Estructura de Computadores

Tema 8. Excepciones e Interrupciones

Ejemplo de excepción: fallo de TLB

- Fallo de TLB
  - El procesador accede a una dirección virtual para la cual el TLB no contiene la traducción

Ejemplo de excepción: fallo de TLB

- Fallo de TLB
  - El procesador accede a una dirección virtual para la cual el TLB no contiene la traducción
- Dos tipos de excepciones:
  - ExcCause = TLBL: lectura de instrucción o load
  - ExcCause = TLBS: store

Ejemplo de excepción: fallo de TLB

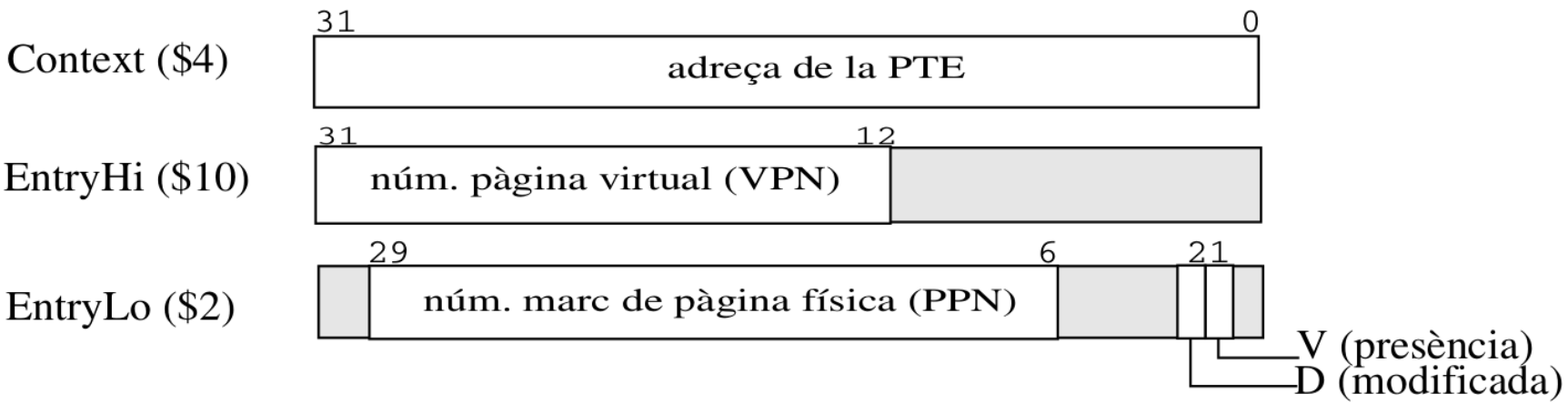
- Fallo de TLB
  - El procesador accede a una dirección virtual para la cual el TLB no contiene la traducción
- Dos tipos de excepciones:
  - ExcCause = TLBL: lectura de instrucción o load
  - ExcCause = TLBS: store
- Tratamiento del fallo de TLB
  - Copiar la entrada correspondiente de la tabla de páginas en el TLB: 24 bits de PPN, bit de presencia (V) y bit de dirty (D)
  - Escribir en la entrada del TLB el VPN (número de página virtual)

Fallo de TLB en MIPS

- En MIPS los fallos de TLB producen una excepción y se gestionan por software (Sistema Operativo)
- Los fallos de TLB son frecuentes y conviene tratarlos de forma eficiente
  - La RSE genérica (dirección 0x80000180) es demasiado compleja y lenta
  - Se utiliza la rutina TLBmiss (dirección 0x80000000)
    - Muy rápida (normalmente menos de 13 ciclos)
    - No salva los registros en la pila, solo modifica $k1 que está reservado al Sistema Operativo

