UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Arquitectura de Computadors

Tema 8
Excepcions i Interrupcions

Estructura de Computadors (EC)
2023 - 2024 Q2
Adrià Armejach (adria.armejach@upc.edu)

Cas especial d'excepció: la fallada de TLB

Fallada de TLB en MIPS

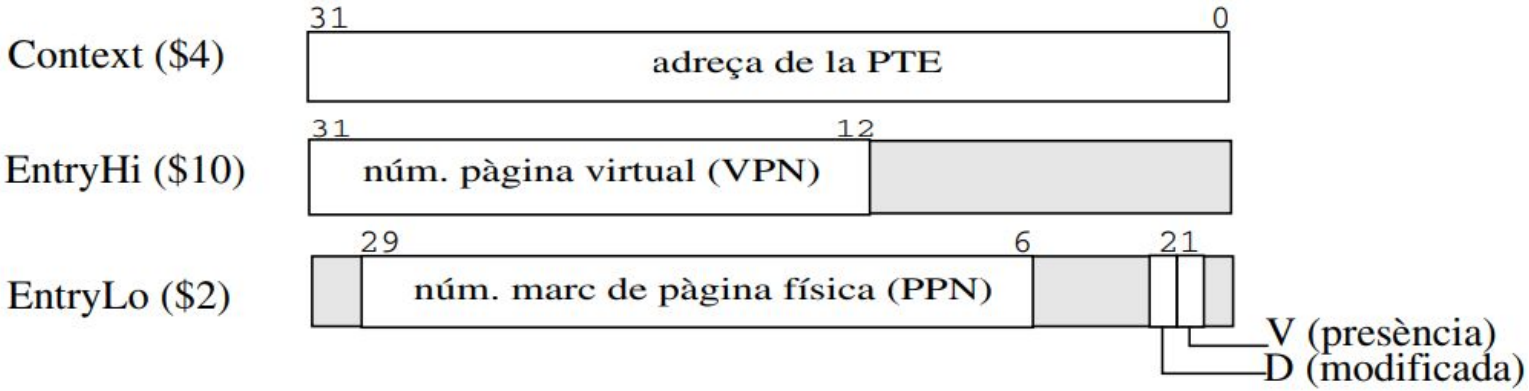
- Fallada de TLB
  - El processador accedeix a una adreça virtual per la qual el TLB no conté una traducció
- Dos tipus d'excepció:
  - ExcCode = TLBL: lectura d'instrucció o load
  - ExcCode = TLBS: store
- Tractament de la fallada de TLB
  - Copiar l'entrada corresponent de la taula de pàgines al TLB: PPN, bit de presencia (V) i bit de dirty (D)
  - Escriure a l'entrada del TLB el VPN (número de pàgina virtual)

Fallada de TLB en MIPS

- En MIPS les fallades de TLB produeixen una excepció i es gestionen per software (SO)
- Les fallades de TLB són freqüents i convé tractar-les eficientment
  - La RSE genèrica (adreça 0x80000180) resulta massa complexa i lenta
  - MIPS utilitza la rutina TLBmiss (adreça 0x80000000)
    - Molt ràpida (normalment menys de 13 cicles)
    - No salva els registres a la pila, només modifica `$k1` que està reservat al SO

