

Bloc IV. Unitat d'entrada/eixida

Tema 7. Adaptadors i interfícies

Tema 8. Mecanismes de sincronització

Tema 9. Tècniques de transferència

Objectius

- Connectar interfícies de perifèric al bus
- Escriure programes en ensamblador que tracten interfícies de perifèric senzilles
- Escriure programes que se sincronitzen amb els perifèrics fent consulta d'estat o responent a interrupcions
- Manipular els registres de configuració de les excepcions del MIPS R2000
- Escriure fragments del manejador d'excepcions del R2000

Contingut

- Interfícies de perifèric
 - Conceptes i exemples
 - Adaptadors
 - Implementació de la selecció
 - Sincronització
 - Conceptes.
 - Sincronització per consulta d'estat
 - Sincronització per interrupcions
 - Interrupcions i excepcions
 - Fonts de les excepcions (MIPS)
 - Modes de funcionament (MIPS)
 - Registres de maneig (MIPS)
 - Maneig d'excepcions
 - Gestió del context
 - Variables del manejador
 - Estructura del manejador
 - Entrada/eixida mitjançant el sistema operatiu
 - Excepcions i sistema operatiu
 - Funcions del sistema
 - Commutació de processos
 - Mecanismes de transferència
 - Dispositius de bloc
 - Transferència per programa
 - Transferència per accés directe a la memòria
-

Les interfícies dels perifèrics

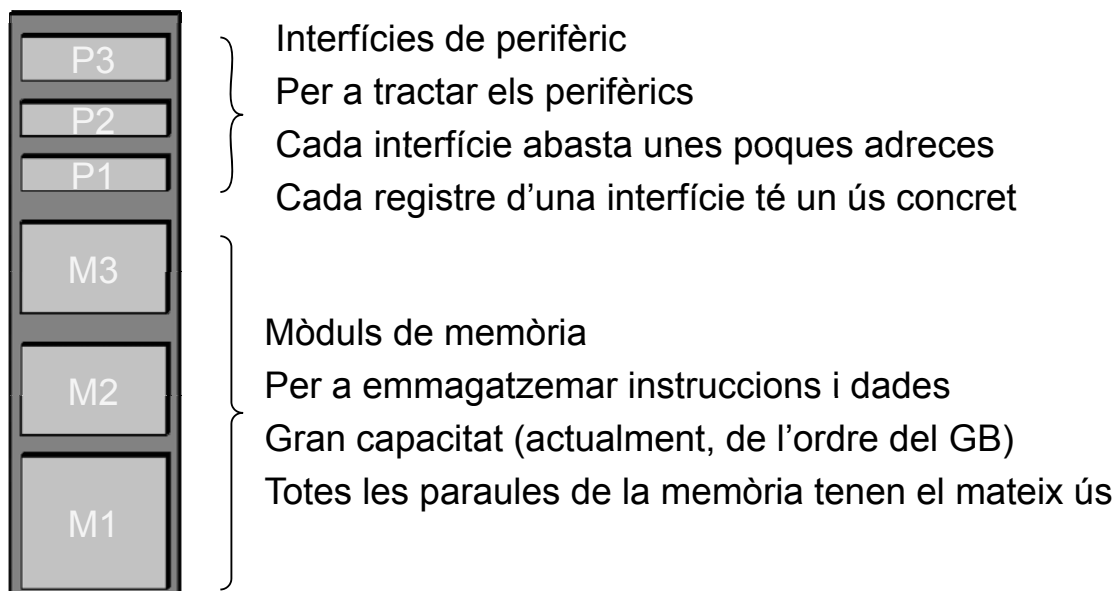
Concepte d'interfície

- Interfície d'un perifèric
 - És un conjunt (heterogeni) de registres que permeten els programes comunicar-se amb un perifèric donat
 - Cada perifèric té la seua interfície que, en general, serà distinta de la de qualsevol altre perifèric quant al nombre i a l'ús dels registres
 - Cada registre de la interfície té una adreça en l'espai d'adreçament del processador
 - El conjunt de registres de cada interfície ocupa adreces consecutives en l'espai d'adreçament a partir d'una certa adreça: l'adreça base de la interfície
 - Els registres són accessibles mitjançant instruccions de lectura i escriptura en l'espai d'adreçament, però la seua funció no és d'emmagatzemar dades o instruccions

5

Adreçament de les interfícies

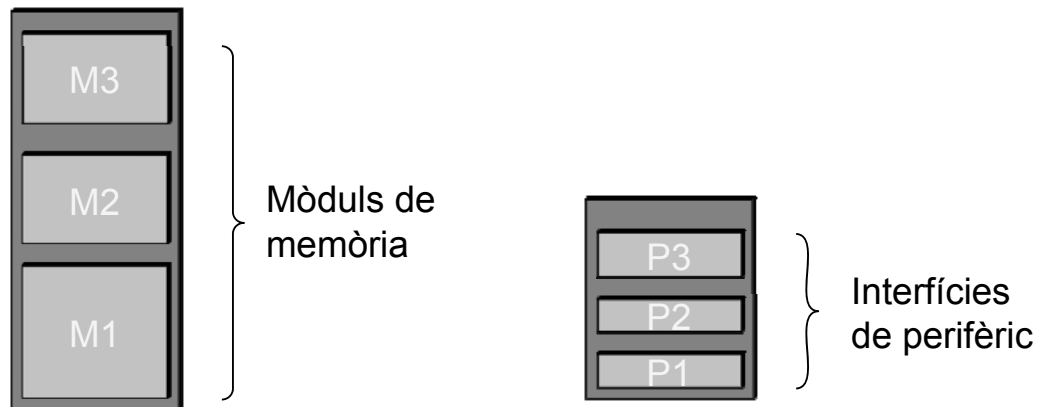
- Mapa d'adreçament únic (MIPS)
 - Les interfícies comparteixen l'espai d'adreçament amb la memòria
 - L'accés als registres es fa amb les instruccions load i store



6

Adreçament de les interfícies

- Mapa d'adreçament separat (Intel)
 - Les interfícies i la memòria tenen espais d'adreçament separats
 - L'accés a la memòria es fa amb instruccions de tipus load/store
 - L'accés als perifèrics es fa amb instruccions de tipus input/output



7

Concepte d'interfície

- Registres
 - Un registre pot ser accessible per a lectura, per a escriptura o per a ambdues operacions
 - Escriure en un registre de lectura no té cap efecte; llegir d'un registre d'escriptura no dona cap informació útil
 - El valor d'un registre pot estar estructurat o no. Si està estructurat, cada bit o grup de bits del registre té un significat propi, independent de la resta
 - En general, entre els bits útils d'un registre hi pot haver d'altres indefinits

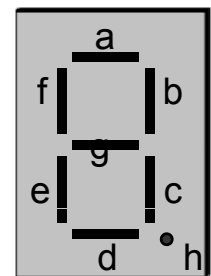
8

Exemples d'interfície

Exemple 1: interfície d'un visualitzador

- Visualitzador de 7 segments amb intermitència de 0 a 7 Hz
- Adreça base (AB) = 0xFFFF0100
- Descripció dels registres:

Nom	Adreça	Accés	Estructura
Ordres	AB	Esc.	<p>ON (bit 0): encés a 1, apagat a 0 <i>freq</i> (bits 6..4): freqüència intermitent en Hz <i>freq</i> = 0: continu</p>
Dades	AB+4	Esc.	<p>bits <i>a...h</i>: a 1 activen el segment corresponent</p>



9

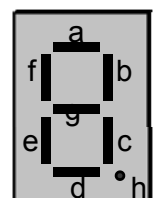
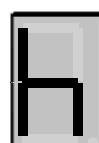
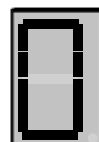
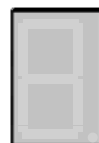
Exemples d'interfície

Exemple 1: interfície d'un visualitzador

```

    la $t0,0xFFFF0100
# apaga el visualitzador
    sw $zero,0($t0)
    ...
# presenta un zero continu
    li $t1,0x3F
    sw $t1,4($t0)
    li $t1,1
    sw $t1,0($t0)
    ...
# presenta una "h" a 2 Hz
    li $t1,0x74
    sw $t1,4($t0)
    li $t1,0x21
    sw $t1,0($t0)

```

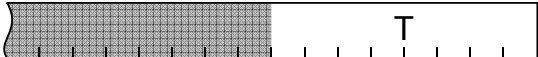


Exemples d'interfície

▪ Exemple 2: interfície d'un termòmetre

– Adreça base (AB) = 0xFFFF0200

– Descripció dels registres:

Nom	Adreça	Accés	Estructura
Estat	AB	Lect.	(per definir)
Ordres	AB	Esc.	(per definir)
Dades	AB+4	Lect.	
T (bits 7...0) temperatura entre 0 i 255 °C			

11

Adaptació dels perifèrics

▪ Diversitat dels perifèrics

– Uns quants exemples:

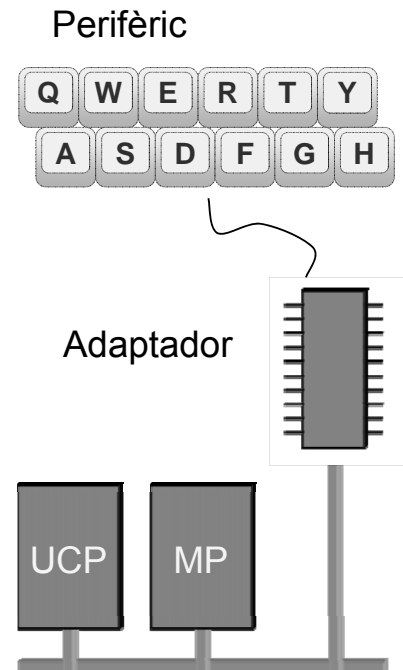
Dispositiu	Ús	Medi físic
Teclat	Entrada de text	Electromecànic
Ratolí	Entrada gràfica	Òptic
Monitor	Visualització	CRT, matriu TFT
Disc dur	Emmagatzemament	Magnètic
DVD	Emmagatzemament	Òptic
Xarxa	Comunicació	Cable Ethernet, línia telefònica
Impressora	Visualització	Electrostàtic, injecció de tinta, etc

12

Adaptació dels perifèrics

▪ Adaptadors

- Un adaptador és el dispositiu que permet connectar el perifèric a un bus
 - Cada perifèric necessita el seu adaptador
 - Fa la conversió entre la tecnologia pròpia del perifèric i els senyals del bus
- Construeix la interfície visible pel programador
 - Conté els registres
 - Tradueix les ordres fetes sobre la interfície en accions sobre el perifèric
 - Tradueix les actuacions externes fetes sobre el perifèric en informació d'estat
 - Permet la transferència de les dades

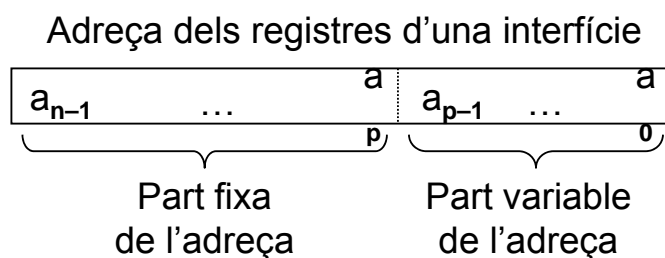


13

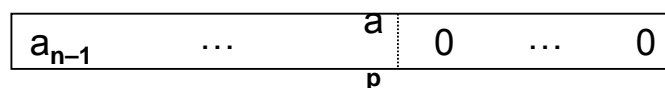
Selecció

▪ Selecció de la interfície

- Els registres de la interfície ocupen adreces consecutives en l'espai d'adreçament