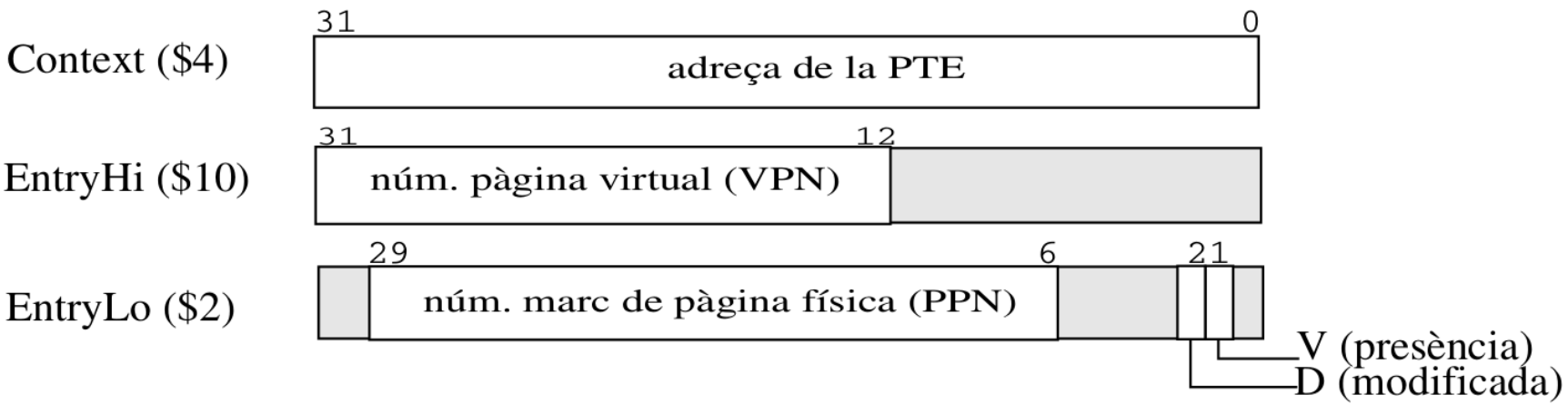
Fallo de TLB en MIPS

- En MIPS el TLB forma parte del coprocesador CP0 y se gestiona mediante los siguientes registros:



Fallo de TLB - Acciones hardware

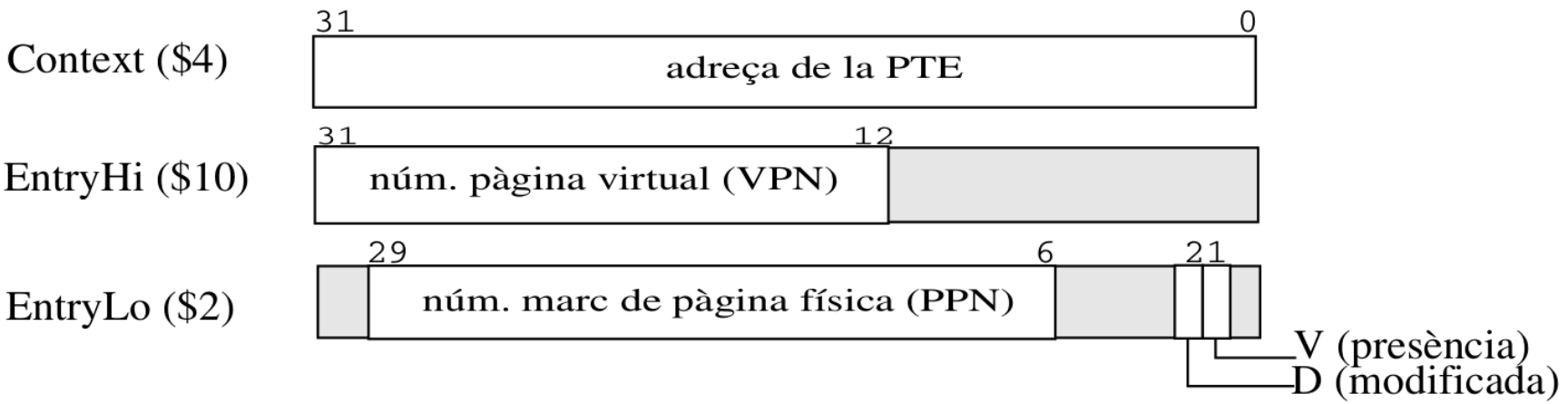
- Al detectar un fallo de TLB, el hardware realiza las siguientes acciones:
  - Escribe la dirección de la entrada de la tabla de páginas (PTE) en el registro Context
  - Escribe los 20 bits del VPN en el registro EntryHi



