UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Arquitectura de Computadors

Estructura de Computadores

Tema 8: Excepciones e Interrupciones

Agustín Fernández

Departament d'Arquitectura de Computadors

Facultat d'Informàtica de Barcelona

Universitat Politècnica de Catalunya

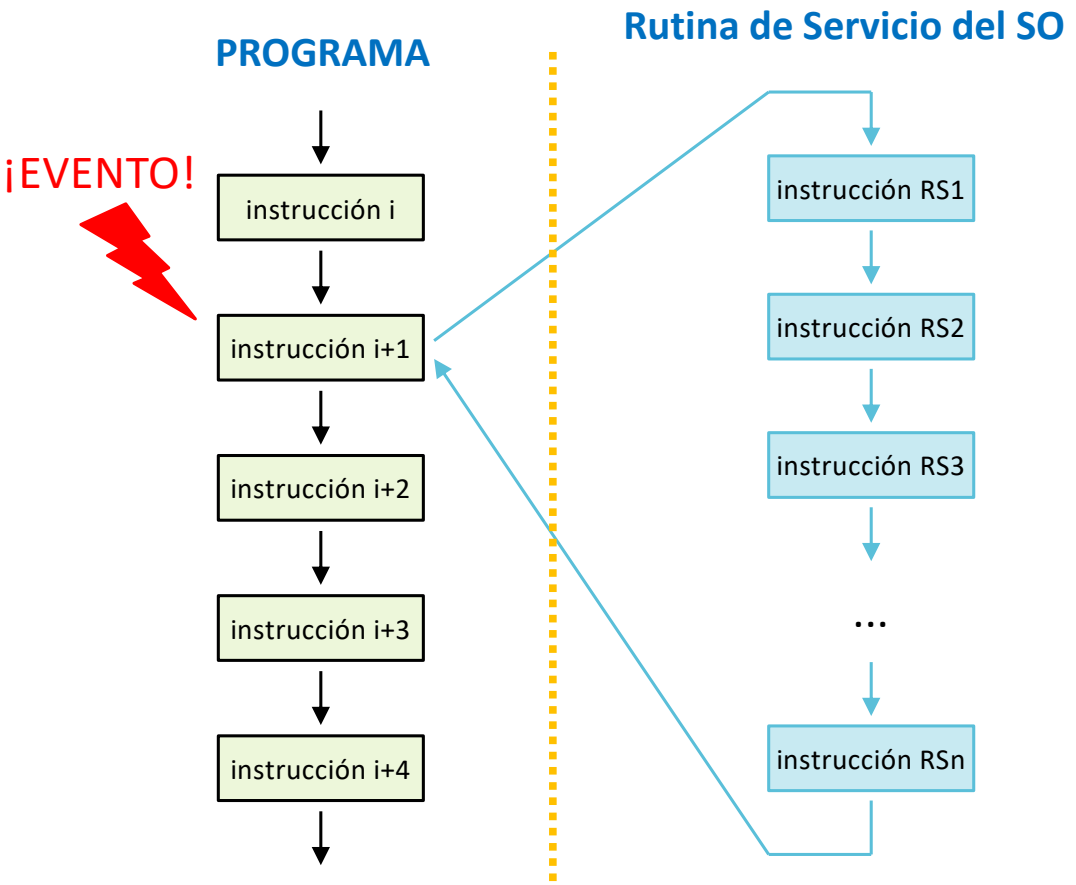
Introducción

- La instrucción **add** de MIPS provoca una excepción si se produce un overflow.
- Un fallo de página provoca una excepción.
- Un fallo de TLB provoca una excepción
- Una división por 0 provoca una excepción.
- Un acceso a memoria con una dirección NO ALINEADA provoca una excepción

¿Qué es una EXCEPCIÓN?

Introducción

- Una excepción (o interrupción) es un mecanismo que altera el flujo de ejecución de un programa sin que intervenga ninguna instrucción de salto
  - Provocado por un evento inesperado o raro (externo o interno)
  - Necesita ser tratado por una **Rutina de Servicio del Sistema Operativo**



Tipología de excepciones e interrupciones

- **EXCEPCIONES** (interrupciones síncronas), provocadas por la ejecución de una instrucción.
  - Excepciones por violación de una restricción.
    - ✓ Intento de ejecutar una instrucción privilegiada
    - ✓ Intento de ejecutar una instrucción inválida
    - ✓ Overflow aritmético, división por cero, acceso a una dirección no alineada
  - Excepciones relacionadas con la Memoria Virtual
    - ✓ Fallo de página, Fallo de TLB, violación de protecciones
  - Causadas expresamente por el programador
    - ✓ Llamada al Sistema Operativo (invocadas con un syscall)
    - ✓ Relacionadas con debug: traps, breakpoint, ...

Tipología de excepciones e interrupciones

- **INTERRUPCIONES** (interrupciones asíncronas), provocadas por un evento externo.
  - Peticiones de servicio de un dispositivo de E/S.
    - ✓ Alguien ha pulsado una tecla
    - ✓ Ha acabado una operación de E/S programada
    - ✓ Ha llegado un paquete de datos por la red
  - Temporizador
  - Errores Hardware

Rutina de Servicio de Excepciones (RSE)

- **RSE** (exception handler)
  - Código del Sistema Operativo que se encarga de gestionar las excepciones e interrupciones.