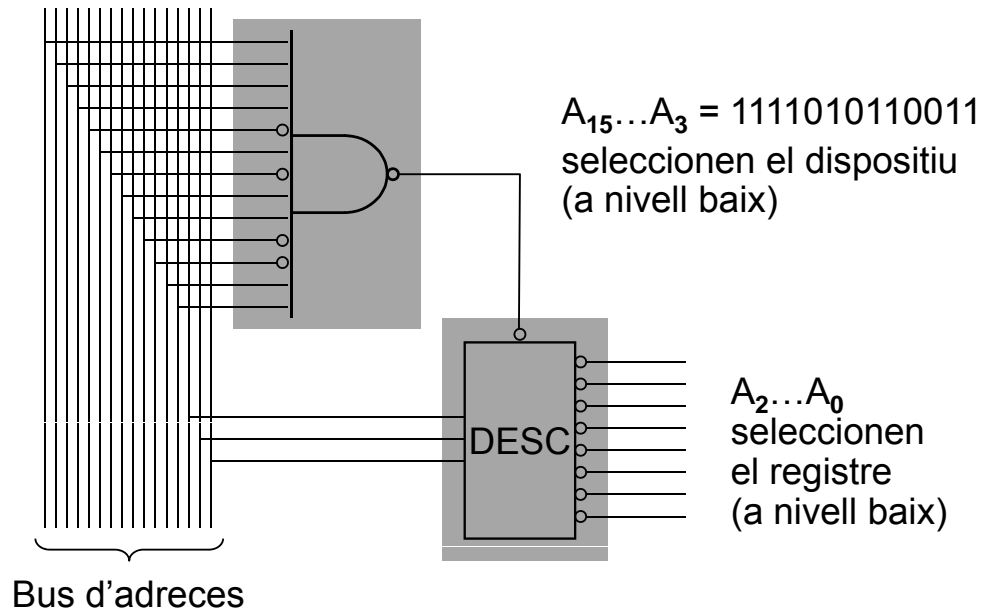
- L'adreça base té la part variable = 0



14

Selecció

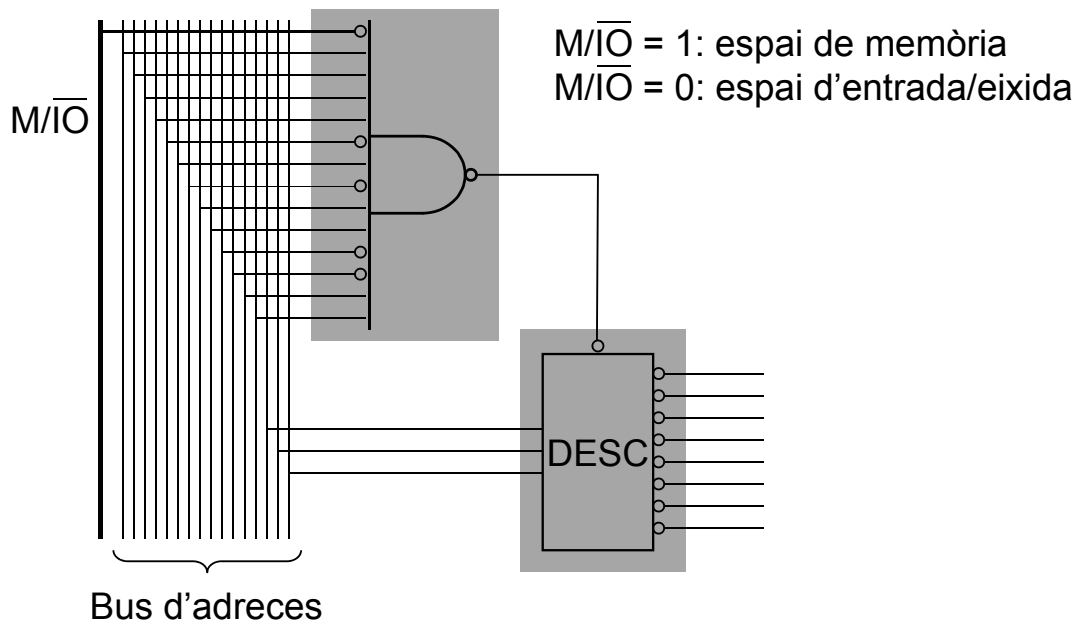
- Implementació de la selecció (mapa únic)
 - Exemple amb 16 línies d'adreça, amplitud de paraula de 8 bits
 - Interfície amb huit registres ($n=16$, $p=3$), $AB = 0xF598$



15

Selecció

- Implementació de la selecció (mapa separat)
 - Una línia en el bus permet distingir entre els dos espais d'adreçament

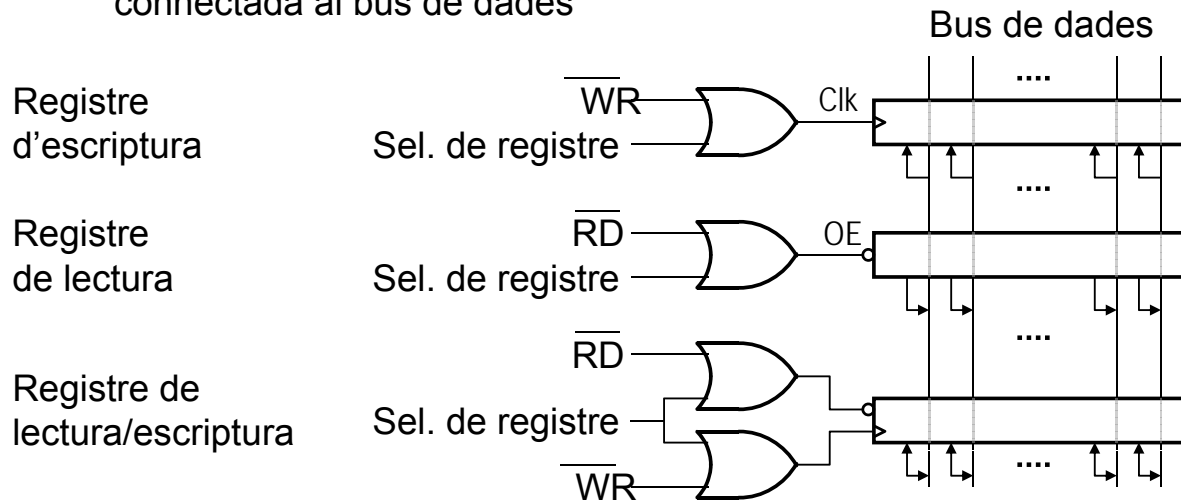


16

Selecció

Selecció i operació dels registres

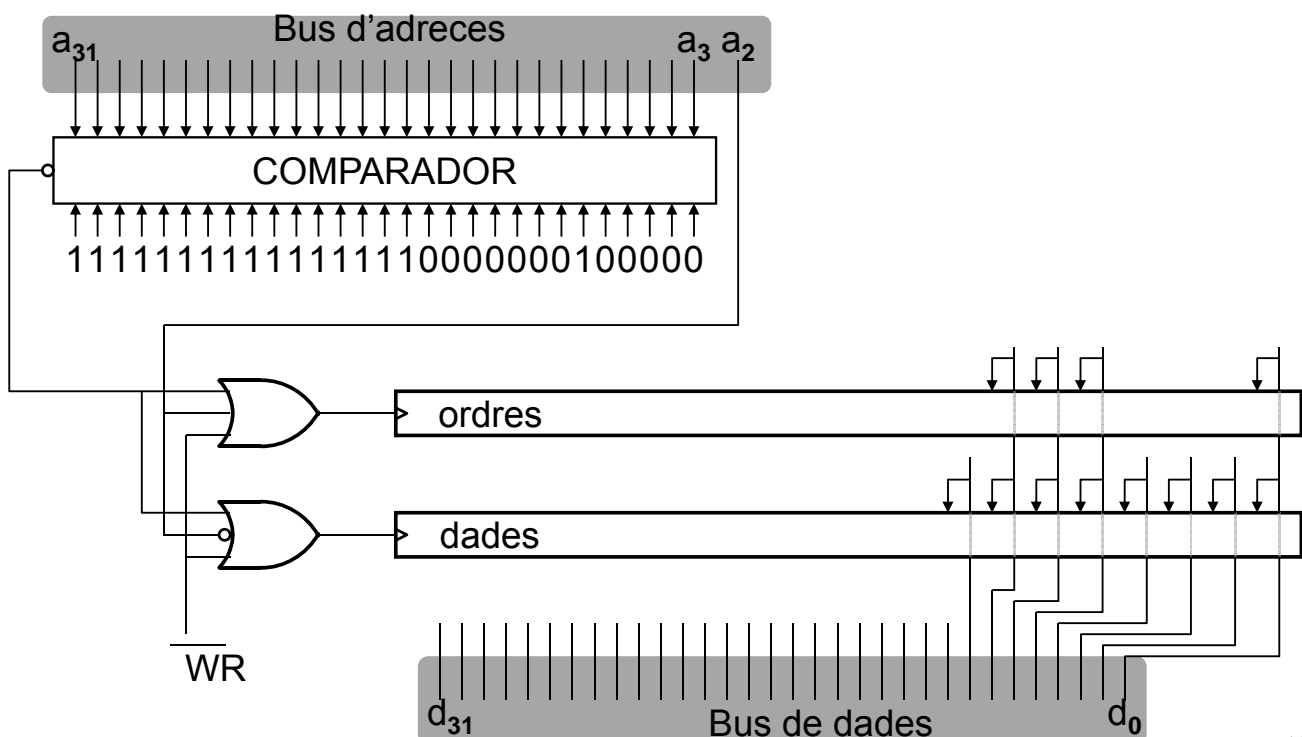
- Suposem dues línies de control en el bus actives a nivell baix: RD per a lectura i WR per a escriptura
- Els registres tenen senyal d'escriptura (Clk, flanc de rellotge), de lectura (OE) o ambdues i entrada/eixida de dades paral·lela connectada al bus de dades



17

Selecció

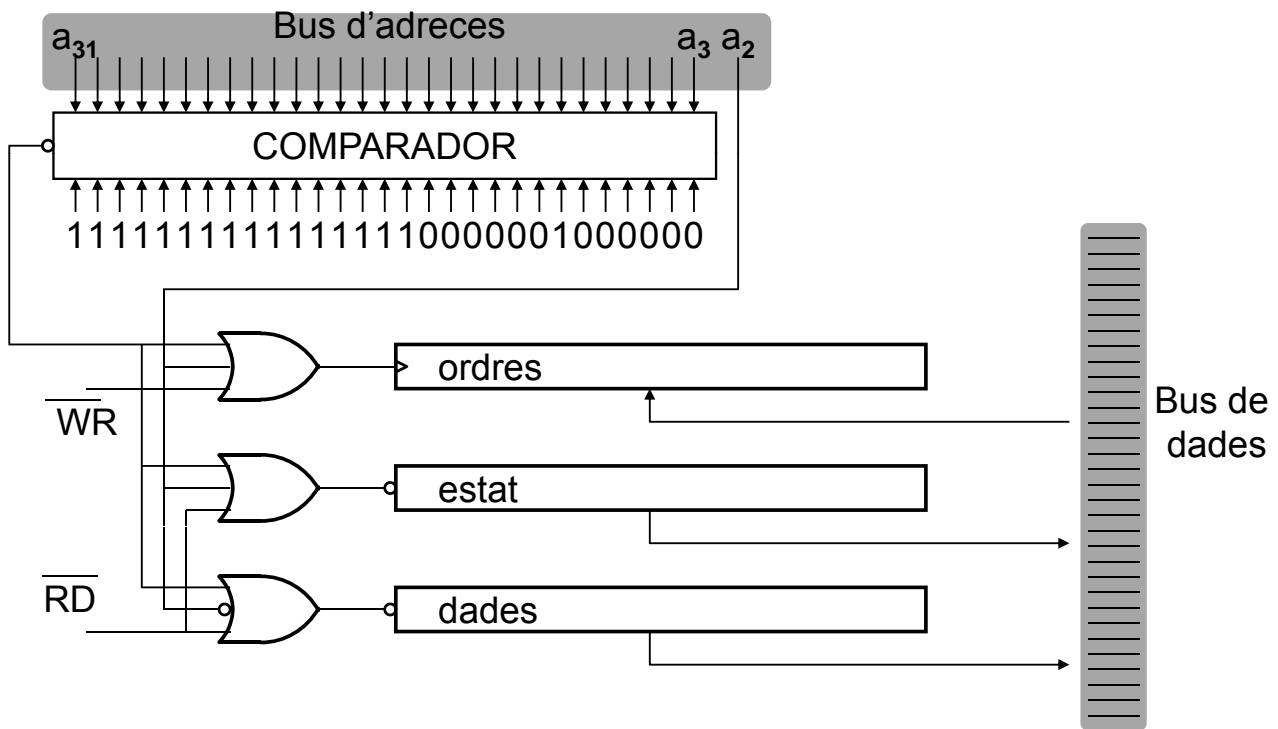
Selecció de la interfície (exemple 1)



18

Selecció

Selecció de la interfície (exemple 2)



Mecanismes de sincronització

Sincronització

▪ Conceptes

- L'atenció a una interfície ha d'ajustar-se al ritme de treball del perifèric.
 - Un teclat mereix atenció cada vegada que es prem una tecla
 - Una impressora necessita temps per a processar un caràcter (o un bloc de caràcters) abans d'acceptar-ne de nous
- Un perifèric està preparat (ready) quan pot transferir dades
 - En un perifèric d'entrada: quan hi ha noves dades per transferir
 - En un perifèric d'eixida: quan ha processat la transferència anterior
 - Hi ha perifèrics que sempre estan preparats (Exemple: el visualitzador)
- La sincronització és l'operació que permet tractar un perifèric quan està preparat

21

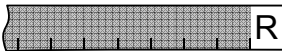
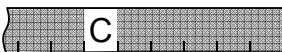

Elements de la interfície per sincronitzar

- Bit de preparat
 - sol aparèixer en el registre d'estat, amb el nom R, Ready, etc.
 - s'activa (per exemple, val R=1) quan el dispositiu està preparat
- Cancel·lació (reinici, reset...)
 - explícita: hi ha un bit en el registre d'ordres (CL, Clear, etc) que desactiva el bit de preparat
 - automàtica: el controlador desactiva el bit de preparat quan un programa accedeix a qualsevol registre de la interfície (o a algun en concret)