Fallo de TLB - Acciones software

- Ejemplo de TLBmiss:

```
      .ktext 0x80000000
TLBmiss:
      mfc0   $k1, $4        # Copia Context (adreça de la PTE) en $k1
      lw     $k1, 0($k1)    # Llegeix la PTE (traduccció) en $k1
      mtc0   $k1, $2        # Copia la traducció a EntryLo
      tlbwr                 # Escriu EntryHi|EntryLo al TLB
      eret                  # Retorna al programa
```



- Instrucción **tlbwr**: la entrada a reemplazar se selecciona de forma pseudoaleatoria (random)

Bit V del TLB

- La rutina TLBmiss no comprueba el bit de presencia (V)
- Al terminar TLBmiss, se salta a la instrucción que produjo el fallo de TLB
- Al reejecutar la instrucción, se producirá un acierto de TLB y se comprobará el bit de presencia (V)
  - Si V=1, la instrucción se ejecuta con normalidad
  - Si V=0, se produce una excepción por fallo de página y se invoca la RSE genérica

Bit V del TLB (Recordatorio)

- El bit V del TLB es una copia del bit de presencia de la tabla de páginas
- Si V=1 la página se encuentra en MP en un marco de página, si V=0 la página solo está en disco
- El acierto o fallo de TLB no depende del bit V, solo depende de que el VPN se encuentre en el TLB
- El bit V se utiliza en caso de reemplazo
  - El algortimo de reemplazo seleccionará preferentemente entradas del TLB con bit V=0

Bit D del TLB (Recordatorio)

- El bit D (dirty) indica si la página en memoria física ha sido modificada respecto a la copia en disco
- El bit D es el único bit del TLB modificado por la ejecución del programa
  - Se pone a 1 la primera vez que el programa ejecuta un *store* sobre la página
  - Se utiliza una política de escritura inmediata para actualizar el bit D en la entrada de la tabla de páginas

Bit D del TLB

- Cuando se produce un fallo de página:
  - La RSE copia la página del disco a memoria física y actualiza las entradas correspondientes de la tabla de páginas y del TLB
  - Si el acceso es una escritura el bit D se pone a 1, mientras que si es una lectura se pone a 0

Bit D del TLB

- Cuando se produce un fallo de página:
  - La RSE copia la página del disco a memoria física y actualiza las entradas correspondientes de la tabla de páginas y del TLB
  - Si el acceso es una escritura el bit D se pone a 1, mientras que si es una lectura se pone a 0
- Cuando una escritura produce una acierto de TLB:
  - Se comprueba el bit D. Si vale 0 es la primera escritura, en cuyo caso se produce una excepción de "página modificada". La RSE pondrá el bit D a 1 en el TLB y la tabla de páginas.

Bit D del TLB

- Cuando se produce un fallo de página:
  - La RSE copia la página del disco a memoria física y actualiza las entradas correspondientes de la tabla de páginas y del TLB
  - Si el acceso es una escritura el bit D se pone a 1, mientras que si es una lectura se pone a 0
- Cuando una escritura produce una acierto de TLB:
  - Se comprueba el bit D. Si vale 0 es la primera escritura, en cuyo caso se produce una excepción de "página modificada". La RSE pondrá el bit D a 1 en el TLB y la tabla de páginas.
- Como se utiliza escritura inmediata, al reemplazar una entrada del TLB se puede sobreescribir directamente

Llamadas al sistema

- El Sistema Operativo (SO) proporciona acceso seguro y eficiente a los recursos compartidos
  - Memoria física
  - Dispositivos de E/S
  - Procesador (ejecución concurrente de múltiples programas)

Llamadas al sistema

- El Sistema Operativo (SO) proporciona acceso seguro y eficiente a los recursos compartidos
  - Memoria física
  - Dispositivos de E/S
  - Procesador (ejecución concurrente de múltiples programas)
- La gestión de dichos recursos se realiza en modo sistema
  - EXL=1 en el registro Status

Llamadas al sistema

- El Sistema Operativo (SO) proporciona acceso seguro y eficiente a los recursos compartidos
  - Memoria física
  - Dispositivos de E/S
  - Procesador (ejecución concurrente de múltiples programas)
- La gestión de dichos recursos se realiza en modo sistema
  - EXL=1 en el registro Status
  - Permite acceder a código/datos del SO en la región de memoria protegida (direcciones virtuales con bit 31 a 1).
    - Excepción AdEL (lectura) o AdES (escritura) al acceder a la región protegida en modo usuario