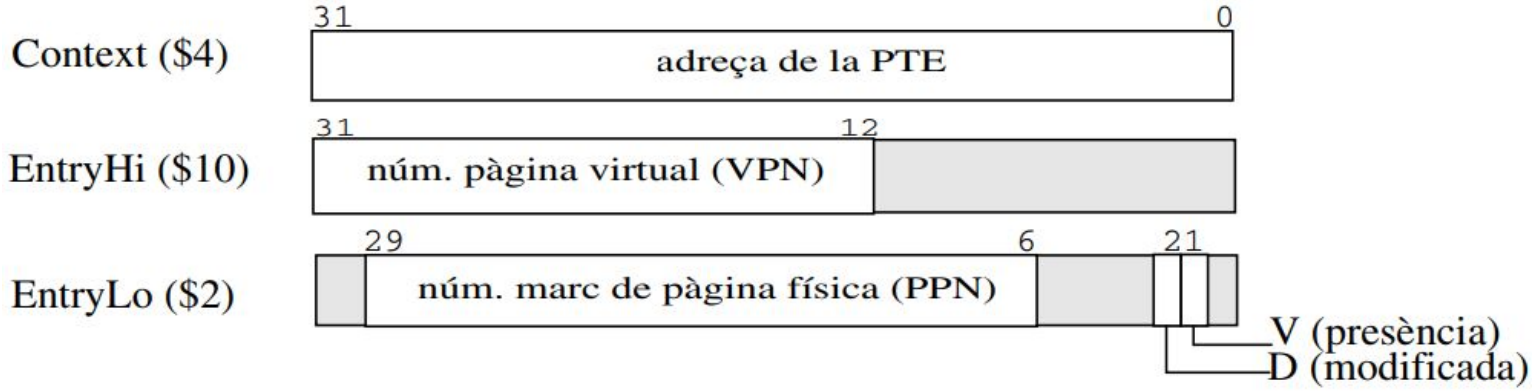
Fallada de TLB en MIPS

- En MIPS el TLB forma part del coprocessador CP0 i es gestiona mitjançant els següents registres:



Fallada de TLB en MIPS - Accions Hardware

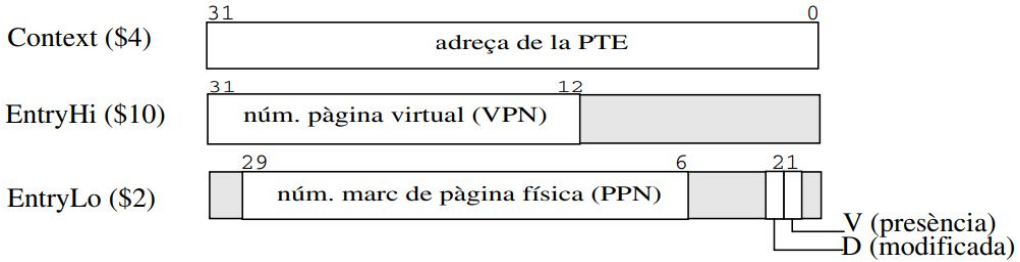
- Al detectar una fallada de TLB, el hardware realitza les següents accions:
  - Escriu l'adreça de l'entrada de la taula de pàgines (PTE) al registre *Context*
  - Escriu els 20 bits del VPN al registre *EntryHi*



Fallada de TLB en MIPS - Accions Software

- Exemple 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
```



- Instrucció **tlbwr:** l'entrada a reemplaçar es selecciona de manera pseudoaleatòria (random)

Bit V del TLB

- La rutina TLBmiss no comprova el bit de presència (V)
- Un cop resolta la fallada de TLB es reexecutarà la mateixa instrucció
- Al reexecutarse, es produirà un encert de TLB i es comprovara el bit de presència (V) del TLB
  - Si V = 1, la instrucció s'executa amb normalitat
  - Si V = 0, es produeix una nova excepció per fallada de pàgina
    - S'invoca la RSE genèrica

Bit V del TLB (recordatori)

- El bit V del TLB és una copia del bit de presència de la taula de pàgines
  - Si V=1 la pàgina es troba a MP en un marc de pàgina
  - Si V=0 la pàgina només està a disc
- L'encert o fallada de TLB no depen del bit V, només depen de que el VPN es trobi al TLB
  - Podem tenir encert de TLB amb V = 0
- El bit V s'utilitza en cas de reemplaçament
  - Se seleccionen preferentment les entrades del TLB amb bit V = 0

Bit D del TLB (recordatori)

- El bit D (dirty) indica si la pàgina en memòria física ha estat modificada respecte la còpia en disc
- El bit D és l'únic bit del TLB modificat per l'execució del programa
  - Es posa a 1 la primera vegada que el programa realitza un store sobre la pàgina
  - S'utilitza política d'escriptura immediata per actualitzar el bit D a l'entrada de la taula de pàgines

Bit D del TLB

- Quan es produeix una fallada de pàgina:
  - La RSE copia del disc a memòria física i actualitza les entrades corresponents de la taula de pàgines i del TLB
  - Si l'accés és una escriptura el bit D es posa a 1, si és una lectura a 0
- Quan una escriptura produeix un encert al TLB:
  - Es comprova el bit D. Si val 0 és la primera escriptura, es produeix una excepció de "pàgina modificada". La RSE posarà el bit D a 1 a la taula de pàgines i el TLB.
- Com que s'utilitza escriptura immediata, al reemplaçar una entrada del TLB es pot sobreescriure directament

Cas especial d’excepció: crides al sistema

Protecció: mode usuari/sistema

- El sistema operatiu (SO) proporciona accés segur i eficient als recursos compartits
  - Memòria física
  - Dispositius d'E/S
  - Processador (execució concurrent de múltiples programes)
- La gestió d'aquests recursos es realitza en mode sistema
  - EXL = 1 al registre *Status*
  - Permet accedir a codi/dades del SO a la regió de memòria protegida
    - Excepció AdEL (lectura) o AdES (escriptura) al accedir a la regió protegida en mode usuari
  - Permet executar instruccions que operen sobre el CP0
    - Excepció CpU al executar una instrucció privilegiada (CP0) en mode usuari

Crides al sistema

- L'accés als serveis del SO es realitza mitjançant una excepció
- En MIPS s'utilitza la instrucció syscall
  - Canvia a mode sistema (EXL = 1)
  - Salta a la RSE, que forma part del SO
- Funcionen de manera similar a una subrutina
  - El codi del servei es pasa en `$v0`
    - El SO té una subrutina per cada servei
  - Els paràmetres es passen en `$a0` - `$a3` (`$f12` per coma flotant)
  - S'invoca al SO mitjançant syscall (similar a jal)
  - Si la subrutina retorna resultat es pasa en `$v0` (`$f0` per coma flotant)

Codis de serveis 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`	$v0 = address
exit	10		
print_char	11	`$a0 = char`	
read_char	12		$v0 = char
open	13	`$a0 = filename (string)` `$a1 = flags, $a2 = mode`	$a0 = file descriptor
read	14	`$a0 = file descriptor,` `$a1 = buffer, $a2 = length`	$a0 = num chars read
write	15	`$a0 = file descriptor,` `$a1 = buffer, $a2 = length`	$a0 = num chars written
close	16	`$a0 = file descriptor`	
exit2	17	`$a0 = result`	

Exemples de crida 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
```