22

Elements de la interfície per sincronitzar

Ampliació de la interfície 2: el termòmetre

Nom	Adreça	Accés	Estructura
Estat	AB	Lect.	 R (bit 0) val 1 quan varia la temperatura
Ordres	AB	Escr.	 C (bit 5): hi escriurem un 1 per a fer R = 0 escriure C=0 no té cap efecte
Dades	AB+4	Lect.	 T (bits 7...0): temperatura

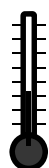


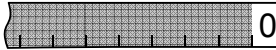
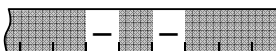
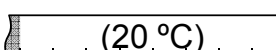
23

Elements de la interfície per sincronitzar

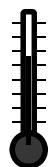
Ampliació de la interfície 2: el termòmetre

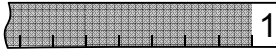
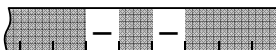
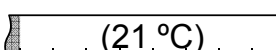
Estat inicial:
la temperatura
és de 20 °C



Estat	
Ordres	
Dades	

Perifèric preparat:
la temperatura
varia a 21 °C

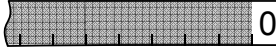
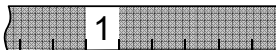
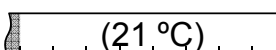


Estat	
Ordres	
Dades	

Ordre de
cancel·lació

1a \$t0,0xFFFF0010
1i \$t1,0x20
sb \$t1,0(\$t0)



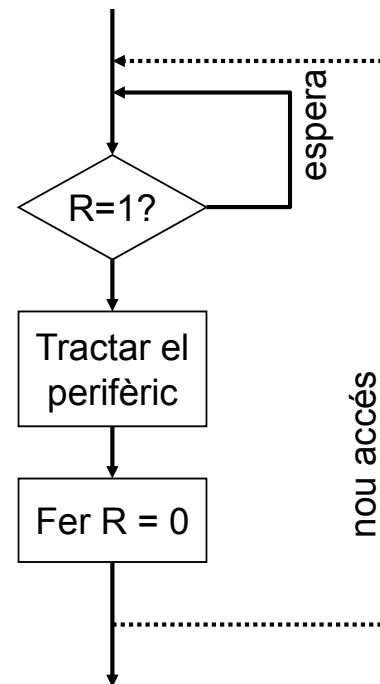
Estat	
Ordres	
Dades	

24

Sincronització per consulta d'estat

▪ Esquema general

- El programa fa un bucle d'espera fins que el bit R de preparat s'active
- Durant el tractament, ha de cancel·lar el bit de preparat (o siga, fer el bit R = 0) perquè funcione la següent consulta d'estat



25

Sincronització per consulta d'estat

▪ Operacions bàsiques

- Adreçament: accés a les adreces concretes de la interfície
- Sincronització: procés d'espera a que el perifèric estiga preparat
- Transferència de dades
- Exemple: lectura d'una temperatura nova:

	<code>la \$t0,0xFFFF0010</code>	Preparació de l'adreçament
CdE:	<code>lb \$t1,0(\$t0)</code>	} Sincronització
	<code>andi \$t1,\$t1,1</code>	
	<code>beqz \$t1,CdE</code>	
	<code>lb \$t2,4(\$t0)</code>	Transferència
	<code>li \$t1,0x20</code>	} Cancel·lació
	<code>sb \$t1,0(\$t0)</code>	

26

Sincronització per consulta d'estat

- Exemple de tractament (termòmetre + visualitzador)

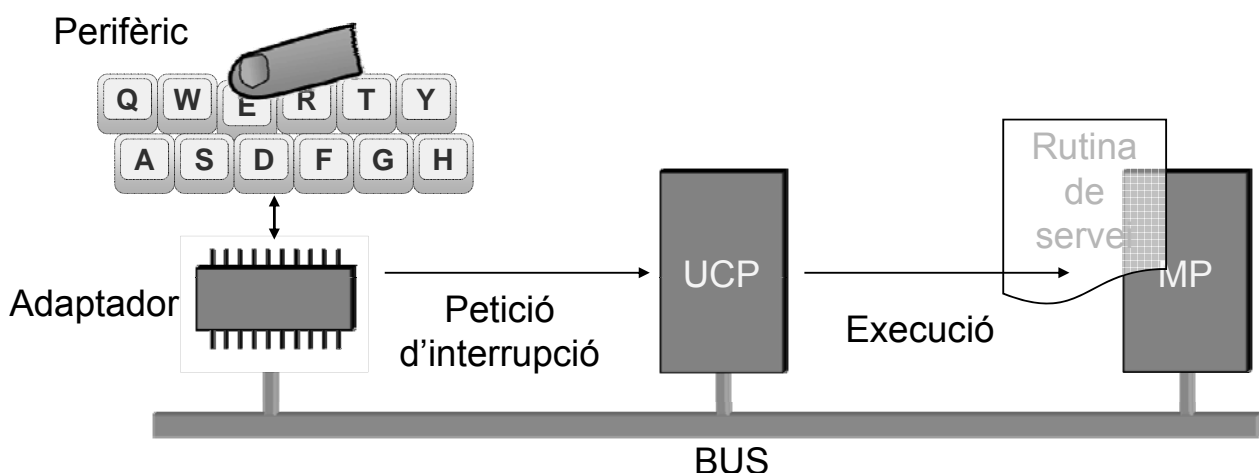
- Si temperatura < 100 °C: visualitzador apagat
- Si $100\text{ °C} \leq$ temperatura: el visualitzador mostra la lletra “E”

```
inici:  la $t0,0xFFFF0010 # TERMÒMETRE
CdE:    lb $t1,0($t0)      # Espera
        andi $t1,$t1,1    # R = 1
        beqz $t1,CdE
        li $t1,0x20
        sb $t1,0($t0)     # Cancel·la
        lb $t2,4($t0)     # Llig temp.
        la $t0,0xFFFF0020 # VISUALITZ.
        li $t3,100        # Si temp<100
        bge $t2,$t3,alarma
        sb $0,0($t0)      # ON = 0
        j inici           # si no
alarma: li $t1,0x79       # "E"
        sw $t1,4($t0)
        li $t1,1          # ON = 1
        sw $t1,0($t0)
        j inici
```

27

Sincronització per interrupció

- La petició d'interrupció
 - És un senyal elèctric que envia l'adaptador de perifèric al processador quan està preparat
 - La resposta del processador és executar un cert fragment de codi anomenat “rutina de servei” o “manejador”

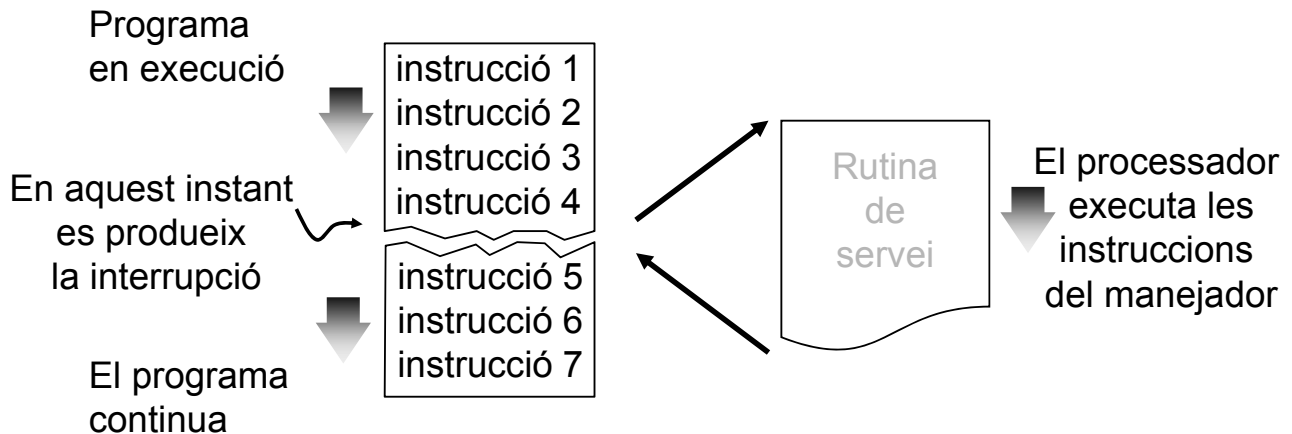


28

Sincronització per interrupció

■ Programes i interrupcions

- Les interrupcions tenen sentit especial quan el processador executa processos concurrents sota un sistema operatiu modern
- Com que l'instant en què arriba el senyal del perifèric és imprevisible, el processador ha d'interrompre el programa que estava en execució (potser sense cap relació amb el perifèric) de forma que el pugui reprendre en acabar el manejador



29

Suport a les interrupcions

■ Les interrupcions en el processador

- Els processadors donen suport a les interrupcions dins d'un esquema més general d'esdeveniment: **les excepcions**
- El circuit de control i el joc d'instruccions permeten la detenció dels programes i la seua represa posterior
- La gestió de les excepcions demanen la presència de registres de control addicionals dins del processador
- El sistema d'excepcions dona suport als sistemes operatius

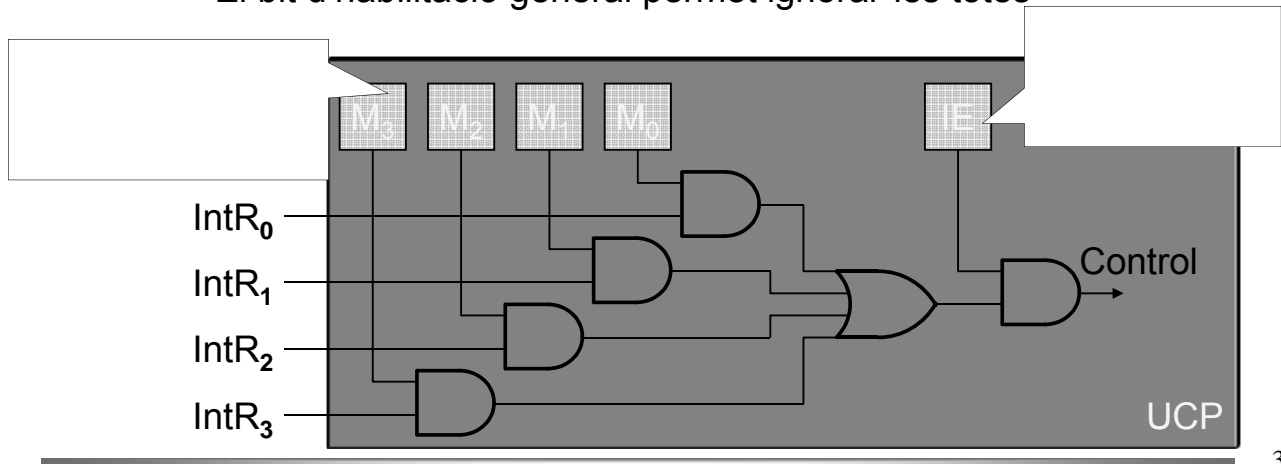
■ Les interrupcions en l'adaptador

- Perquè un perifèric admeti sincronització per interrupcions, ha de disposar de la lògica adient
- La interfície de l'adaptador inclou bits i registres de configuració del mecanisme d'interrupció

30

Suport a les interrupcions

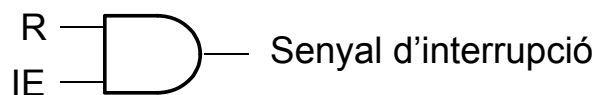
- Les interrupcions en el processador
 - El patillatge del processador té una sèrie de entrades d'interrupció
 - El processador pot ignorar totes o algunes de les entrades
 - Hi ha diversos bits accessibles als programes:
 - Els bits de màscara permeten emascarar (= ignorar) cada entrada
 - El bit d'habilitació general permet ignorar-les totes



31

Suport a les interrupcions

- Interfície dels perifèrics: l'habilitació d'interrupció
 - És un bit que sol estar present en el registre d'ordres de la interfície del perifèric. Nom típic: E o IE
 - Permet habilitar (enable) o inhabilitar (disable) l'enviament del senyal d'interrupció
 - Funcionament més freqüent:
 - Quan les interrupcions estan habilitades, la senyal d'interrupció s'activa si $R=1$
 - Quan les interrupcions estan inhabilitades, el senyal d'interrupció està sempre inactiu