- Al detectar una excepción o interrupción
  1. Se finaliza o cancela la instrucción en curso
  2. Se salta a la RSE
     - ✓ Se escribe en el PC la dirección inicial de la RSE
  3. Se ejecutan las instrucciones de la RSE
  4. Se aborta o se reanuda la ejecución del programa interrumpido

Cronología de una excepción

1. **Detección de la excepción o interrupción**
   - **Decodificando la instrucción en curso**: código operación ilegal, instrucción privilegiada, syscall
   - **En la MMU**: fallo de página, fallo de TLB
   - **En la ALU**: dirección no alineada, overflow, división por cero
   - **Un dispositivo de E/S**: ha activado una señal externa INT (interrupción)
2. **La Unidad de Control de la CPU decide**
   - Si es una excepción, aborta la instrucción en curso. Inhibe la escritura en registros, memoria, etc.
   - En caso de interrupción, espera a que acabe la instrucción en curso.
3. **Salta a la RSE (Rutina de Servicio a Excepciones o Exception Handler)**
   - Es el código de SO que hace el tratamiento de excepciones e interrupciones.
   - ¿Cómo se salta? ⇒ Poniendo en el PC la dirección inicial de la RSE
   - Hay que guardar la @ de la instrucción interrumpida
4. **Al acabar la RSE**
   - Puede retornar al programa o abortarlo, dependiendo del tipo de excepción

Tipos de Excepciones e Interrupciones

Excepciones por la violación de una restricción

- Excepciones producidas al ejecutar una instrucción que viola una restricción:
  - Overflow aritmético
  - Acceso a Memoria NO alineado
  - División por cero
  - Instrucción reservada
- La instrucción se interrumpe sin escribir en registros o memoria
- Normalmente es un error y la RSE aborta la ejecución del programa
- En algunos casos la RSE puede resolver el problema
  - Por ejemplo, si el SO emula por software una instrucción, al acabar la emulación, se da por ejecutada la instrucción y se salta a la siguiente instrucción a la que ha provocado la excepción

Tipos de Excepciones e Interrupciones

Excepciones relacionadas con la Memoria Virtual

- Se producen durante la traducción de la @lógica a @física
- Fallo de TLB
  - En MIPS, se resuelve por software (lo normal es por hardware)
  - la RSE, copia la entrada de la TP en una entrada libre del TLB
- Fallo de página
  - La RSE copia la página de disco a un frame libre de la MP, actualiza TP y TLB
  - Si es necesario se reescribe la página (sucia) reemplazada en disco
- En ambos casos, al acabar la RSE se salta a la instrucción que ha provocado la excepción
  - Se ha de ejecutar de nuevo.
- Violación de protección
  - La RSE aborta la ejecución del programa

Tipos de Excepciones e Interrupciones

Excepciones de trap y llamadas al Sistema Operativo

- Las excepciones de trap se usan al depurar un programa
  - Ejecución instrucción a instrucción, entre medias entra el debugger
  - Uso de breakpoints
- Se utiliza una instrucción especial del ISA para acceder a un servicio del Sistema Operativo.
  - Instrucción **`syscall`** en MIPS
  - Es similar a una llamada a una subrutina
    - ✓ Con paso de parámetros y retorno de resultados
  - Al finalizar la RSE, continúa ejecutando el programa a partir de la instrucción siguiente a la que produjo la excepción (**`syscall`**)

Tipos de Excepciones e Interrupciones

Interrupciones

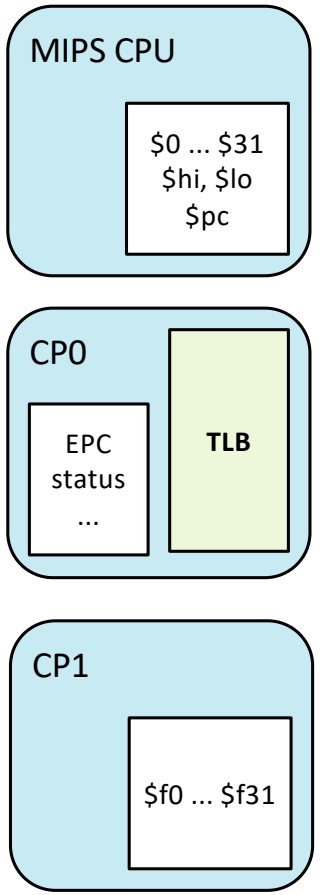
- Causadas por la activación de la señal INT (petición de interrupción) desde algún dispositivo de Entrada/Salida
- En este caso, la petición se atiende al final de la ejecución de la instrucción en curso.
  - La petición queda registrada en un registro de peticiones pendientes
  - Cuando se atiende, la RSE puede transferir datos entre la CPU y el dispositivo
  - Las peticiones pueden necesitar mucho tiempo, el SO puede planificar otro proceso mientras se termina la operación de E/S.
- Al acabar la RSE
  - el programa continúa la ejecución a partir de la siguiente instrucción

Implementación en MIPS

- El coprocesador de Sistema CP0
- Acciones del hardware en una excepción
- Acciones del software en una excepción

El coprocesador de Sistema CP0

- Controla el tratamiento de excepciones y la traducción con el TLB
- Añade instrucciones adicionales
  - **Instrucciones Privilegiadas**, sólo se pueden ejecutar en modo sistema
- Tiene un banco de registros específicos, que sólo son accesibles por las instrucciones privilegiadas.
  - Por ejemplo, para copiarlos de CP0 a CPU o viceversa:
    - ✓ `mfc0 rt,rd_c0  # rt ← rd_c0`
    - ✓ `mtc0 rt,rd_c0  # rd_c0 ← rt`