Llamadas al sistema

- El Sistema Operativo (SO) proporciona acceso seguro y eficiente a los recursos compartidos
  - Memoria física
  - Dispositivos de E/S
  - Procesador (ejecución concurrente de múltiples programas)
- La gestión de dichos recursos se realiza en modo sistema
  - EXL=1 en el registro Status
  - Permite acceder a código/datos del SO en la región de memoria protegida (direcciones virtuales con bit 31 a 1).
    - Excepción AdEL (lectura) o AdES (escritura) al acceder a la región protegida en modo usuario
  - Permite ejecutar instrucciones que operan sobre el CP0
    - Excepción CpU al ejecutar una instrucción privilegiada (CP0) en modo usuario

Llamadas al sistema

- El acceso a los servicios del SO se realiza mediante una excepción

Llamadas al sistema

- El acceso a los servicios del SO se realiza mediante una excepción
- En MIPS se utiliza la instrucción syscall
  - Cambia a modo sistema (EXL=1)
  - Salta a la RSE, que forma parte del SO

Llamadas al sistema

- El acceso a los servicios del SO se realiza mediante una excepción
- En MIPS se utiliza la instrucción syscall
  - Cambia a modo sistema (EXL=1)
  - Salta a la RSE, que forma parte del SO
- Funcionan de forma similar a una llamada a subrutina
  - El código del servicio se pasa en $v0
    - El SO tiene una subrutina para cada servicio

Llamadas al sistema

- El acceso a los servicios del SO se realiza mediante una excepción
- En MIPS se utiliza la instrucción syscall
  - Cambia a modo sistema (EXL=1)
  - Salta a la RSE, que forma parte del SO
- Funcionan de forma similar a una llamada a subrutina
  - El código del servicio se pasa en $v0
    - El SO tiene una subrutina para cada servicio
  - Los parámetros se pasan en $a0-$a3 ($f12 para coma flotante)

Llamadas al sistema

- El acceso a los servicios del SO se realiza mediante una excepción
- En MIPS se utiliza la instrucción syscall
  - Cambia a modo sistema (EXL=1)
  - Salta a la RSE, que forma parte del SO
- Funcionan de forma similar a una llamada a subrutina
  - El código del servicio se pasa en $v0
    - El SO tiene una subrutina para cada servicio
  - Los parámetros se pasan en $a0-$a3 ($f12 para coma flotante)
  - Se invoca el SO mediante syscall (similar a jal)

Llamadas al sistema

- El acceso a los servicios del SO se realiza mediante una excepción
- En MIPS se utiliza la instrucción syscall
  - Cambia a modo sistema (EXL=1)
  - Salta a la RSE, que forma parte del SO
- Funcionan de forma similar a una llamada a subrutina
  - El código del servicio se pasa en $v0
    - El SO tiene una subrutina para cada servicio
  - Los parámetros se pasan en $a0-$a3 ($f12 para coma flotante)
  - Se invoca el SO mediante syscall (similar a jal)
  - El valor de retorno (si existe) se pasa en $v0 ($f0 para coma flotante)

Llamadas al sistema

- Códigos de servicios del SO

Servei	Codi	Paràmetres	Resultat
print_int	1	`$a0 = int`	
print_float	2	`$f12 = float`	
print_double	3	`$f12 = double`	
print_string	4	`$a0 = string`	
read_int	5		$v0 = integer
read_float	6		$f0 = float
read_double	7		$f0 = double
read_string	8	`$a0 = buffer, $a1 = length`	
sbrk	9	`$a0 = amount`	$v0 = address
exit	10		

Llamadas al sistema

- Ejemplos de llamadas al sistema

```
        .data
cadena:     .asciiz "Introdueix una frase\n"
        .text
        ...
        li      $v0, 4          # crida num 4: print_string(char *p)
        la      $a0, cadena
        syscall

        li      $v0, 10         # crida num 10: exit()
        syscall
```

Interrupciones desde dispositivos de E/S

- El SO proporciona acceso a los dispositivos de E/S
- Un programa de usuario puede acceder a un dispositivo de E/S mediante una llamada al sistema
- El dispositivo de E/S puede iniciar la comunicación mediante una señal de **interrupción**
  - Cuando la CPU detecta la interrupción se invoca a la RSE