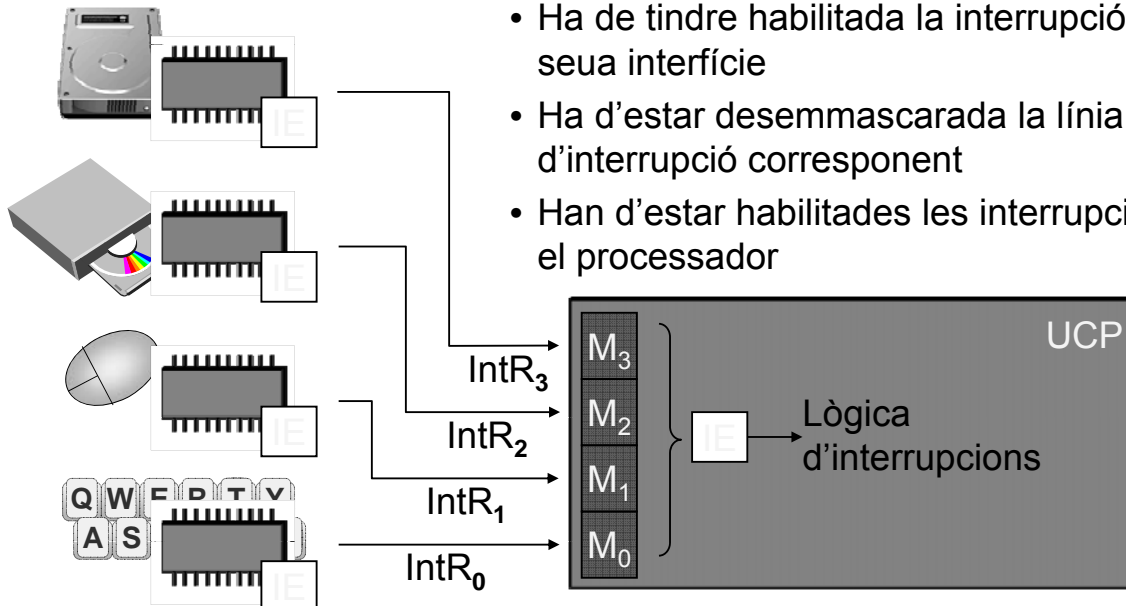
32

Suport a les interrupcions

El camí de les interrupcions

Perquè un perifèric preparat provoqe una interrupció

- Ha de tindre habilitada la interrupció en la seua interfície
- Ha d'estar desemmascarada la línia d'interrupció corresponent
- Han d'estar habilitades les interrupcions en el processador



33

Sincronització per interrupció

- Ampliació de la interfície 2: el termòmetre
– Connectat a la línia IntR₀

Nom	Adreça	Accés	Estructura
Estat	AB	Lect.	<p>R (bit 0) val 1 quan varia la temperatura</p>
Ordres	AB	Escr.	<p>E (bit 3): hi escriurem un 1 per a habilitar les interrupcions i un 0 per a deshabilitar-les C (bit 5): hi escriurem un 1 per a fer R = 0 escriure C=0 no té cap efecte</p>
Dades	AB+4	Lect.	<p>T (bits 7...0): temperatura</p>

34

Sincronització per interrupció

- Inicialització
 - Cal associar la rutina de servei RdS a la línia IntR₀ (ja ho veurem més endavant)
 - Cal habilitar i desemmascarar el camí de la interrupció IntR₀.
- Tractament
 - Ara no cal escriure codi per a sincronitzar
 - El tractament acaba amb un retorn al programa interromput (ho veurem)

RdS:	la \$t0,0xFFFF0010	} Preparació de l'adreçament Transferència Cancel·lació
	lb \$t2,4(\$t0)	
	li \$t1,0x20	
	sb \$t1,0(\$t0)	
	j tornar	

35

Sincronització per interrupció

- Exemple de tractament (termòmetre + visualitzador)

- Si la temperatura < 100 °C:
visualitzador apagat
- Si 100 °C ≤ temperatura: lletra "E" contínua
- Treball previ: les interrupcions estan habilitades en el processador
- Habilitació inicial

```
RS:    la $t0,0xFFFF0010
        li $t1,0x28
        sb $t1,0($t0)
        lb $t2,4($t0)
        li $t3,100
        la $t0,0xFFFF0028
        bge $t2,$t3,alarma
        sb $0,0($t0)
        j fin
alarma: li $t1,0x79 # "E"
        sw $t1,4($t0)
        li $t1,1
        sw $t1,0($t0)
        j tornar
```

36

Conclusió: esquemes de sincronització

- Per consulta d'estat
 - Configuració del perifèric
 - Interrupcions inhibides
 - Bucle:
 - Llegir estat
 - iterar mentres no preparat
 - Tractament
 - Cancel·lació explícita o automàtica
- Per interrupció
 - Configuració del perifèric
 - Interrupcions habilitades
 - No hi ha bucle
 - Cal codificar només el tractament
 - Cancel·lació explícita o automàtica

Interrupcions i excepcions

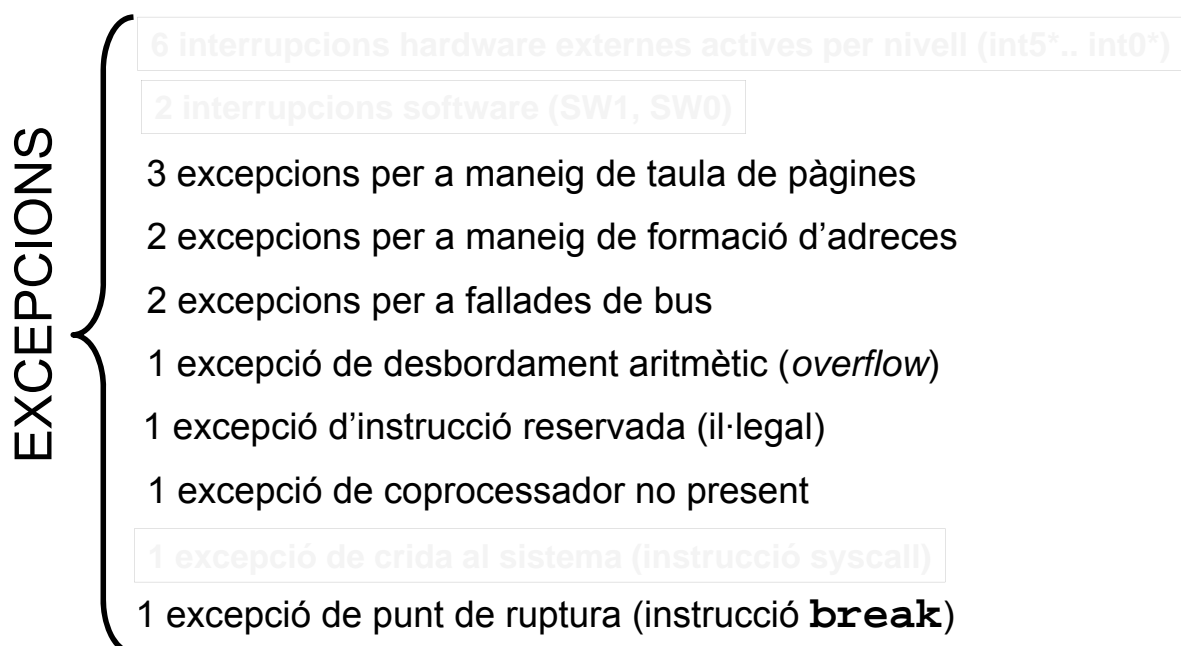
Les excepcions

▪ Definició

- Una excepció és un mecanisme que, davant d'una situació especial, canvia el flux normal d'execució
- Fonts d'excepció
 - Les línies d'interrupció de perifèric
 - Dins de la UCP: la UAL, la lògica d'accés al bus, la TLB
 - Certes instruccions dels programes
- Aplicacions diverses
 - Sincronització amb els perifèrics
 - Tractament d'errors dels programes en execució
 - Suport a la memòria virtual
 - Execució controlada de programes
 - Implementació eficient de funcions del sistema operatiu

39

Tipus d'excepcions en el MIPS R2000



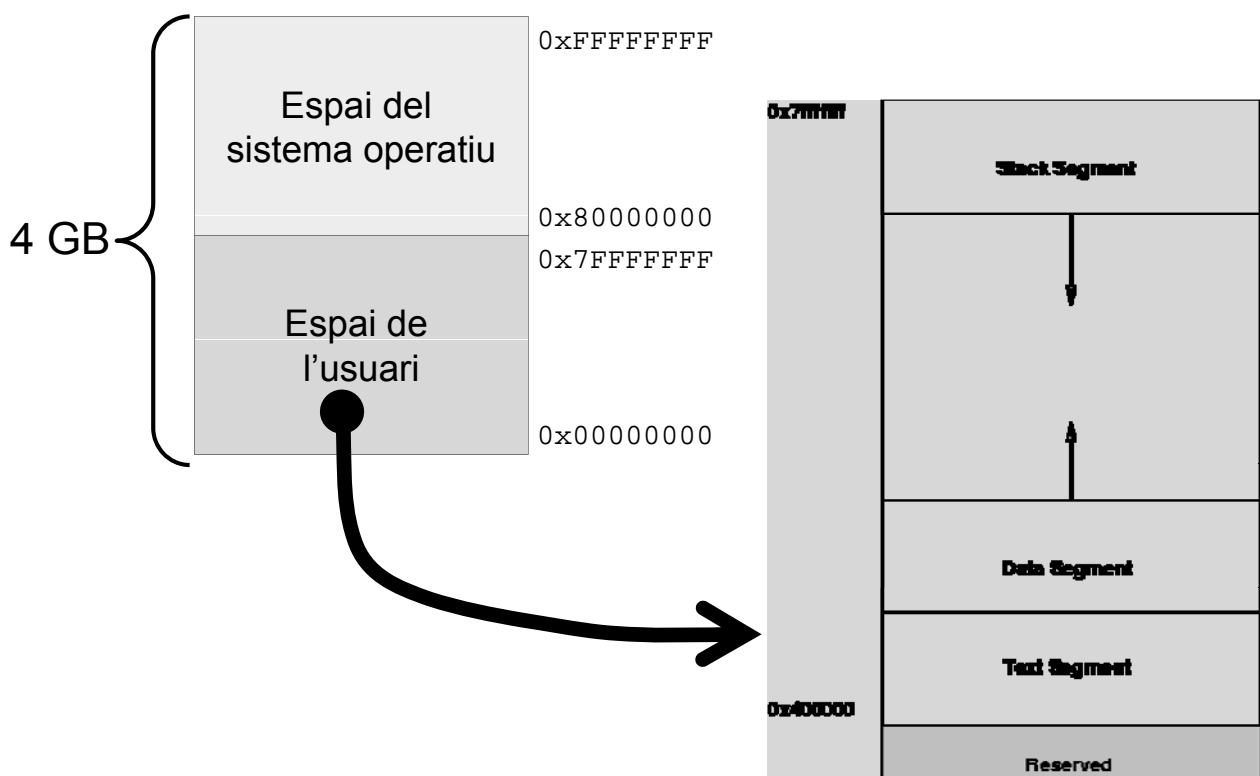
40

Modes d'operació en el MIPS R2000

- El processador té dos modes d'operació
 - Mode usuari (*user mode*)
 - Mode supervisor (*kernel mode*)
- Cal estar en mode supervisor per a:
 - Accedir a un conjunt addicional de registres
 - Executar un conjunt addicional d'instruccions
 - Accedir a tot l'espai de memòria
 - Mode usuari: de 0x00000000 a 0x7fffffff
 - Mode supervisor: de 0x00000000 a 0xffffffff
- Ús més freqüent
 - Els programes corrents funcionen en mode usuari
 - El nucli del sistema operatiu i els programes especialment habilitats funcionen en mode supervisor

41

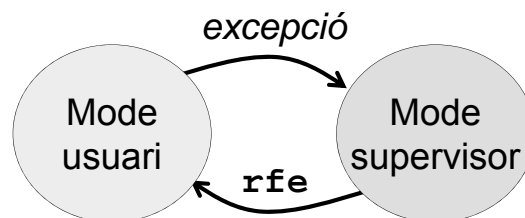
Espai de memòria del MIPS R2000



42

El canvi de mode

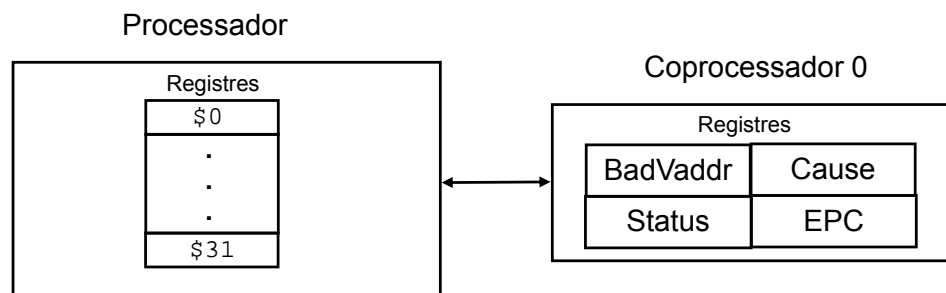
- De mode usuari a supervisor
 - És la reacció del processador quan es produeix l'excepció
 - les interrupcions queden inhibides
 - canvia a mode supervisor
 - comença a llegir instruccions en l'adreça 0x80000080
- De mode supervisor a usuari
 - Cal executar la instrucció `rfe` (*restore from exception*)
 - aquesta instrucció privilegiada només es pot executar en mode supervisor



43

Coprocessador d'excepcions

- Per tal de suportar els dos modes, l'arquitectura MIPS inclou el coprocessador CP0
 - Només és accessible en mode supervisor
 - Conté els registres necessaris per al maneig i control de les excepcions: tipus d'excepció, adreça on es produeix i altres detalls rellevants



44

Registres de maneig d'excepcions (en CP0)

Núm.	Nom	Descripció
8	Bad virtual address	(si escau) Adreça virtual que ha generat la fallada de pàgina
12	Status	Màscara i habilitació d'interrupcions
13	Cause	Tipus d'excepcions i interrupcions pendents
14	EPC	Adreça de la instrucció (PC) on es produeix l'excepció.