El coprocesador de Sistema CP0

EPC ($14) | Dirección de la instrucción interrumpida

- **Registro EPC (Exception Program Counter, $14)**
  - Registro de Lectura / Escritura
  - Antes de invocar la RSE, el procesador guarda el valor actual del PC en EPC
  - Es la **dirección de retorno** de la RSE
- ¿Dónde apunta el PC que se guarda en EPC?
  - **En una excepción**, teniendo en cuenta que la instrucción en curso se aborta:
    - ✓ ⇒ El PC no se incrementa, guardamos la dirección de la instrucción en ejecución.
  - **En una interrupción**, teniendo en cuenta que la instrucción en curso se acaba:
    - ✓ ⇒ El PC se incrementa, y guardamos la dirección de la siguiente instrucción

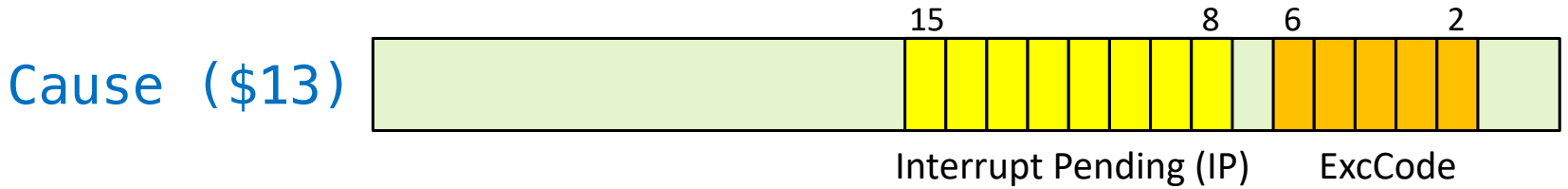
¿Porqué no se puede guardar la dirección de retorno en $ra?

El coprocesador de Sistema CP0

Registro Status ($12)

- **Bit EXL (Exception Level)**, tiene 2 funciones:
  - **Modo Usuario / Modo Sistema**
    - ✓ Si EXL = 0, el procesador está en modo usuario
    - ✓ Si EXL = 1, el procesador está en modo sistema, tiene acceso al espacio de direcciones de SO y puede ejecutar instrucciones privilegiadas.
  - Si EXL=1, **todas las interrupciones están inhibidas** y se ignoran (pero no las excepciones).
- **8 bits de IM (Interrupt Mask)**
  - 1 bit para cada tipo de interrupción
  - Si el bit vale 1, las interrupciones de ese tipo están habilitadas.

El coprocesador de Sistema CP0



Registro Cause ($13), sólo lectura

- **5 bits de ExcCode,** codifica la causa de la excepción
- **8 bits de IP (Interrupt Pending)**
  - Un bit para cada tipo de interrupción
  - Cuando se recibe una petición de interrupción, se activa el bit correspondiente
  - El bit permanece a 1 mientras el dispositivo mantenga la señal de petición activada.

El coprocesador de Sistema CP0

Núm	ExcCode	Causa
0	Int	Interrupción de un dispositivo hardware de E/S
1	Mod	Fallo de TLB por página modificada (1ª escritura en la página)
2	TLBL	Fallo de TLB (o fallo de página) por lectura
3	TLBS	Fallo de TLB (o fallo de página) por escritura
4	AdEL	Error de dirección por lectura (acceso no alineado, o al espacio de SO no permitido)
5	AdES	Error de dirección por escritura (acceso no alineado, o al espacio de SO no permitido)
6	IBE	Error de bus (instruction fetch) p.e. dirección física inexistente
7	DBE	Error de bus (load/store datos) p.e. dirección física inexistente
8	Sys	Llamada al Sistema Operativo (causada por la instrucción syscall)
9	Bp	Breakpoint (debugging)
10	RI	Instrucción reservada, p.e. código de operación inexistente
11	CpU	Coprocesador no implementado, p.e. acceso a CP0 en modo usuario
12	Ov	Overflow aritmético de enteros (instrucciones add, sub, addi)
13	Tr	Trap (debugging)
15	FPE	Excepción de coma flotante: hay que consultar los detalles en los registros de CP1

