea INT1*).

Registro de datos (Sólo escritura. Dirección = Base + 4).
Ambos registros son de 8 bits.

El siguiente programa escribe el carácter de la posición de memoria "Letra" en la consola, sincronizando por consulta de estado (la interrupción estará deshabilitada). Completad las instrucciones que hay detrás del bucle de consulta de estado (que no se muestra) que deben cancelar el bit R y escribir el carácter en la consola.

```
        .data  0x1000000
Letra: .ascii 'p'

        la  $t0, 0xFFFF0100

... bucle de consulta de estado

        sb  Letra , 4($t0)      $zero  0($t0)

        lbu $t1, 4              Letra

        sb  $t1, 0($t0)         4($t0)
```

Preguntas 2 de 6

0.5/ 2.0 Puntos

Tenemos una interfaz de reloj en DB=0xFFFF0010 que tiene un único registro de Estado/Ordenes accesible para lectura y escritura. Su contenido es :

- R (bit 1, lectura/escritura). Indicador de dispositivo preparado: R = 1 a cada segundo.
  - Cancelación: se escribirá un 0 el en bit R .
- E: (bit 0, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo)

El dispositivo está conectado a la línea de interrupción INT2*.

Ordena las instrucciones para que el código realice un bucle de consulta de estado que espere hasta que haya pasado un segundo para (1) cancelar el bit R y (2) incrementar la variable *tiempo* de tipo word. Asumir que en $t0 está cargada la dirección base de la interfaz.

A.

```
sb $zero,0($t0)
```

B.

```
sw $t1,0($t0)
```

C.

```
addui $t1,$t1,1
```

D.

```
andi $t1,$t1,2
```

E.

```
la $t0,tiempo
```

F.

```
lw $t1,0($t0)
```

G.

```
lb $t1,0($t0)
```

H.

```
beqz $t1,bucle
```

1.

F 1 bucle:

D 2. 2

H 3. 3

G 4. 4

A 5. 5

E 6. 6

7.

B 7

C 8. 8

Parte 3 de 5 - **1.33/ 2.0 Puntos**

Preguntas 3 de 6 1.3333334/ 2.0 Puntos

Completar las siguientes instrucciones de un fragmento de código de inicialización que habilite las interrupciones generales, la INT3 y activa el modo usuario:

```
li $t0, 0x 0803
mtc0 $t0, $12
```

Completar las siguientes instrucciones de un fragmento de código que habilita las interrupciones generales, la INT1,INT3 y deshabilita la INT2 dejando el resto sin modificar:

```
mfc0 $t0, $12
ori $t0, $t0, 0x 0a00     0A01
andi $t0, $t0, 0x f4ff    FBFF
mtc0 $t0, $12
```

Parte 4 de 5 - **0.0/ 2.0 Puntos**

Preguntas 4 de 6 0.0/ 2.0 Puntos

La interfaz del reloj de PCSpim es la siguiente:

Dirección Base=0xFFFF0010, con un único registro de órdenes y estado. Interrumpe por la línea Int2

Registro de órdenes y estado (Lectura/escritura)

- R (bit 1, lectura/escritura). Indicador de dispositivo preparado: R = 1 a cada segundo.
  - Cancelación (R = 0): se escribirá en bit R un 0.
- E: (bit 0, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo)

Considere que el inicio del sistema ha habilitado completamente la interrupción del reloj y que el manejador contiene el siguiente tratamiento:

```
int2:   la $t0,0xFFFF0010
        li $t1,3
        sb $t1,0($t0)
        j retexc
```

Indique qué afirmaciones de las siguientes son correctas:

☐ A.