Instruccions d'accés als registres del coprocessador 0

Move from C0: `mfc0 Rgeneral, Rcopro` $R_{\text{general}} \leftarrow R_{\text{copro}}$

Move to C0: `mtc0 Rgeneral, Rcopro` $R_{\text{copro}} \leftarrow R_{\text{general}}$

Exemple: lectura del registre de causa en \$t0: `mfc0 $t0, $13`

45

Registre d'estat (*Status Register, SR*)

Bits de màscara d'interrupció (*Interrupt Mask IM*)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM ₇	IM ₆	IM ₅	IM ₄	IM ₃	IM ₂	IM ₁	IM ₀			KU _O	IE _O	KU _P	IE _P	KU _C	IE _C

Software Hardware
 IM₇ per a SW₁ IM_i per a la interrupció int_i*
 IM₆ per a SW₀ 0: emmascarada (inhibida)
 1: desemmascarada (habilitada)

Bits de mode i habilitació

Antic Previ Actual
 KU = mode kernel/user
 0: mode supervisor
 1: mode usuari

IE = Habilitació d'interrupcions
 (*Interrupt Enable*)
 0: deshabilitades
 1: habilitades

46

Registre de causa (*Cause Register, CR*)

Interrupcions pendents, 1: pendent								Codi d'excepció							
IP ₅	IP ₄	IP ₃	IP ₂	IP ₁	IP ₀	SW ₁	SW ₀			EC ₃	EC ₂	EC ₁	EC ₀		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Codi	Nom	Motiu que provoca l'excepció
0	INT	Interrupció (hardware o software)
1	TLBPF	(TLB) Intent d'escriptura en pàgina protegida
2	TLBML	(TLB) Intent de lectura d'instrucció en pàgina invàlida
3	TLBMS	(TLB) Intent de lectura de dades en pàgina invàlida
4	ADDRL	Error d'adreça en lectura
5	ADDRS	Error d'adreça en escriptura
6	IBUS	Error en el bus durant la cerca d'instrucció
7	DBUS	Error en el bus durant lectura o escriptura de dades
8	SYSCALL	Execució d'instrucció de crida al sistema operatiu
9	BKPT	Execució d'instrucció de ruptura de flux d'execució
10	RI	Execució d'instrucció reservada (il·legal)
11	CU	Coprocessador no utilitzable
12	OVF	Desbordament aritmètic

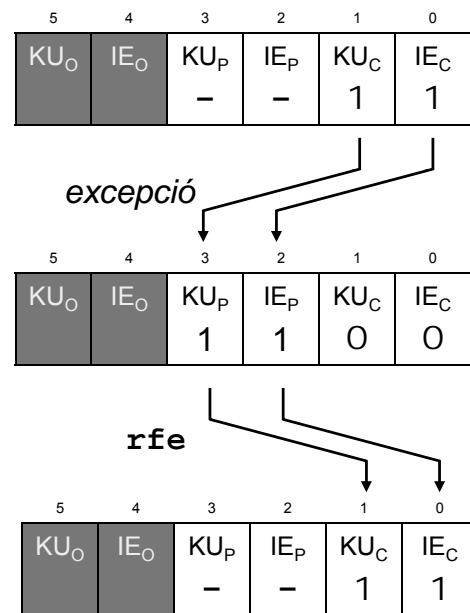
47

Adreça d'excepció (*Exception PC, EPC*)

- Registre que conté el valor del PC quan es produeix l'excepció
 - és a dir, conté l'adreça de la instrucció afectada
 - aquesta instrucció no ha pogut acabar la seua execució
- L'EPC apunta, segons el tipus d'excepció, a:
 - *Fallades de pàgina*: la instrucció (de qualsevol tipus) que no s'ha pogut llegir o la instrucció load o store que ha generat l'accés.
 - *Errors aritmètics, de bus, etc*: la instrucció causant
 - *Interrupcions*: la instrucció que s'haguera executat si no fóra per que ha vingut la interrupció

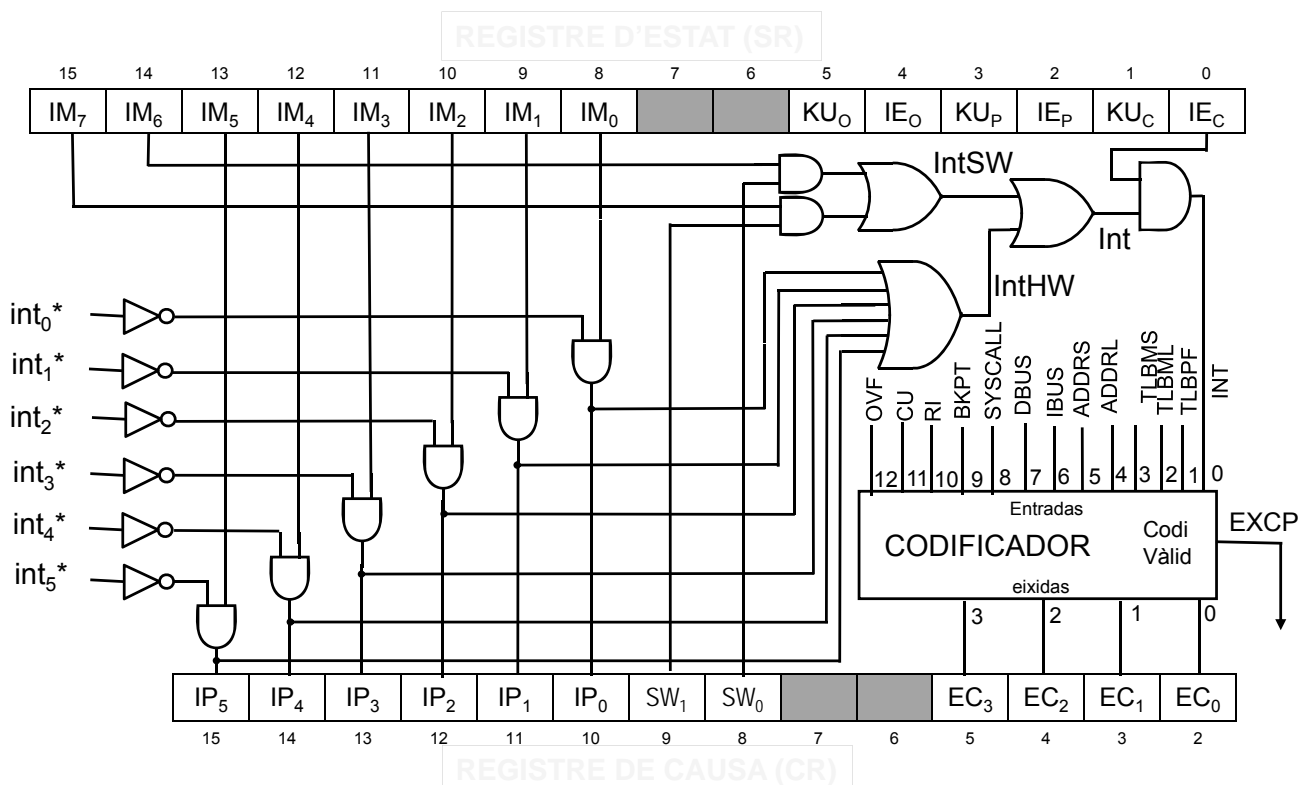
El canvi de mode en CP0

- Programa d'usuari en execució
 - Interrupcions habilitades
 - Mode usuari
- S'alça l'excepció
 - Bits de mode i habilitació = 0
 - EPC = valor del PC
 - CR i BadVaddr s'actualitzen
 - PC = adreça d'inici del manejador
- Execució del manejador
 - Interrupcions inhibides
 - Mode supervisor
- Retorn (**rfe**)
 - (torna a l'estat inicial)



49

Hardware associat a les excepcions

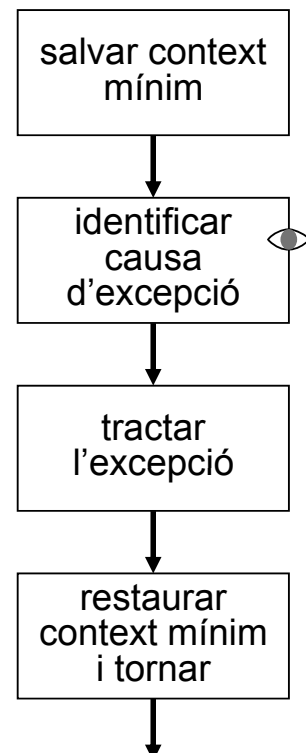


50

Maneig d'excepcions

Disseny del manejador

- Gestió del context
 - El manejador s'ha de poder executar de manera que no afecte el programa
- Identificació de causa i tractament
 - Qualsevol causa d'excepció provoca l'execució del manejador
- Atomicitat
 - L'execució del manejador no es pot interrompre
 - El codi ha de ser correcte (per tal que no provoqui excepcions)
 - Les interrupcions estan inhibides

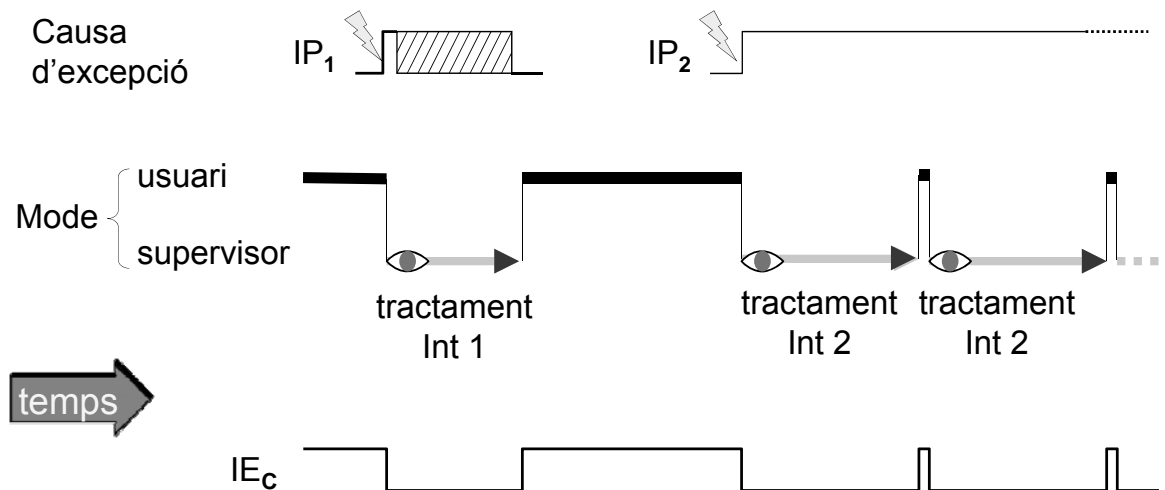


Maneig d'interrupcions

▪ Cronograma exemple 1:

– El tractament que es dona al perifèric ha de garantir que aquest cancel·le la petició d'interrupció.