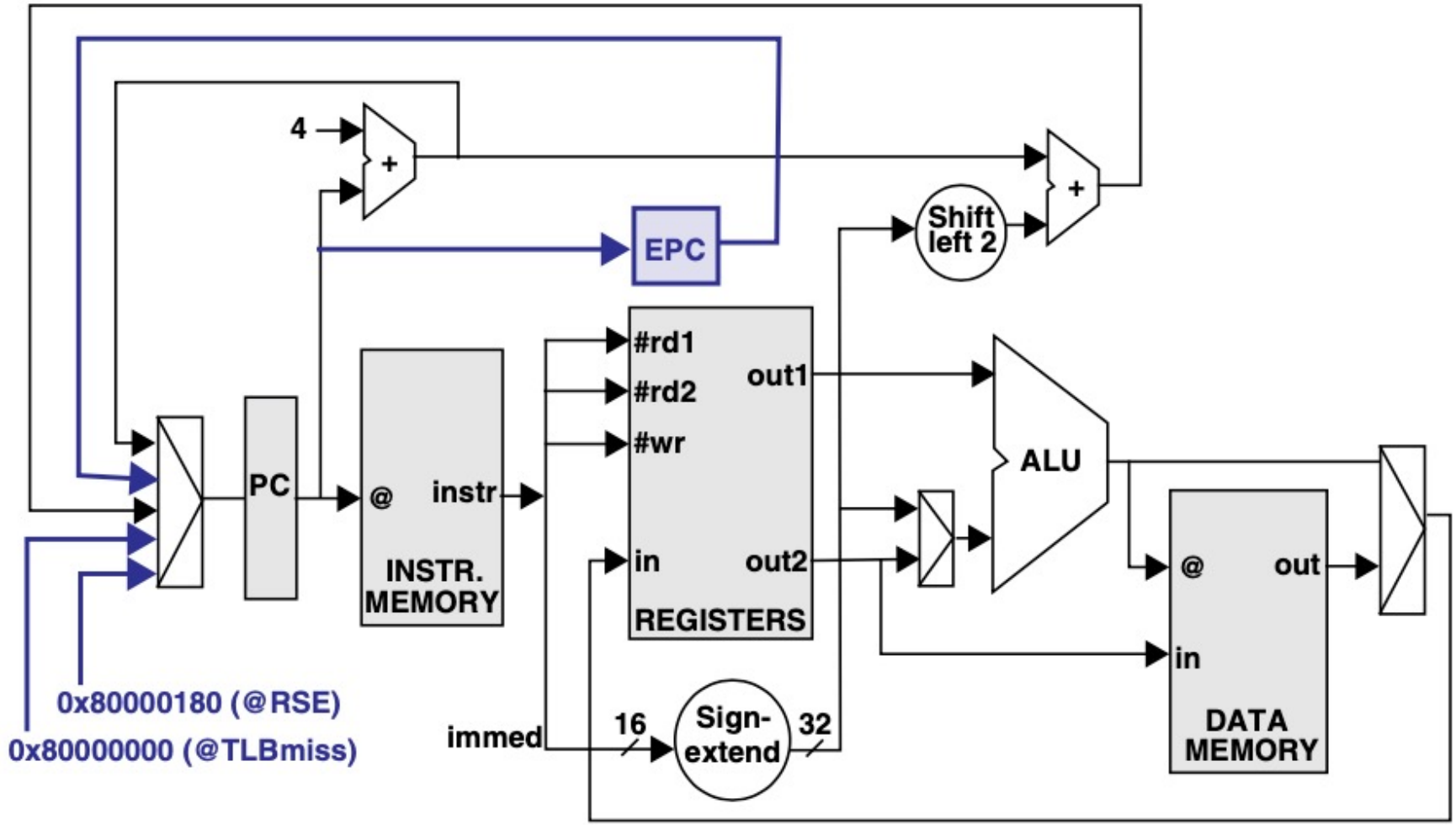
Acciones del Hardware en una Excepción

1. Cuando el hardware detecta una ...
   - ... **excepción**, la instrucción en curso se aborta, no escribe en registros o memoria
     - ✓ No se incrementa el PC
   - ... **interrupción**, finaliza la instrucción en curso
     - ✓ Se comprueban las peticiones pendientes (IPi=1) y habilitados (IMi=1)
     - ✓ Si hay más de una, se selecciona la más prioritaria
2. Guarda el PC en el **registro EPC**
3. Escribe la causa en el campo ExcCode del **registro Cause**
4. Pone a 1 el **bit EXL** (modo sistema, interrupciones inhibidas)
5. Escribe en el **PC** la dirección de la Rutina de Servicio (en MIPS, hay 2 diferentes):
   - **RSE** (genérica), PC = 0x800000180
   - **TLBmiss** (sólo fallos de TLB), PC = 0x80000000

Acciones del Software en una Excepción

1. La RSE ha de preservar el estado del programa interrumpido
   - Salva en la pila **TODOS LOS REGISTROS** ($1 - $31) excepto $k0, $k1
   - No es necesario salvar los registros de coma flotante ($f0 - $f31)
     - ✓ Sólo se salvan si se invoca una rutina en donde se modifiquen
2. Identificar la causa (**ExcCode** en el registro Cause)
3. Saltar a una rutina específica que puede ...
   - ... abortar el programa
   - ... bloquearlo temporalmente, dando paso a otros procesos
   - ... o solucionar el problema y continuar la ejecución del proceso normalmente.
4. En caso de **syscall** hay que sumar 4 a EPC, para que apunte a la siguiente instrucción
5. Restaurar todos los registros salvados en la pila
6. Retornar, usando la instrucción **eret** (exception return)
   - Pone EXL=0 (modo usuario, interrupciones permitidas)
   - Copia EPC en el PC (salta a la dirección de retorno)

Cambios en el procesador para soportar Excepciones



Un ejemplo sencillo de RSE (1)

```
  .ktext 0x80000180
RSE:
  # Salvar Registros
  addiu $sp,$sp,-128
  sw $1,0($sp)
  sw $2,4($sp)
  sw $3,8($sp)
  ...
  sw $31,128($sp)

  # Pasar parámetros Cause y EPC
  ...
```

Salvar TODOS los registros, incluyendo $hi y $lo, pero NO $k0, $k1, $sp y $0.

¿Porqué no salvamos $0?

Un ejemplo sencillo de RSE (2)

```
  .ktext 0x80000180
RSE:
  # Salvar Registros

  # Pasar parámetros Cause y EPC
  mfc0 $a0,$13
  mfc0 $a1,$14
  jal handler_dispatcher
  mtc0 $v0,$14

  # Restaurar Registros
  ...
```

Pasamos Registro Cause y EPC como parámetros en $a0, $a1

La rutina handler_dispatcher invocará al handler que corresponga.

Devuelve como resultado la dirección de retorno que se copia en EPC

Un ejemplo sencillo de RSE (3)

```
  .ktext 0x80000180
RSE:
  # Salvar Registros
  # Pasar parámetros Cause y EPC

  # Restaurar Registros
  sw $1,0($sp)
  sw $2,4($sp)
  ...
  sw $31,128($sp)
  addiu $sp,$sp,128

 # Retorno
```

En un sistema real, estaríamos usando la pila del SO y no la de usuario.