El tratamiento de la interrupción permite ofrecer servicios de espera a los procesos

☐ B. El manejador sólo atenderá la primera interrupción del reloj porque el tratamiento la inhabilita

☑ C. Al terminar el manejador, la interrupción del reloj continuará activa y el manejador volverá a ejecutarse

☐ D. El código cancela el bit R

Parte 5 de 5 - **2.0/ 2.0 Puntos**

Preguntas 5 de 6

1.0/ 1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué punto se utilizarían las siguientes acciones o instrucciones:

```
acceder al adaptador del dipositivo (consola en el caso de print_char)
terminar ejecutando: b retexc
```

- (●) A.Se hace en el código de la llamada al sistema (después de saltar a la etiqueta print_char)
- ( ) B.Se hace en el manejador de execepciones al inicio.
- ( ) C.Se hace en el manejador de excepciones después de comprobar que se trata de una llamada al sistema, pero antes de analizar $v0
- ( ) D.No se hace ya que el código no tiene nada que ver con el procesamiento de una llamada al sistema
- ( ) E.Se hace en la inicialización del sistema operativo

Preguntas 6 de 6

1.0/ 1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué punto se utilizarían las siguientes acciones o instrucciones:

```
mfc0 $k0, $14 # EPC
addi $k0,$k0,4
sw $k0, dirret
```

- ( ) A.Se hace en el código de la llamada al sistema (después de saltar a la etiqueta print_char)
- ( ) B.No se hace ya que el código no tiene nada que ver con el procesamiento de una llamada al sistema
- ( ) C.Se hace en la inicialización del sistema operativo
- ( ) D.Se hace en el manejador de execepciones al inicio.
- (●) E.Se hace en el manejador de excepciones después de comprobar que se trata de una llamada al sistema, pero antes de analizar $v0

Preguntas 1 de 6 2.0 Puntos

Tenemos una interfaz de consola en la dirección 0xFFFF2000 con los siguientes registros:

Registro de órdenes y estado (Lectura/escritura. Dirección = Base).
• R (bit 4, sólo lectura). Indicador de dispositivo preparado: R = 1 cuando la consola está disponible.
• E: (bit 7, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo, línea INT1*).
• C: (bit 2, escritura). Cancela la interrupción al escribir un '1', y R vuelve a valer 0 y se desactiva la línea de interrupción si la hubiera.
Registro de datos (Sólo escritura. Dirección = Base + 4).
Ambos registros son de 8 bits.

El siguiente programa escribe el carácter de la posición de memoria "Valor" en la consola, sincronizando por consulta de estado (la interrupción estará deshabilitada). Completar las instrucciones que hay detrás del bucle de consulta de estado (el cual no se muestra) que primero deben cancelar el bit R y luego escribir el valor en la consola.

```
        .data  0x1000000
Valor: .ascii 'p'

        la  $t0, 0xFFFF2000
... bucle de consulta de estado
        li $t1, 0x [01 ~~01~~ 04]
        sb $t1, [0($t0)]
        lbu $t1, [valor]
        sb  $t1, [4($t0)]
```

Parte 2 de 5 -

Preguntas 2 de 6 2.0 Puntos

Tenemos una interfaz de teclado en DB=0xFFFF0000 que tiene un registro de 8 bits de Estado/Ordenes, de Lectura/escritura con Dirección = DB, con los siguientes bits:

• R (bit 3, sólo lectura). Indicador de dispositivo preparado: R = 1 cada vez que se pulsa una tecla. Para hacer R = 0 es necesario realizar un acceso de lectura en el registro de datos.

• E: (bit 0, lectura/escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo). Línea INT3*.

Y un registro de datos (Sólo lectura. Dirección = DB + 1).

• COD (bits 7...0). Código ASCII de la tecla pulsada. Leer de este registro provoca que R = 0.

Seleccionar las instrucciones que usarías indicando el orden para realizar un bucle de consulta de estado que espere hasta que se haya pulsado una tecla y posteriormente borre el bit R. Asumir que en $t1 hemos cargado la dirección base.
NOTA: el programa sólo tiene 4 instrucciones, hay 3 que no son correctas (un branch, un and y una load o store)