Sincronización con los dispositivos de E/S

- La comunicación con los dispositivos de E/S está dominada por la necesidad de tolerar las enormes latencias de las operaciones

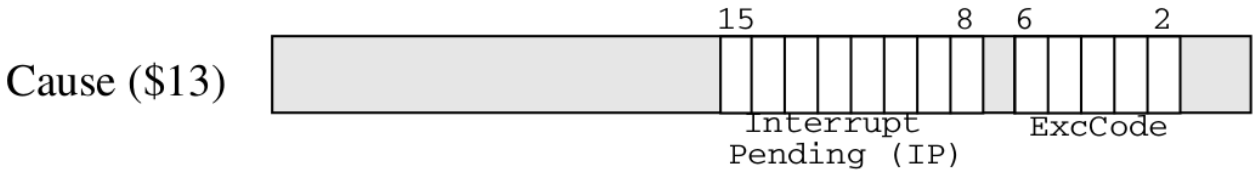
Sincronización con los dispositivos de E/S

- La comunicación con los dispositivos de E/S está dominada por la necesidad de tolerar las enormes latencias de las operaciones
- Dos tipos de sincronización
  - **Sincronización por encuesta**: El progama espera a que la operación termine, consultando repetidamente el estado de la operación

Sincronización con los dispositivos de E/S

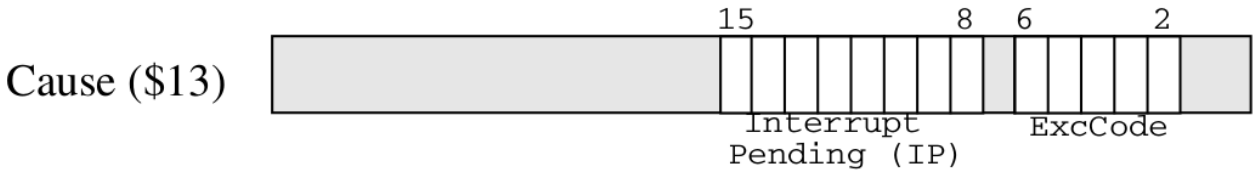
- La comunicación con los dispositivos de E/S está dominada por la necesidad de tolerar las enormes latencias de las operaciones
- Dos tipos de sincronización
  - **Sincronización por encuesta**: El progama espera a que la operación termine, consultando repetidamente el estado de la operación
  - **Sincronización por interrupciones**: El SO planifica y ejecuta otros programas mientras se realiza la operación. Cuando el dispositivo de E/S está listo, envía una señal de interrupción

Interrupt Pending del registro Cause



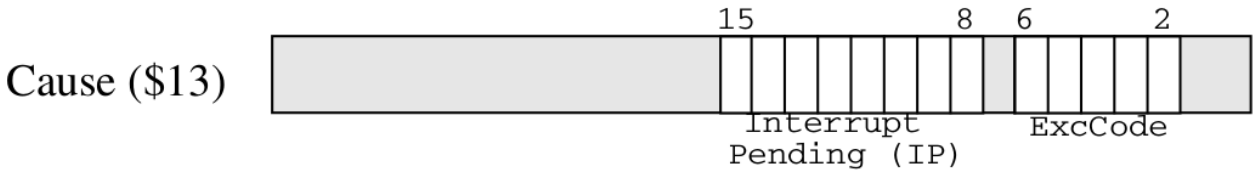
- Cada dispositivo de E/S está asociado con un bit del campo Interrupt Pending (IP) en el registro Cause del CP0
  - El bit se activa cuando el dispositivo quiere notificar que ha finalizado una operación o que necesita atención
  - El bit se desactiva cuando el procesador atiende la interrupción o cuando ha pasado un determinado tiempo de espera