Un ejemplo sencillo de RSE (4)

```
  .ktext 0x80000180
RSE:
  # Salvar Registros
  # Pasar parámetros Cause y EPC
  # Restaurar Registros

  # Retorno
  eret
```

Retornamos al programa de usuario.
Ponemos EXL=0 y PC = EPC

3 Ejemplos concretos

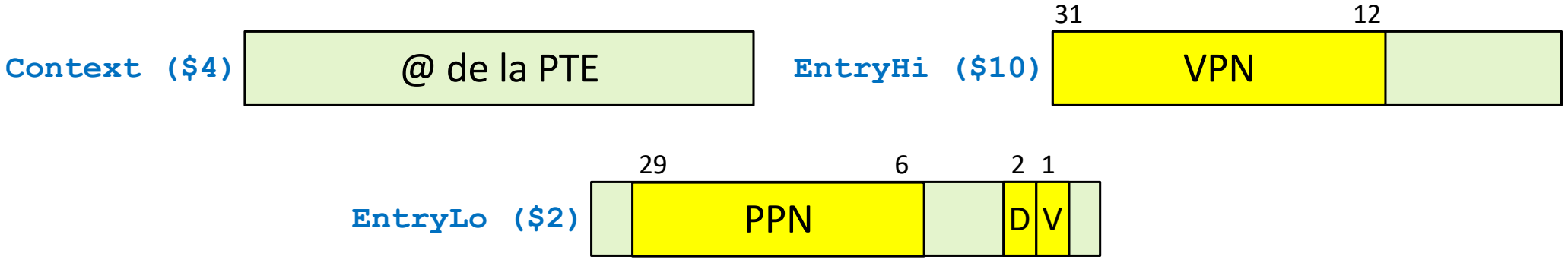
- La excepción por fallo de TLB
- La excepción por llamada al sistema
- Las interrupciones de E/S

Fallo de TLB

- Fallo de TBL
  - La CPU intenta acceder a una dirección de memoria y el TLB no tiene la traducción
- Tratamiento
  - Buscar una entrada libre en TLB (o seleccionar una a reemplazar)
  - Copiar la entrada correspondiente de la TP (TPE)
    - ✓ PPN
    - ✓ V, bit de presencia
    - ✓ D, dirty bit
    - ✓ Añadir la VPN (etiqueta de traducción)

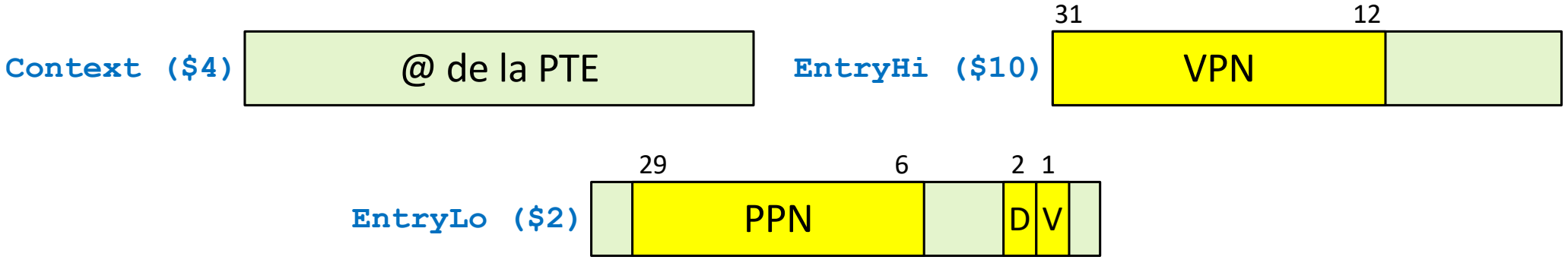
Fallo de TLB en MIPS

- En MIPS, los fallos de TLB causan una excepción
  - Las trata una rutina de Sistema Operativo (software)
- Son bastante frecuentes ⇒ conviene tratarlas eficientemente
  - RSE genérica (@ = 0x80000180), demasiado compleja y lenta
  - Se tratan con la rutina TLBmiss (@ = 0x80000000)
    - ✓ Muy rápida (13 ciclos)
    - ✓ No salva registros en la pila, sólo usa $k1 (reservado por el SO)
- Registros del CP0 para gestionar el TLB



Fallo de TLB en MIPS

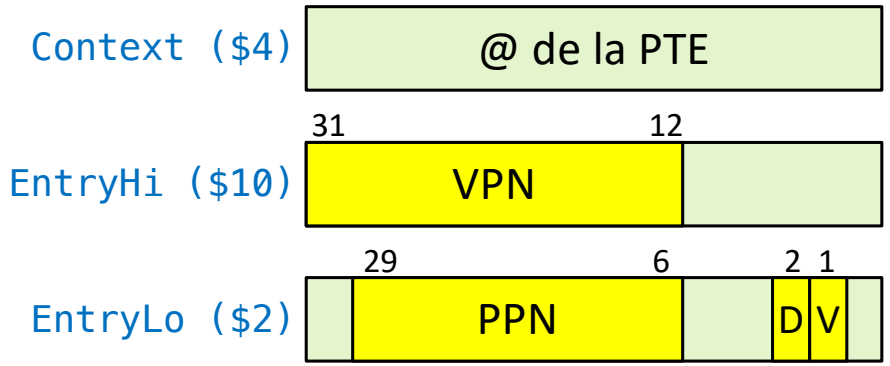
- Cuando el hardware detecta un fallo de TLB ...
  - ... copia el PC en EPC, y pone el bit EXL=1
  - ... escribe en ExcCode el código TLBL (fetch o load) o TLBS (store)
- Además ...
  - ... escribe en el PC la dirección de la rutina TLBmiss (0x80000000)
  - ... escribe la direción de la entrada de la TP (PTE) en el **registro Context**
  - ... escribe los 20 bits del VPN en el **registro EntryHi**