Cas especial d'excepció: les interrupcions

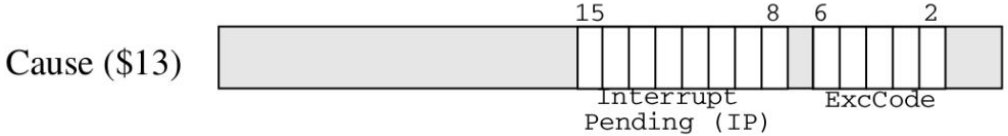
Interrupcions des de dispositius d'E/S

- El SO proporciona accés als dispositius d'E/S
- Un programa d'usuari pot accedir a un dispositiu d'E/S mitjançant una crida al sistema
- El dispositiu d'E/S pot iniciar la comunicació mitjançant una senyal d'interrupció
    - Quan la CPU detecta la interrupció s'invoca la RSE

Sincronització amb dispositius d'E/S

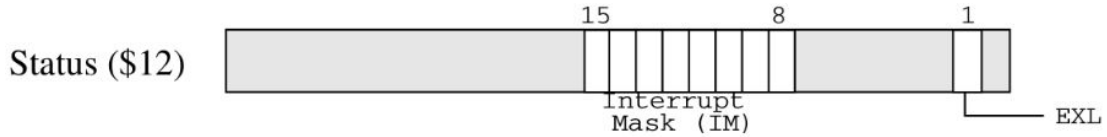
- La comunicació amb els dispositius d'E/S està dominada per la necessitat de tolerar les enormes latències de les operacions
- Dos tipus de sincronització
  - **Sincronització per enquesta:** El programa espera a que la operació acabi, consultant repetidament l'estat de la operació
  - **Sincronització per interrupcions:** El SO planifica i executa altres programes mentres es realitza l'operació. Quan el dispositiu d'E/S ha acabat envia una senyal d'interrupció

Interrupt pending del registre Cause



- Cada dispositiu d'E/S està associat a un bit del camp Interrupt Pending (IP) del registre *Cause* del CP0
  - El bit s'activa quan el dispositiu vol notificar que ha finalitzat una operació o que necesita atenció
  - El bit es desactiva quan el processador ha atès la interrupció o quan ha passat un determinat temps d'espera
- El camp IP guarda les peticions d'interrupció que han arribat i estan pendents de ser ateses
- NO s'interromp la instrucció en curs
  - Quan finalitza l'execució de la instrucció, el processador comprova el camp IP i genera una excepció si hi ha algun bit a 1 (posa *ExcCode* = Int)