- Exemple: tractament Int 1 correcte, Int 2 incorrecte

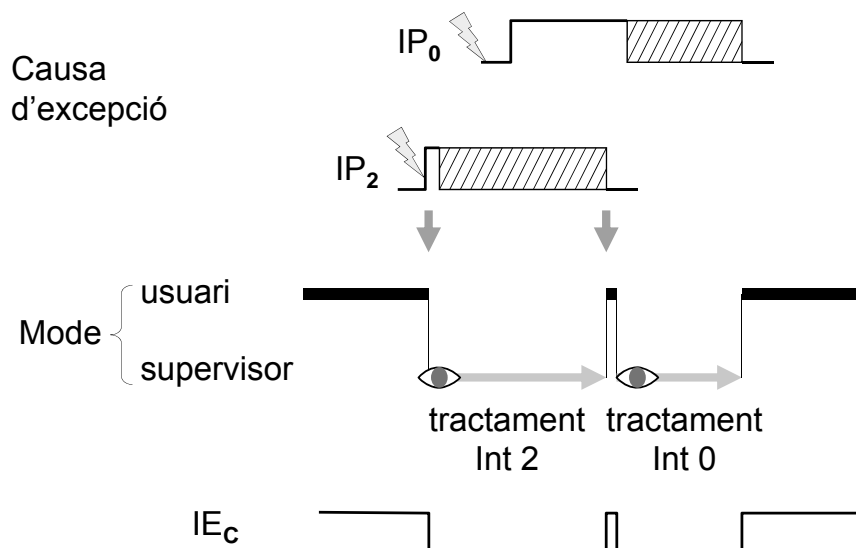


53

Interrupcions durant l'execució del manejador

▪ Cronograma exemple2:

– Mentre s'atén la interrupció 2, hi arriba la interrupció 0



54

Variables del manejador

- Necessitats:
 - Espai per a guardar temporalment el context mínim
 - De moment, és prou amb *\$at*, *\$t0* i *\$t1*
 - Espai per a ubicar l'adreça de retorn
 - Variables que calguen per a cada tractament
- Registres dedicats
 - Els registres *\$k0* i *\$k1* estan dedicats al sistema operatiu i no formen part del context del programa
 - *\$k0* actuarà com a registre temporal del manejador i contindrà l'adreça de retorn en l'últim moment
 - *\$k1* contindrà l'adreça base del context mínim

```
.kdata
## espai per a context mínim
salvareg: .word 0,0,0
adretorn: .word 0
```

55

Gestió del context mínim

```
.ktext 0x80000080 # Punt d'entrada al manejador
sw $at,0($k1)      # Salve $at
sw $t0, 4($k1)     # $t0 contindrà adreces
sw $t1, 8($k1)     # $t1 contindrà dades
mfc0 $k0, $14      # llig EPC
sw $k0,adretorn    # guarde adreça de retorn
```

identificar
causa
d'excepció

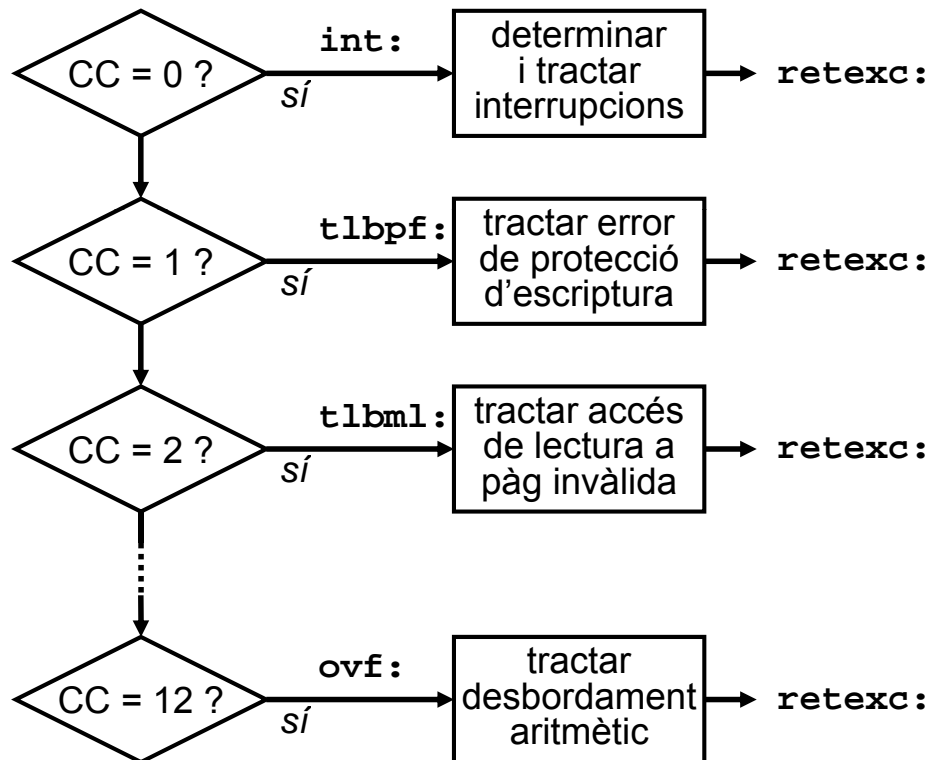
tractar
l'excepció

```
retexc: lw $k0, adretorn # adreça de retorn en $k0
        lw $at, 0($k1)  # restaure $at
        lw $t0, 4($k1)  # restaure $t0
        lw $t1, 8($k1)  # restaure $t1
        rfe              # passe a mode usuari
        jr $k0           # torne al programa
```

Considerarem que les dues últimes instruccions (*rfe* i *jr*) formen un bloc indivisible. Açò evitarà el processament d'una petició d'interrupció al finalitzar *rfe*. Ja discutirem en el tema 6 la manera de fer possible açò

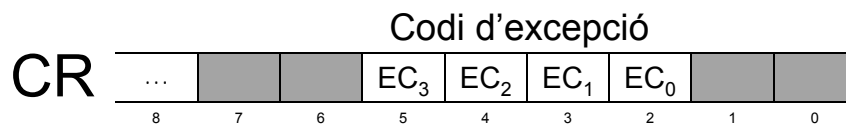
56

Anàlisi de causa d'excepció: flux



57

Anàlisi de causa d'excepció: ensamblador

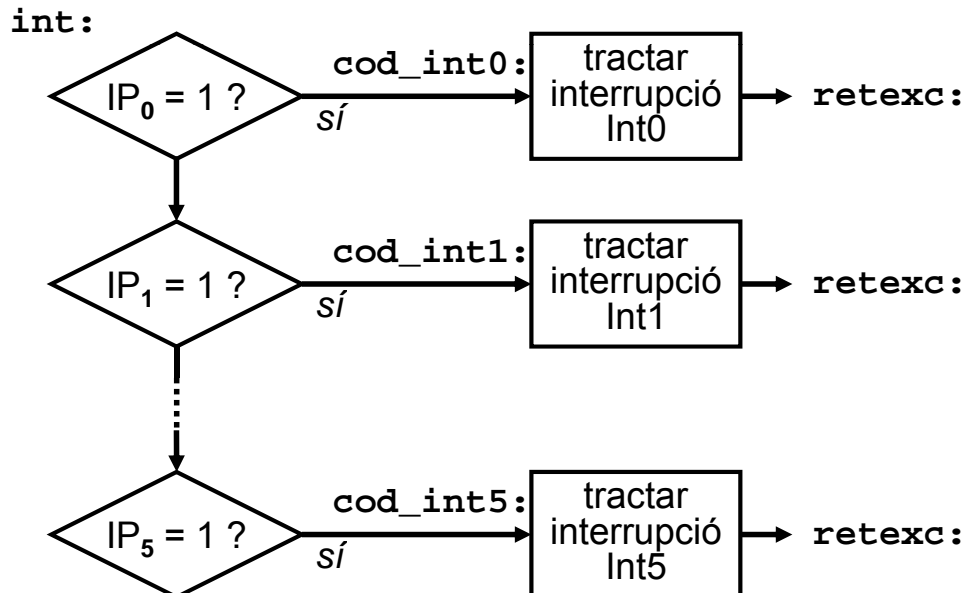


```
mfc0 $k0, $13          # Llig registre de Causa
andi $t0, $k0, 0x003c  # Aïlla el codi*4
beq $t0, $zero, int     # Compara amb 0 i salta
li $t1, 4               # Codi 1*4
beq $t0, $t1, tlbfp     # compara i salta si cal
li $t1, 8               # Codi 2*4
beq $t0, $t1, tlbml     # compara i salta si cal
...
li $t1, 0x30            # Codi 12*4
beq $t0, $t1, ovf       # compara i salta si cal
```

58

Anàlisi de causa d'interrupció: flux

- Pot haver més d'una interrupció pendent
 - L'ordre de consulta és rellevant



59

Anàlisi de causa d'interrupció: ensamblador

```
int:
    andi $t0, $k0, 0x400    # mire IP0
    bne  $t0, $zero, cod_int0
    andi $t0, $k0, 0x800    # mire IP1
    bne  $t0, $zero, cod_int1
    ...
    andi $t0, $k0, 0x8000   # mire IP5
    bne  $t0, $zero, cod_int5
    b retexc                # interrupció espúrea

cod_int0:  ### codi de tractament d'interrupció int0*
    ...
    b retexc ### fi del codi de tractament d'int0*

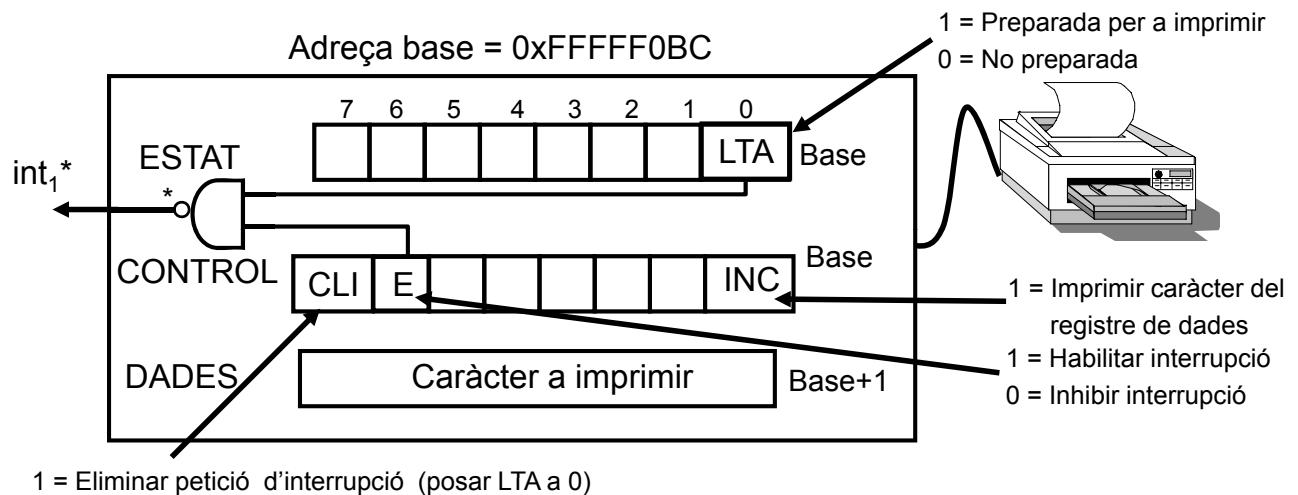
cod_int1:  ### codi de tractament d'interrupció int1*
    ...
    b retexc ### fi del codi de tractament d'int1*

cod_int5:  ### codi de tractament d'interrupció int5*
    ...
    b retexc ### fi del codi de tractament d'int5*
```



60

Exemple: tractament d'una interrupció



```
cod_int1: ### codi de tractament de la int. de impressora
### arribar fins ací suposa que IEC=IM1=E=1 (int. habilitada)
li    $t0,0xFFFFF0BC
lb    $t1,car_A      # llig caràcter de memòria
sb    $t1,1($t0)     # Escriu registre de Dades
li    $t1,0xC1       # col.loca a 1 els bits 0, 6 i 7
sb    $t1,0($t0)     # Escriu R. Control (CLI=E=INC=1)
b     retint        ### fi de codi de tractament
```

61

5. Entrada/eixida mitjançant el sistema operatiu

62

Excepcions i sistema operatiu

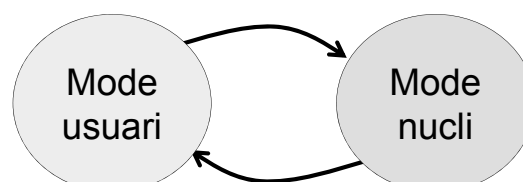
- Les excepcions permeten resoldre de forma homogènia moltes necessitats dels sistemes operatius
 - El hardware gestiona de forma simple i segura els modes d'execució d'usuari i supervisor
 - Són una forma general d'esdeveniment que provoca l'atenció del sistema operatiu
 - Simplifiquen el canvi de context durant la commutació de processos
 - Implementen el canvi d'estat dels processos (actiu, en espera, etc.)

63

Excepcions i sistema operatiu

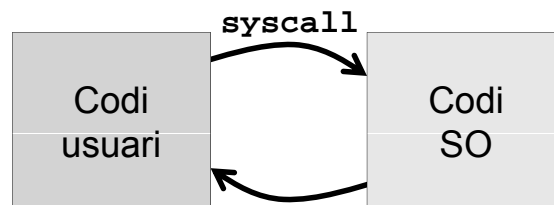
Recordatori SO1

- Organització d'un sistema operatiu
 - El sistema operatiu està format per diversos fragments de codi que gestionen el processador, la memòria i l'entrada/eixida
 - Els programes corrents s'executen en mode usuari
 - Un conjunt d'esdeveniments provoquen l'execució de fragments de codi del sistema operatiu en mode supervisor
- El sistema operatiu gestiona l'entrada/eixida
 - Les interfícies dels perifèrics es mapejen en adreces restringides i només són accessibles en mode supervisor
 - Les interrupcions provoquen el canvi al mode supervisor i l'execució del manejador inclòs en el sistema operatiu



64

- Independència entre els programes d'usuari i els del sistema operatiu
 - Els programes d'usuari han d'executar-se sota diferents configuracions d'un computador i amb diferents versions del sistema operatiu
- Les crides al sistema permeten enllaçar de forma segura els programes d'usuari i el codi del sistema operatiu
 - Els programes provoquen l'execució de codi del sistema mitjançant instruccions de **crida al sistema**
 - Així, les crides al sistema són tractades per manejadors, igual que la resta de causes (interrupcions, fallades de memòria virtual, errors, etc.)