Fallo de TLB en MIPS

- Ejemplo de rutina TLBmiss

```
  .ktext 0x80000180
TLBmiss:
  mfc0 $k1,$4   # copia Context (@PTE) en $k1
  lw $k1,0($k1) # lee la PTE (traducción) en $k1
  mtc0 $k1,$2   # copia la traducción a EntryLo
  tlbwr         # escribe EtryHi|EntryLo en el TLB
  eret          # Retorna el programa
```



TLBWR "TLB WRITE RANDOM"
- Selecciona una entrada del TLB aleatoriamente
- Escribe en ella, EntryHy (VPN) y EntryLo (traducción)

SOFTWARE vs HARDWARE

Fallo de TLB en MIPS

- ¿Qué pasa si en un fallo de TLB el bit V vale 0?
  - Durante el fallo de TLB
    - ✓ La rutina TLBmiss **NO COMPRUEBA** el bit de presencia V
    - ✓ Cuando TLBmiss acaba, retorna a la instrucción que ha provocado el fallo
  - Cuando la instrucción se reejecuta
    - ✓ El TLB (en HIT) comprueba el bit de presencia V
    - ✓ Si V=1, la instrucción se ejecuta normalmente
    - ✓ Si V=0, produce una excepción por **fallo de página** y se invoca la RSE

El TLB en MIPS

Función del bit V del TLB

- Es una copia del bit de presencia (P) de la TP
- Indica si la página está en MP o sólo en disco
- No interviene para determinar si es fallo de TLB o no
  - El fallo de TLB sólo depende de si encuentra el VPN o no
- El bit V interviene cuando reemplazamos una entrada del TLB
  - El algoritmo da preferencia a ocupar entradas con V=0

El TLB en MIPS

Función del bit D del TLB

- En un **fallo de página**, se trae la página de disco a MP.
  - Además, se inicializa el bit D en la TP (0 si es un load, 1 si es un store)
  - El bit D indica si la página en MP se ha modificado respecto al disco
    - ✓ La escritura de páginas de MP usa la política de escritura retardada
- En un **fallo de TLB**, se copia la PTE en el TLB, incluyendo el bit D.
- Cuando un store acierta en el TLB
  - Si D=0 (primer store en la página) se activa D=1
  - También se ha de poner D=1 en la TP
    - ✓ La escritura del bit D del TLB tiene política de escritura inmediata
  - Para poner D=1 en la TP se genera la excepción de "página modificada"
    - ✓ El SO comprueba los permisos de escritura y actualiza el bit D

La Excepción de Llamada al Sistema Operativo

- El **Sistema Operativo**
  - Proporciona acceso seguro y eficiente a los recursos compartidos
    - ✓ Memoria Física
    - ✓ Dispositivos de Entrada/Salida
    - ✓ Tiempo de CPU (ejecución concurrente de múltiples procesos)
- La gestión de los recursos se realiza en **Modo Sistema** (EXL=1)
  - Permite acceder al espacio de memoria (código y datos) reservado al SO
    - ✓ Espacio reservado al SO: **0x80000000 – 0xFFFFFFFF**
    - ✓ Si el usuario accede a este espacio ⇒ excepción (ExcCode = AdEL, AdES)
  - Permite ejecutar instrucciones privilegiadas
    - ✓ Como las que operan en el CP0
    - ✓ Si el usuario ejecuta una instrucción privilegiada ⇒ excepción (ExcCode = CpU)

La Excepción de Llamada al Sistema Operativo

- El acceso a los servicios del SO se hace con una excepción
  - En MIPS, con la instrucción **syscall**
  - La Excepción (ExcCode = Sys) cambia a modo sistema (EXL = 1) y salta a la RSE (que es parte del SO)
- Funciona como una llamada a una subrutina
  - Se pone en $v0 el **código del servicio** solicitado
    - ✓ El SO tiene una subrutina para cada servicio
  - Los **parámetros**, en $a0-$a3 ($f12,$f14 los de CF)
  - Se hace la llamada con **syscall** (no con jal)
  - El **valor de retorno** (si lo hay) se devuelve en $v0 (o $f0 en CF)
- A diferencia de las subrutinas
  - La excepción salta a un punto de entrada único del SO (la RSE)
  - La excepción activa el modo sistema e inhibe interrupciones.