Interrupt Mask del registre Status



- Les interrupcions només s'atenen en mode usuari (bit `EXL = 0`)
  - En mode sistema (bit EXL = 1) les interrupcions estan deshabilitades
- El SO pot deshabilitar les interrupcions dels dispositius d'E/S de manera selectiva
  - Utilitzant el camp de Interrupt Mask (IM) del registre *Status*
    - Si el dispositiu *i* realitza una petició de interrupció només es genera una excepció si $IM_i = 1$ i EXL = 0

Identificació del dispositiu per software

- Quan el processador detecta una petició de interrupció i EXL = 0
  - La RSE comprova el camp Interrupt Pending per determinar quin és el dispositiu que ha fet la petició
  - … i el camp Interrupt Mask per decidir si té permís per interrompre
    - En cas de múltiples peticions s'estableix un sistema de prioritats

Tests

Test 1!

www.menti.com

codi: 1761 2485

Test 2!

www.menti.com

codi: 4123 5693

Cert o fals

	Afirmació	V	F
1.-	Si la CPU rep una petició d'interrupció per part d'un dispositiu d'E/S, la instrucció en curs s'interromp sense arribar a produir cap resultat, i la CPU passa a executar la rutina RSE.		
2.-	En MIPS, la traducció d'adreces virtuals a físiques en una instrucció `lw` pot arribar a produir fins a quatre excepcions.		
3.-	Quan cerquem la traducció d'un número de pàgina (VPN) al TLB, perquè hi hagi un encert de TLB cal trobar una entrada amb el mateix VPN i que tingui el bit de presència V=1.		
4.-	Les escriptures del bit de Dirty del TLB segueixen una política d'escriptura immediata.		
5.-	Les fallades de TLB en el MIPS provoquen una excepció i s'executa la RSE genèrica.		
6.-	En MIPS, quan s'acaba d'executar la RSE, es torna a la instrucció següent a la que ha provocat l'excepció o la interrupció.		
7.-	La sincronització dels dispositius d'entrada/sortida en un computador, es gestiona exclusivament per interrupcions.		
8.-	En MIPS, una instrucció `lb` mai pot produir una excepció per accés no alineat.		
9.-	En un computador que disposa de memòria virtual amb TLB sempre cal fer dos accessos a memòria principal per cada referència: un a la Taula de Pàgines i l'altre a l'adreça de la referència.		
10.-	L'excepció de divisió per zero pot ser inhibida a través del camp Interrupt Mask.		

Cert o fals

	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		

Problema

Problema

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
- Capacitat total: 8 blocs
- Mida del bloc: 64 bytes (16 words)
- Escriptura immediata amb assignació (és la que implementa el MARS)

Partim del següent programa en alt nivell (podeu trobar el programa equivalent en MIPS al fitxer subgrups41_42-excache.s):

```
int  V[16]
int  M[8][16]

main()
{
      int x, i, j;/* Es guarden en registres */
      x = 0;
      for (i=0; i<8; i++)
          for (j=0; j<16; j++) {
              x = x + M[i][j];
              x = x + V[j];
          }
}
```

Problema

Considera que les variables globals que conté el programa estan emmagatzemades a memòria a partir de l'adreça 0x10010000. Considera també que la MC és inicialment buida. Es demana (respon al fitxer subgrups41_42-excache.txt):

a. Quantes fallades es produeixen en cada iteració del bucle extern?

b. Si has contestat bé l'apartat anterior observaràs que en la iteració i=7 es produeixen moltes més fallades que en les altres iteracions. Descriu amb precisió quina és la causa que en aquesta iteració hi hagi moltes mes fallades.

c. Quin és el mínim nombre de fallades que podem aconseguir en aquest programa canviant el grau d'associativitat de la cache (però sense alterar ni la capacitat total ni la mida del bloc)? Quin és el mínim grau d'associativitat amb el qual s'aconsegueix aquest resultat? Justifica per què l'associativitat redueix el nombre de fallades en aquest programa.

d. Quin és el mínim nombre de fallades que podem aconseguir en aquest programa canviant la mida del bloc i el grau d'associativitat (però sense alterar la capacitat total)? Quins són el grau d'associativitat i la mida de cache amb els quals s'aconsegueix aquest resultat? Justifica per què el canvi de la mida del bloc redueix el nombre de fallades en aquest programa.