Interrupt Pending del registro Cause



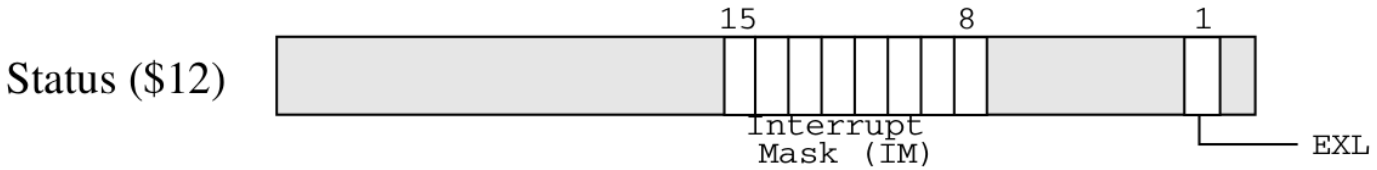
- Cada dispositivo de E/S está asociado con un bit del campo Interrupt Pending (IP) en el registro Cause del CP0
  - El bit se activa cuando el dispositivo quiere notificar que ha finalizado una operación o que necesita atención
  - El bit se desactiva cuando el procesador atiende la interrupción o cuando ha pasado un determinado tiempo de espera
- El campo IP guarda las peticiones de interrupción que han llegado y están pendientes de ser atendidas

Interrupt Pending del registro Cause



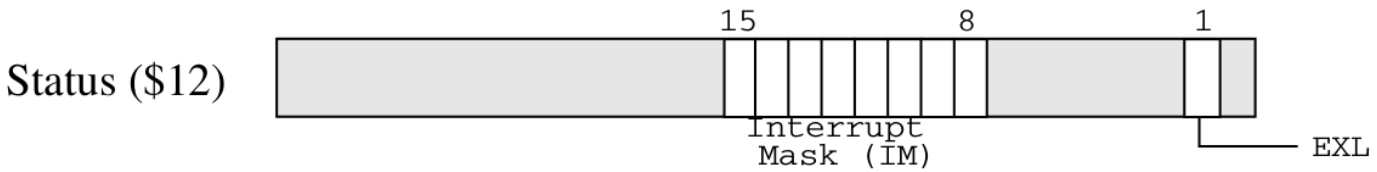
- Cada dispositivo de E/S está asociado con un bit del campo Interrupt Pending (IP) en el registro Cause del CP0
  - El bit se activa cuando el dispositivo quiere notificar que ha finalizado una operación o que necesita atención
  - El bit se desactiva cuando el procesador atiende la interrupción o cuando ha pasado un determinado tiempo de espera
- El campo IP guarda las peticiones de interrupción que han llegado y están pendientes de ser atendidas
- NO se interrumpe la instrucción el curso
  - Cuando finaliza la ejecución de la instrucción el procesador comprueba el campo IP y genera una excepción si hay algún bit a 1 (ExcCode = Int)

Interrupt Mask del registro Status



- Las interrupciones solo se atienden en modo usuario (bit EXL=0)
  - En modo sistema (bit EXL=1) las interrupciones están deshabilitadas

Interrupt Mask del registro Status



- Las interrupciones solo se atienden en modo usuario (bit EXL=0)
  - En modo sistema (bit EXL=1) las interrupciones están deshabilitadas
- El SO puede deshabilitar las interrupciones de los dispositivos de E/S de forma selectiva
  - Usando el campo Interrupt Mask (IM) del registro Status
  - Si el dispositivo i realiza una petición de interrupción, solo se genera una excepción si $IM_i = 1$ y EXL=0

Identificación del dispositivo por software

- La RSE comprueba los campos Interrupt Pending e Interrupt Mask para determinar el dispositivo que ha realizado la petición de interrupción
  - En caso de múltiples peticiones simultáneas se establece un sistema de prioridades

Identificación del dispositivo por software

- La RSE comprueba los campos Interrupt Pending e Interrupt Mask para determinar el dispositivo que ha realizado la petición de interrupción
  - En caso de múltiples peticiones simultáneas se establece un sistema de prioridades
- Puede pasar que no haya ninguna interrupción pendiente si el dispositivo de E/S no ha mantenido la petición activa el tiempo suficiente
  - La RSE ha de salvar todos los registros en la pila: puede pasar mucho tiempo hasta que se comprueba el campo Interrupt Pending