Ejemplos de Servicios del SO

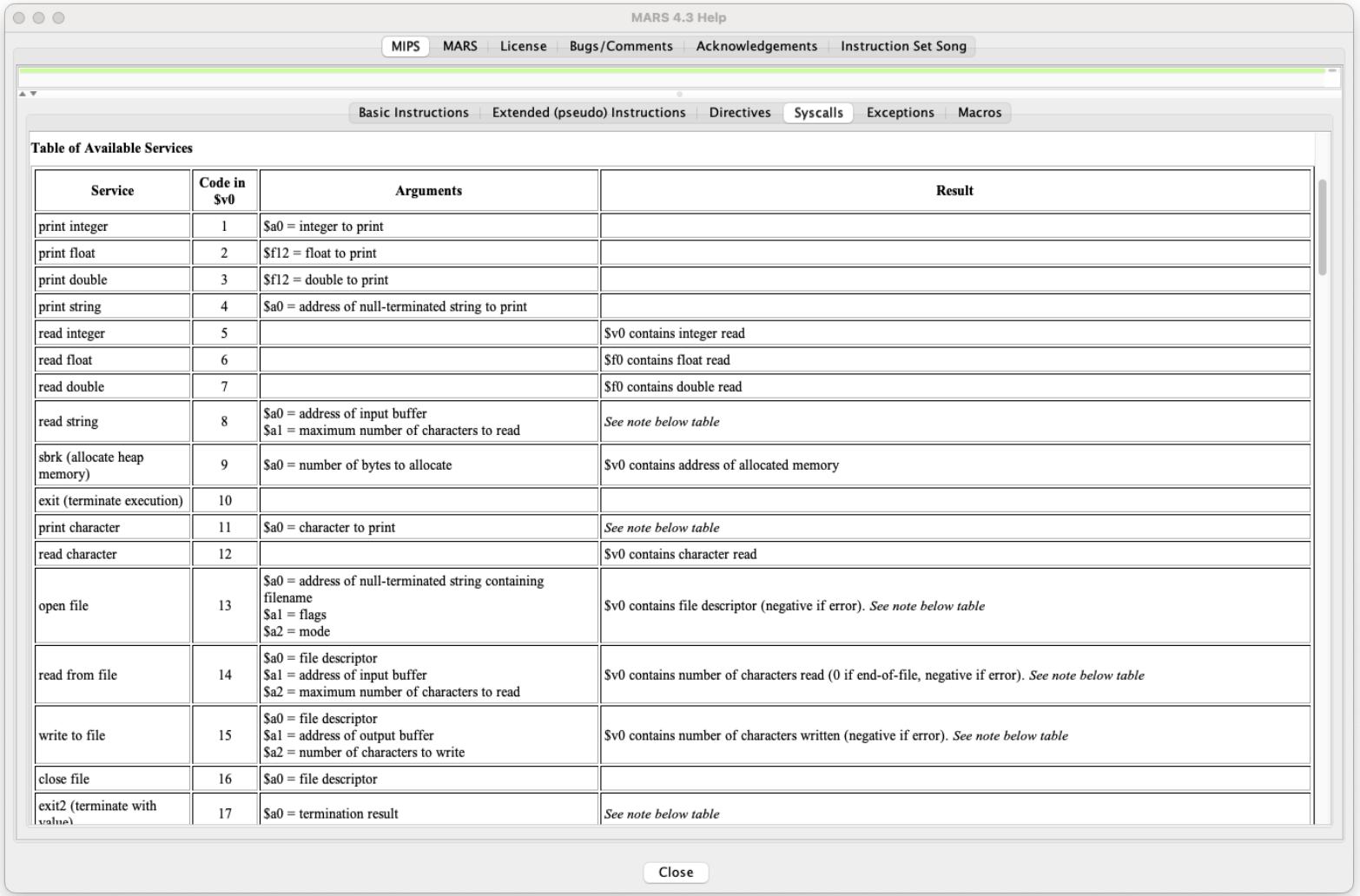


Table of Available Services

Service	Code in $v0	Arguments	Result
print integer	1	$a0 = integer to print	
print float	2	$f12 = float to print	
print double	3	$f12 = double to print	
print string	4	$a0 = address of null-terminated string to print	
read integer	5		$v0 contains integer read
read float	6		$f0 contains float read
read double	7		$f0 contains double read
read string	8	$a0 = address of input buffer $a1 = maximum number of characters to read	*See note below table*
sbrk (allocate heap memory)	9	$a0 = number of bytes to allocate	$v0 contains address of allocated memory
exit (terminate execution)	10		
print character	11	$a0 = character to print	*See note below table*
read character	12		$v0 contains character read
open file	13	$a0 = address of null-terminated string containing filename $a1 = flags $a2 = mode	$v0 contains file descriptor (negative if error). *See note below table*
read from file	14	$a0 = file descriptor $a1 = address of input buffer $a2 = maximum number of characters to read	$v0 contains number of characters read (0 if end-of-file, negative if error). *See note below table*
write to file	15	$a0 = file descriptor $a1 = address of output buffer $a2 = number of characters to write	$v0 contains number of characters written (negative if error). *See note below table*
close file	16	$a0 = file descriptor	
exit2 (terminate with value)	17	$a0 = termination result	*See note below table*

Ejemplo con llamadas al SO

```
  .data
cadena: .asciiz “Esto es una frase\n”
  .text
  ...

  li $v0,4          #llamada 4: print_string(char *p)
  la $a0,cadena
  syscall

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

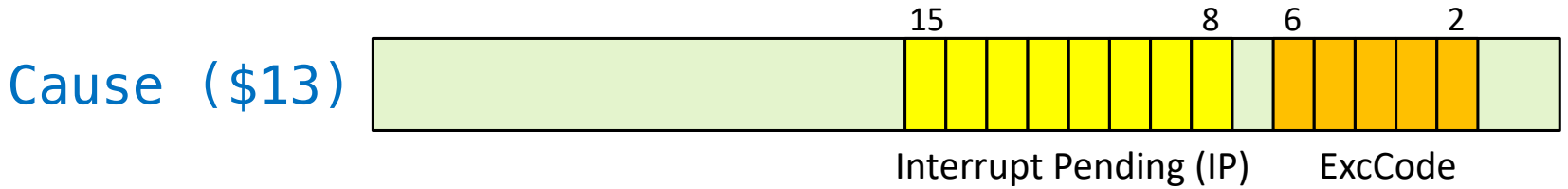
Las Interrupciones de Entrada/Salida

- Los dispositivos de Entrada/Salida sólo son accesibles por el SO
  - Los programas de usuario puede acceder solicitándolo al SO, por medio de una llamada al sistema (**syscall**)
- Las operaciones de E/S tienen latencias enormes (si las comparamos con la velocidad de la CPU). Por ello la comunicación entre CPU y dispositivos de E/S requiere **sincronización**.
- Dos tipos de **Sincronización**
  - **Por Encuesta**, el programa espera que el dispositivo esté listo, consultando repetidamente su estado (espera activa)