A.

```
sb $zero, 0($t1)
```

B.

```
beqz $t0, espera
```

C.

```
lbu $t0, 0($t1)
```

D.

```
lbu $t0, 1($t1)
```

E.

```
andi $t0, $t0, 0x01
```

F.

```
andi $t0, $t0, 0x08
```

G.

```
bnez $t0, espera
```

---

1. C

```
(1) espera:
```

2. F

```
(2)
```

3. B

```
(3)
```

4. D

```
(4)
```

5. G

```
(branch incorrecto)
```

6. E

```
(and incorrecto)
```

7. A

(load o store incorrecto)

Parte 3 de 5 -

Preguntas 3 de 6 2.0 Puntos

Completar las siguientes instrucciones de un fragmento de código de inicialización que habilite las interrupciones generales, la INT5 y activa el modo usuario:

```
li $t0, 0x 2003
mtc0 $t0, $12
```

Completar las siguientes instrucciones de un fragmento de código que habilita las interrupciones generales, la INT0,INT3 y deshabilita la INT1 dejando el resto sin modificar:

```
mfc0 $t0, $12
ori $t0, $t0, 0x 0902 0901
andi $t0, $t0, 0x FDFF
mtc0 $t0, $12
```

Parte 4 de 5 -

Preguntas 4 de 6 2.0 Puntos

La interfaz de teclado del simulador PCSPIM contiene dos registros de un byte ubicados a partir de la direción base DB = 0xFFFF0000 y tiene la posibilidad de interrumpir por la linea irq0*. La descripción de los registros es la siguiente:

Registro de Estado/Órdenes (lectura/escritura, dirección = DB)

- R (bit 0, sólo lectura). Indicador de dispositivo preparado. R=1 cada vez que se pulsa una tecla.
  - Cancelación: Para hacer que R=0, es necesario leer el registro de datos.
- E (bit 1, lectura/escritura). Habilita la petición de interrupciones del dispositivo. Mientras E=1, se activará irq0* cada vez que R=1.

Registro de Datos (Sólo lectura, dirección = DB + 4)

- COD (bits 0..7). Código ASCII de la última tecla pulsada. Leer de este registro provoca la puesta a cero del bit R.

Supón que, en el arranque del sistema, se ha habilitado completamente la interrupción del teclado. Considere la siguiente rutina de servicio de la interrupción 0:

```
irq0:  la $t0,0xFFFF0000
       lb $t1,0($t0)
       andi $t1,$t1,2
       sb $t1,0($t0)
       b retexc
```

Indique qué afirmaciones de las siguientes son correctas sobre esta rutina de servicio:

- [x] A. No cancela correctamente la petición de interrupción.
- [ ] B. El carácter leido se almacena en el registro $t1.
- [ ] C. Accede al teclado por consulta de estado.
- [ ] D. Cancela correctamente la petición de interrupción.

Parte 5 de 5 -

Preguntas 5 de 6 1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué punto se utilizarían las siguientes acciones o instrucciones:

```
 li $v0,0x12
li $a0,0x41 # código del carácter A
syscall
```

- (•) A.Se utilizan en el código del proceso usuario cuando hace una llamada al sistema
- ( ) B.Se hace en el manejador de excepciones después de comprobar que se trata de una llamada al sistema, pero antes de analizar $v0
- ( ) C.Se hace en el código de la llamada al sistema (después de saltar a la etiqueta print_char)
- ( ) D.Se hace en la inicialización del sistema operativo
- ( ) E.Se hace en el manejador de execepciones al inicio.

Borra selección

Preguntas 6 de 6 1.0 Puntos

En el procesamiento de una llamada al sistema (por ejemplo print_char) indica en qué

punto se utilizarían las siguientes acciones o instrucciones:

```
mfc0 $k0, $14 # EPC
addi $k0,$k0,4
sw $k0, dirret
```

- A.Se hace en el código de la llamada al sistema (después de saltar a la etiqueta print_char)
- B.Se hace en el manejador de excepciones después de comprobar que se trata de una llamada al sistema, pero antes de analizar $v0
- C.Se hace en la inicialización del sistema operativo
- D.Se utilizan en el código del proceso usuario cuando hace una llamada al sistema
- E.Se hace en el manejador de excepciones al inicio.

Borra selección