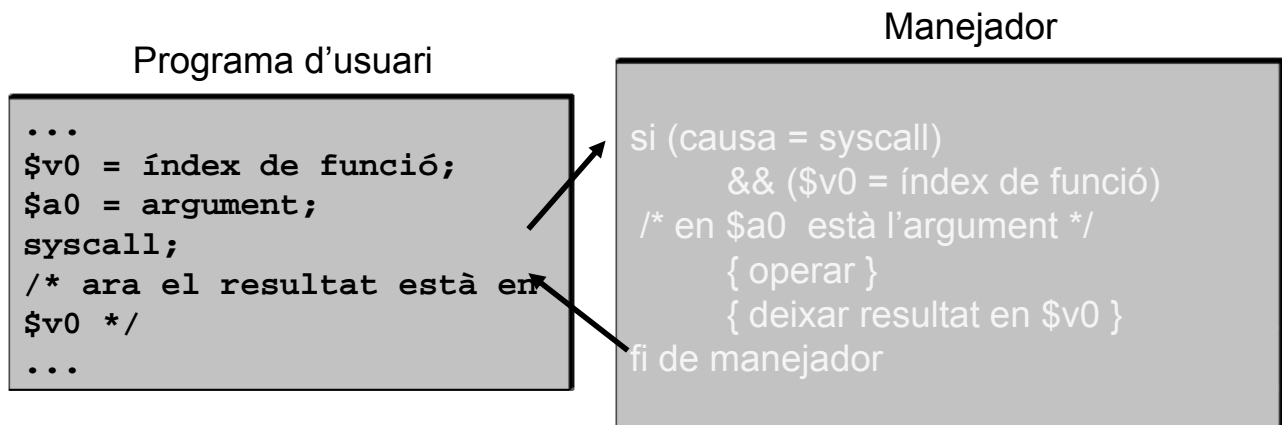


Les funcions del sistema operatiu

- Mecanisme
 - Per a cridar una funció, els programes han de construir uns paràmetres (o arguments) i executar una instrucció de crida al sistema **syscall**
 - Quan el programa recupere el control, trobarà els resultats escaients
 - En cada sistema operatiu s'especifica el conjunt de funcions disponibles i per a cadascuna d'elles els paràmetres adients i els resultats
- Paràmetres i resultats
 - els petits (una paraula o menys) se solen transmetre mitjançant els registres del processador
 - els grans (p. ex. una cadena de caràcters) se solen transmetre mitjançant un àrea de la memòria i un registre amb un punter

Crides al sistema en el MIPS R2000

- La instrucció **syscall** provoca l'excepció amb el codi 8 en el registre de causa (CR)
 - Per conveni, farem servir el registre **\$v0** com a índex que identifica la funció del sistema que se sol·licita
- Exemple amb una funció que demana un argument en **\$a0** i deixa el resultat en **\$v0**



67

Funcions del sistema implementades en PCSpim

Funció	Codi	Arguments	Resultat
<code>print_int</code>	<code>\$v0=1</code>	<code>\$a0 = enter</code>	
<code>print_float</code>	<code>\$v0=2</code>	<code>\$f12 = coma flotant</code>	
<code>print_double</code>	<code>\$v0=3</code>	<code>\$f12 = doble precisió</code>	
<code>print_string</code>	<code>\$v0=4</code>	<code>\$a0 = punter a cadena</code>	
<code>read_int</code>	<code>\$v0=5</code>		Enter (en <code>\$v0</code>)
<code>read_float</code>	<code>\$v0=6</code>		Coma flotant (en <code>\$f0</code>)
<code>read_double</code>	<code>\$v0=7</code>		Doble precisió (en <code>\$f0</code>)
<code>read_string</code>	<code>\$v0=8</code>	<code>\$a0 = punter a cadena</code> <code>\$a1 = longitud</code>	
<code>print_char</code>	<code>\$v0=11</code>	<code>\$a0 = caràcter</code>	
<code>read_char</code>	<code>\$v0=12</code>		Caràcter (en <code>\$v0</code>)

68

Exemple de crida al sistema en PCSpim

■

programa d'usuari

```
li $v0,1          # syscall 1, print_int (a la consola)
li $a0,0x7ffe      # el valor de l'enter
syscall           # invoca a syscall
```

efecte



$$7FFE_{16} = 32766_{10}$$

69

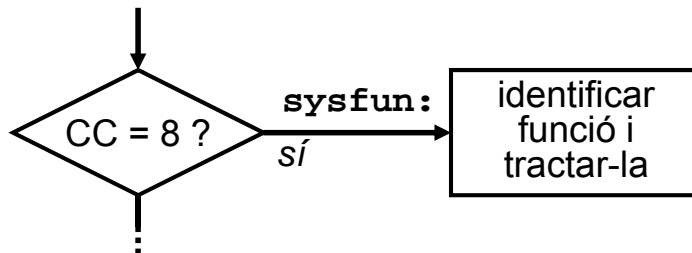
El retorn al programa

- El manejador ha de determinar quina és la primera instrucció que s'ha d'executar quan el control torne al programa detingut
 - Això ho ha de fer sempre, tant si hi ha commutació de processos com si no
- Cal tenir en compte que EPC apunta a la primera instrucció que no s'ha pogut completar
- Casos significatius:
 - En cas d'interrupció cal tornar a l'adreça on apunta EPC
 - En cas de crida al sistema, EPC apunta a la instrucció `syscall` però caldrà tornar a la següent (`EPC+4`)
 - En cas de fallada de memòria virtual, EPC apunta a la instrucció causant i cal tornar a ella perquè s'execute de nou després que el SO carregue la pàgina implicada en la memòria

70

Identificació de la crida al sistema

- En la secció d'anàlisi del codi d'excepció hi ha:
 - El codi d'excepció corresponent a `syscall` és 8

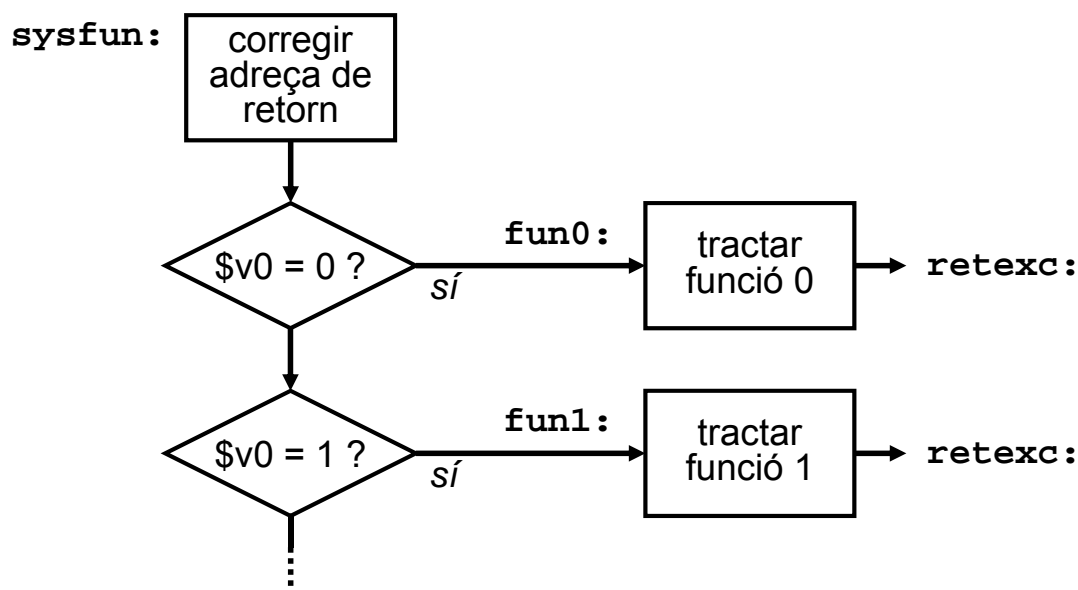


```
mfc0 $k0, $13          # Llig registre de Causa
andi $t0, $k0, 0x003c  # Aïlla el codi*4
beq $t0, $zero, int     # Compara amb 0 i salta
li $t1, 4               # Codi 1*4
beq $t0, $t1, tlbf      # compara i salta si cal
...
li $t1, 0x20            # Codi 8*4
beq $t0, $t1, sysfun    # compara i salta si cal
...
```

71

Tractament de les funcions: flux

- Cal corregir l'adreça de retorn, identificar el codi present en `$v0` i tractar cadascuna per separat



72

Tractament de les funcions: ensamblador

```
Sysfun: ### tractament de syscall
        # corregeix l'adreça de retorn
        lw $t0,adretorn          # corregeixc
        addi $t0,$t0,4           # l' adreça
        sw $t0,adretorn          # de retorn
        # salta segons l'índex present en $v0
        beq $zero, $v0, fun0
        li $t0, 1
        beq $t0, $v0, fun1
        ...
        ...
fun0: ### tractament de la crida al sistema amb índex 0
        ...
        b retexc
fun1: ### tractament de la crida al sistema amb índex 1
        ...
        b retexc
```

73

Cas 1: funcions d'accés a informació

- Especificació:
 - *P_model* torna un codi que identifica el processador
 - *Sys_ver* torna el codi de la versió del SO
- Comentaris:
 - Per a una instal·lació donada, aquests dos codis són constants i el manejador no ha de calcular res
 - El tractament no ha d'accedir a cap perifèric

Funció	Codi	Arguments	Resultat
<i>P_model</i>	<i>\$v0</i> = 9991	–	<i>\$v0</i> = codi de processador
<i>Sys_ver</i>	<i>\$v0</i> = 9992	–	<i>\$v0</i> = codi de versió del SO

74

Cas 1: implementació dels manejadors

```
Sysfun: lw $t0,adretorn    # calcule
        addi $t0,$t0,4     #      l' adreça
        sw $t0,adretorn    #      de retorn
# salta a etiqueta segons l'índex present en $v0
        li $t0, 9991
        beq $t0, $v0, P_model
        li $t0, 9992
        beq $t0, $v0, Sys_ver
        ...

P_model: li $v0, codi_model_proc
        b retexc

Sys_ver: li $v0, codi_versió
        b retexc
```

75

Cas 2: accés a la interfície d'un perifèric

- Problemàtica
 - Donat que els perifèrics no són accessibles en mode usuari, l'accés als registres de la seua interfície s'ha de fer mitjançant funcions del sistema
- Exemple amb el perifèric d'entrada/eixida directa (interruptors i leds) en l'adreça base 0xFFFFF300
 - *Set_Leds* il·lumina els quatre leds
 - *Get_Switches* consulta quins interruptors estan oberts



Funció	Codi	Arguments	Resultat
Set_Leds	\$v0 = 9980	–	–
Get_Switches	\$v0 = 9981	–	\$v0 = estat dels interruptors

76

Cas 2: detall del tractament



```
Set_Leds:
    li $t0,0xFFFFF300
    sb $zero,0($t0)
    b retexc

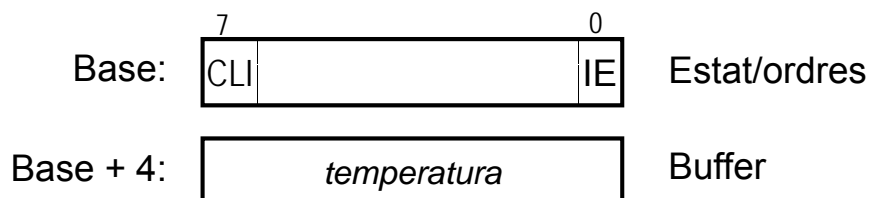
Get_Switches:
    li $t0,0xFFFFF300
    lb $v0,0($t0)
    b retexc
```

77

Cas 3: interrupcions sense espera

- Un sensor: termòmetre
 - Davant d'un canvi de temperatura, el perifèric provoca interrupcions que afecten a una variable privada del nucli del SO
 - Especificació de la interfície del perifèric

Base = 0xFFFFB9000, interrupció per la línia INT4



- Especificació de la funció del sistema

Funció	Codi	Arguments	Resultat
Get_Temp	\$v0 = 9975	–	\$v0 = temperatura

78

¿Com consulta la temperatura l'usuari?