```
do {
  estado = ConsultarEstadoDisp();
} while (estado != READY);
```

  - **Por Interrupciones**, el SO ejecuta otras cosas mientras que el dispositivo esté ocupado (permite optimizar el uso de la CPU)
    - ✓ Cuando el dispositivo está preparado, avisa a la CPU, activando una señal de **petición de interrupción (INT)**

Las interrupciones de E/S en MIPS



- Cuando un dispositivo finaliza una operación de E/S o requiere atención
  - Solicita interrupción, activando la señal INT (asíncrono)
  - La instrucción en curso no se interrumpe
  - La petición queda pendiente y anotada en el bit IP_i asociado al dispositivo de E/S
- Cuando acaba la instrucción en curso se incrementa el PC
  - El procesador comprueba si hay peticiones pendientes en el campo IP
  - Si hay alguna bit $IP_i = 1$, se genera una excepción (ExcCode = Int)
- Cuando la RSE hace el tratamiento de la interrupción
  - Lo notifica al dispositivo (señal INTA)
  - El dispositivo desactiva la petición (señal INT), el bit IP_i se pone a 0

Las interrupciones de E/S en MIPS

- Cuando se invoca la RSE
  - Se pone el bit EXL=1, la rutina se ejecutará en modo sistema
  - Con EXL=1, **todas las interrupciones están deshabilitadas** y se ignoran.
- El SO puede deshabilitar las interrupciones de un dispositivo (i)
  - Poniendo a 0 el bit correspondiente del campo IM en el registro Status (bit IM_i)
- Si el dispositivo (i) solicita una interrupción, sólo se produce una excepción
  - Si $IM_i = 1$ y EXL = 0
- Identificación del dispositivo por software
  - La RSE Comprueba los campos IP e IM para determinar que dispositivo atiende
  - El orden en que se comprueba establece el orden de prioridades

Preguntas TEST ¿cierto o falso? (exámenes 2016-2019)

1. Si el bit EXL vale 1, las interrupciones serán ignoradas.
2. La excepción por acceso no alineado a memoria puede ser inhibida a través del campo Interrupt Mask.
3. Una excepción no puede ser atendida hasta que la instrucción en curso haya finalizado.
4. Un acceso a memoria nunca cambiará el estado de la tabla de páginas si se produce un acierto al TLB en una entrada con el bit de validez a 1.
5. La rutina RSE de tratamiento de excepciones de MIPS sigue las reglas del ABI que se establecen para programar subrutinas
6. En el MIPS se detecta que un acceso a memoria provoca un fallo de página consultando el bit V en el TLB.

Preguntas TEST ¿cierto o falso? (exámenes 2016-2019)

7. Si el acceso a datos de una instrucción produce un acierto en el TLB, pero el bit V vale 0, entonces la instrucción causará una excepción por fallo de página.
8. Una misma instrucción puede causar durante su ejecución 2 fallos de página.
9. Un fallo de TLB no implica que se produzca un fallo de página.
10. Cuando se produzca un fallo de página, habrá uno o dos accesos a disco dependiendo del valor del Dirty Bit de la página a reemplazar
11. Al inicio de la rutina genérica de servicio a excepciones de MIPS (RSE), ésta sólo ha de salvar aquellos registros seguros que se modifique durante la ejecución de la RSE.
12. Un programa en modo usuario puede copiar un registro cualquiera de la CPU en el coprocesador CP0 utilizando la instrucción mtc0.

Preguntas TEST ¿cierto o falso? (exámenes 2016-2019)

13. En el MIPS, el campo IM (Interrupt Mask) del registro Status usado en la gestión de interrupciones, sirve para indicar las peticiones de interrupción que no han de ser atendidas cuando se acabe la ejecución de la instrucción actual.
14. La primera vez que se escribe en una página se genera una excepción, el tratamiento de la cual consiste en escribir la página modificada en disco.
15. Cuando buscamos la traducción de un número de página (VPN) en el TLB, para que se produzca un acierto, hay que encontrar la entrada con el mismo VPN y el bit de presencia V=1.
16. Las escrituras del bit de Dirty del TLB siguen una política de escritura inmediata.
17. En MIPS, una instrucción `lb` nunca puede producir una excepción por acceso no alineado de la dirección

Preguntas TEST ¿cierto o falso? (exámenes 2016-2019)

18. Los fallos de TLB en MIPS provocan una excepción y se ejecuta la RSE genérica.
19. En MIPS, cuando se acaba de ejecutar la RSE, se vuelve a la instrucción siguiente a la que ha provocado la excepción o la interrupción.
20. La sincronización de los dispositivos de Entrada/Salida en un computador, se gestiona exclusivamente por interrupciones.
21. En MIPS, la traducción de direcciones virtuales a físicas al ejecutar una instrucción `lw` puede producir hasta 4 excepciones.

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Arquitectura de Computadors

Estructura de Computadores

Tema 8: Excepciones e Interrupciones

Agustín Fernández

Departament d'Arquitectura de Computadors

Facultat d'Informàtica de Barcelona

Universitat Politècnica de Catalunya