	Afirmació	V	F
1.-	Si l'accés a dades d'un *store* produeix un encert al TLB, però el bit D val 0, llavors es produeix una excepció		
2.-	Si al traduir una adreça amb el TLB es troba una entrada amb el mateix VPN però amb el bit V que val 0, llavors es produeix una fallada de pàgina		
3.-	Si el bit EXL val 1, les excepcions seran ignorades		
4.-	Si s'executa la instrucció tlbwr en mode usuari, es produeix una excepció		
5.-	Després d'una fallada de TLB la instrucció causant de l'excepció s'ha de reexecutar		
6.-	Una rutina de servei a excepcions pot acabar amb `jr $epc` doncs al registre `$epc` es guarda l'adreça de la instrucció que cal seguir executant un cop s'ha atès l'excepció		
7.-	Segons la representació de números en coma flotant IEEE-754 amb simple precisió el valor més gran representable just per sota del +infinit es representa `0x7f7fffff`		
8.-	El bit D del TLB en el processador MIPS estudiat al tema 8 és sempre coherent amb el bit D de la TP.		
9.-	Una excepció no pot ser atesa fins que la instrucció en curs hagi finalitzat		
10.-	En el MIPS, una mateixa instrucció pot causar durant la seva execució 2 fallades de pàgina		

Donat el següent codi en C, escriu al fitxer **torn1-exdat.s** el programa equivalent en assemblador de MIPS.

```
int A[4][4] = {  0,  1,  2,  3,
                 4,  5, -6, -1,
                 7,  8, -1, -1,
                 9, -1, -1, -1 };


main()
{
  int i, j;

  for (i= 0; i< 4; i++)
    for (j= 0; j< 4-i; j++)
    {
      if (A[i][j] > 0)
        A[j][i] = A[i][j];
      else
        A[j][i] = -3*A[i][j];
    }
}
```

Comprova que en acabar d'executar el programa, la matriu global `A` val:

```
int A[4][4] = {  0,  1,  2   3,
                 1,  5, 18, -1,
                 2, 18, -1, -1,
                 3, -1, -1, -1 };
```

Donat el següent codi escrit en C:

```
int a[12] = {1024, 1024, 1024, 1024,
             1024,    0,    0,  512,
             1024, 1024, 1024, 1024};

int jacobi(int *ptr, int disp) {
    return (ptr[disp+1] + ptr[disp-1] + ptr[disp-4] + ptr[disp+4]) / 4;
}

void copy(int *dst, int *src, int n) {
    int i;

    for (i=0; i<n; i++)
        dst[i] = src[i];
}
```

```
void main() {
    int tmp[12], pos=5;

    copy(tmp, a, 12);

    tmp[5] = jacobi(a, pos);
    pos++;
    tmp[6] = jacobi(a, pos);

    copy(a, tmp, 12);
}
```

Tradueix al fitxer **torn1-exsub.s** la funció `main` (en el requadre). El codi de la subrutina jacobi ja està programat i no es pot modificar. Comprova que al final de l'execució del programa la matriu global "a" té els següents valors:

```
int a[12] = { 1024 1024 1024 1024
              1024  768  640  512
              1024 1024 1024 1024 };
```

Considera un sistema computador format per un processador MIPS, una memòria principal (MP) i una memòria cache de dades (MC), amb la següent configuració:

- Correspondència directa
- Mida bloc 32 bytes (8 words)
- Capacitat total 128 bytes (4 blocs)
- Escriptura immediata amb assignació (és la que implementa el MARS)

Partim del seguent programa en alt nivell (podeu trobar el programa equivalent en MIPS al fitxer **torn1-programa-test.s,** el qual **NO** s'ha de lliurar):

```
int X[8];
int Y[4][8];

main() {
    int var = 0, i, j;
    for (i = 0; i < 4; i++) {
        var++;
        for (j = 0; j < 8; j++)
            Y[i][j] = X[j]+var;
    }
}
```

Contesta les següents preguntes al fitxer de respostes **torn1-excache.txt** (que **SÍ** has de lliurar):

a) Quants encerts i fallades es produeixen en els accessos a X i Y?

b) Suposant la configuració inicial, sense variar la mida de bloc, què passaria amb una cache amb el doble de capacitat (256 bytes)? Justifica breument la resposta.

c) Suposant la configuració inicial, sense variar el nombre total de blocs ni la mida de bloc, què passaria si en comptes de correspondència directa és una cache amb correspondència associativa per conjunts de 2 vies? Justifica breument la resposta.