- Fa una crida al sistema amb codi 9975
 - El valor de la temperatura obtingut el guarda en la memòria i l'imprimeix en la consola mitjançant la crida `print_int`

```
valor:    .data
          .byte 0

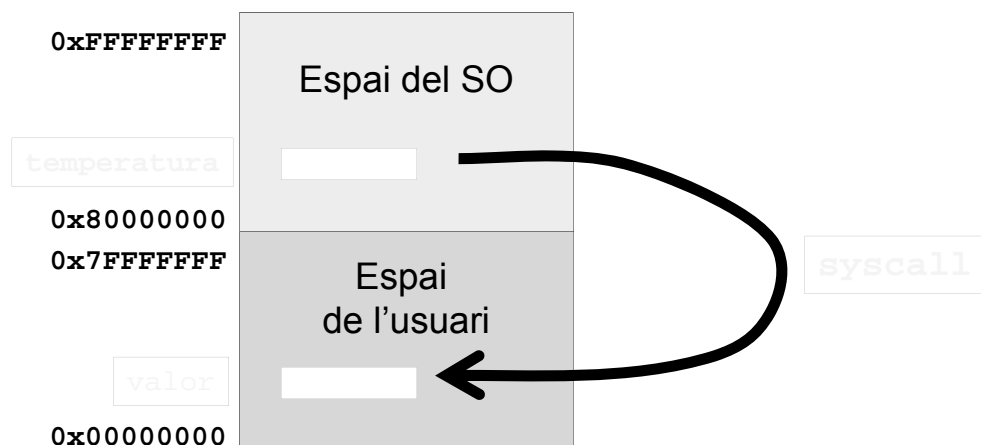
          .text
          # Llig la temperatura
          li $v0, 9975
          syscall          # La temperatura està ara en $v0
          sb $v0, valor    # L'emmagatzema en la memòria

          # Ara la imprimeix en la consola
          move $a0, $v0    # Argument en $a0
          addi $v0, $0, 1  # Codi 1 per a print_int
          syscall          # Impresió en la consola
```

79

Intercanvi entre els dos espais

- El valor de la temperatura s'ha copiat des de l'espai del SO fins el de l'usuari mitjançant la crida al sistema amb índex 9975



80

Cas 3: detall del tractament

```
        .kdata
temperatura: .byte 0          # Variable privada del SO

        .ktext
        ...

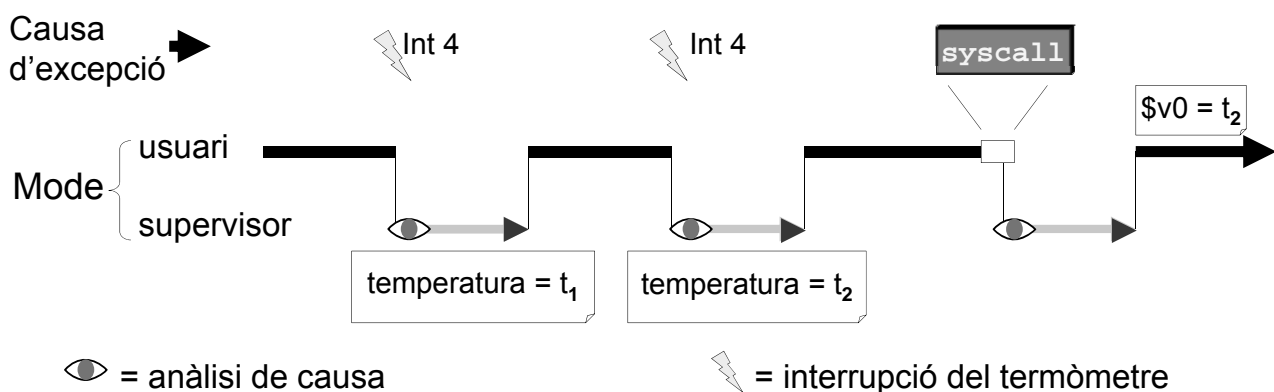
# en la secció de tractament d'interrupcions
int4:    la $t0,0xFFFFB9000
        lb $t1,4($t0)
        sb $t1,temperatura
        li $t1,0x81
        sb $t1,0($t0)    # cancel·le interrupció
        b retexc
        ...

# en la secció de crides al sistema
get_temp: lb $v0,temperatura
        b retexc
        ...
```

81

Cas 3: exemple d'evolució de les variables

- El sensor provoca interrupcions que actualitzen la variable *temperatura*
- En cridar la funció *get_temp*, el programa obté en *\$v0* la còpia de l'últim valor anotat en *temperatura*
- En el cronograma: el sensor fa dues interrupcions amb dos valors t_1 i t_2 abans que el programa faça la consulta



82

Excepcions i commutació de processos

- Les excepcions detenen el procés que hi estava en execució
- Els manejadors també fan la planificació dels processos
- Un manejador pot:
 - Reprendre el procés detingut
 - Canviar l'estat del procés detingut i commutar a un altre
- Casos significatius de canvi d'estat:
 - Els errors fatals (aritmètics, d'adreçament, etc.) acaben el procés
 - Certes funcions d'entrada/eixida i les fallades de pàgina suspenen el procés i l'envien a una cua d'espera
 - Una interrupció de perifèric pot reactivar un procés en espera (distint del detingut) i passar-lo a la cua de preparats
 - La interrupció de rellotge pot enviar el procés detingut a la cua de processos preparats quan ha expirat el quant de temps

83

El canvi de context

- El manejador conserva el context màquina del procés detingut:
 - La variable *salvareg* preserva els registres *\$at*, *\$t0* i *\$t1* del procés
 - La resta de registres de propòsit general encara conserven el seu valor
 - El manejador controla el valor que prendrà el PC en retornar
- El manejador pot canviar el context màquina d'un procés P pel context d'altre procés Q
 - Haurà de transferir el context del procés P des dels registres al BCP (bloc de control de procés) corresponent.
 - Haurà de transferir el context del procés Q des del seu BCP als registres

84

Planificació de processos

- Suposarem que hi ha un planificador de processos en el codi del manejador
 - Tres primitives:
 - fixar_context*,
 - suspen_aquest_proc*
 - activa_proc_en_espera*
 - Les primitives estan implementades com subprogrames sense paràmetres

`fixar_context:`



85

On es pot fer el canvi de context?

fixar_context

- determina quin és el procés actiu que entra en execució al final del manejador.
 - Si el procés escollit és el mateix que s'ha detingut per l'excepció, no fa cap canvi de context
 - Si el procés escollit és distint al detingut, farà el canvi complet de context (registres, adreça de retorn i d'altres dades) i gestionarà la cua de processos preparats
- el manejador el crida sempre tot just abans de canviar de mode

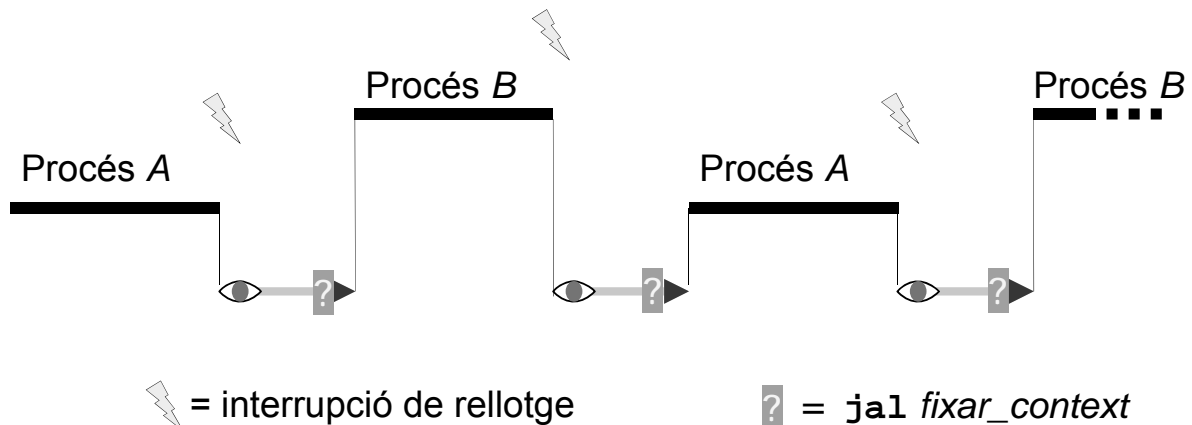
```
retexc: jal fixar_context # Possible canvi de context
        lw $k0, adretorn  # adreça de retorn en $k0
        lw $at, 0($k1)    # )
        lw $t0, 4($k1)    # ) restaure context mínim
        lw $t1, 8($k1)    # )
        rfe               # passe a mode usuari
        jr $k0            # torne al programa
```

86

Processos actius

El planificador selecciona entre els processos actius

- Exemple, amb només dos processos A i B actius.
- esquema de planificació: a cada interrupció de rellotge selecciona el LRU dels processos actius

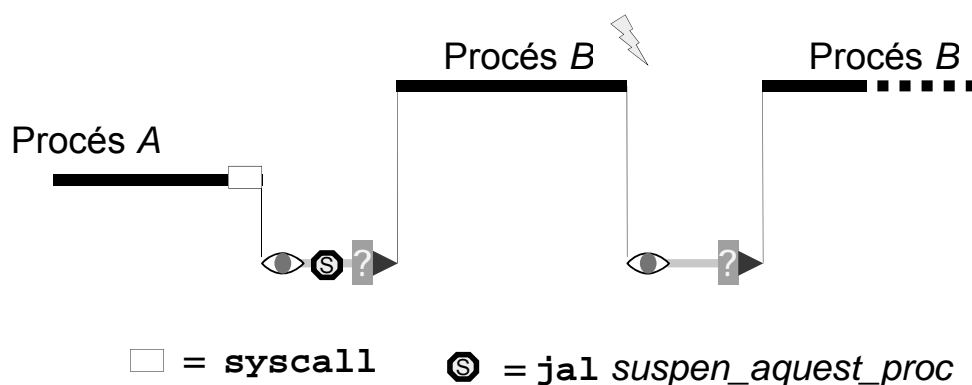


87

Suspensió d'un procés

suspen_aquest_proc

- canvia a suspés l'estat del procés interromput i insereix el seu BCP en la cua d'E/S adient
- Exemple: funció *F* que permet al procés A esperar fins que el perifèric P està preparat
 - el tractament de *F* suspén el procés A
 - mentre A no canvie d'estat, aquest procés mai no serà triat per *fixar_context*

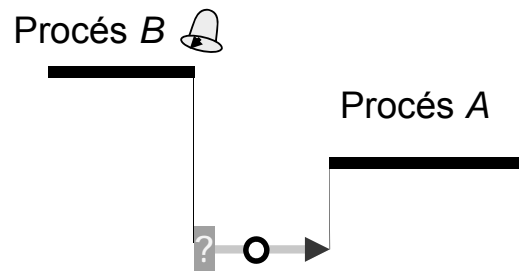


88

Reactivació d'un procés

activa_proc_en_espera

- canvia a actiu el procés que es troba esperant a un perifèric i insereix el seu BCP en la cua de processos preparats
- Continua l'exemple anterior:
 - el tractament de la interrupció del perifèric *P* haurà de reactivar el procés *A*
 - el planificador podrà triar-lo de nou i tornar-lo a execució



= interrupció del perifèric *P*

● = ja1 *activa_proc_en_espera*

89

Cas 4: Entrada amb espera

- Perifèric: el sensor que fa interrupció en *int4** en canviar la temperatura
- Funció del sistema:
 - *get_temp_wait* que permet que un programa espere fins que el sensor subministre una temperatura nova
 - El tractament ha d'executar el procediment *suspen_aquest_proces* per a suspendre el programa i deixar-lo en la cua d'espera del sensor
- Interrupció:
 - El procediment *transmet_valor* modifica el contingut de *\$v0* en el context màquina dels possibles processos (0, 1 o més) que estan esperant al sensor
 - El procediment *activa_proc_en_espera* conté el codi que fa actius els possibles processos que esperen el sensor

90

Cas 4: Implementació de la funció

- Només cal canviar l'estat del procés
 - El procés queda en estat d'espera d'una nova temperatura

```
.kdata
temperatura: .byte 0

.ktext

# funció del sistema
get_temp_wait: jal suspen_aquest_proces
               b retexc
               ...
```

- En *retexc* el planificador commutarà el context perquè el procés detingut ja no està actiu

91

Cas 4: maneig de la interrupció

- El tractament ha de transmetre el valor als processos que esperen i canviar el seu estat
 - Quan acabe el manejador, qualsevol d'aquests processos pot entrar en execució

```
.ktext
# tractament de la interrupció int4
int4:  la $t0,0xFFFFB9000      # adreça base
       lb $t1,4($t0)          # llig temperatura
       sb $t1,temperatura($0)  # guarda valor

       jal transmet_valor
       jal activa_proc_en_espera

       li $t1,0x81             # màscara per IE = 1
       sb $t1,0($t0)           # habilita interrupcions
       b retexc
```

92

6. Mecanismes de transferència

- Per programa
- Per accés directe a memòria

93

Dispositius de blocs

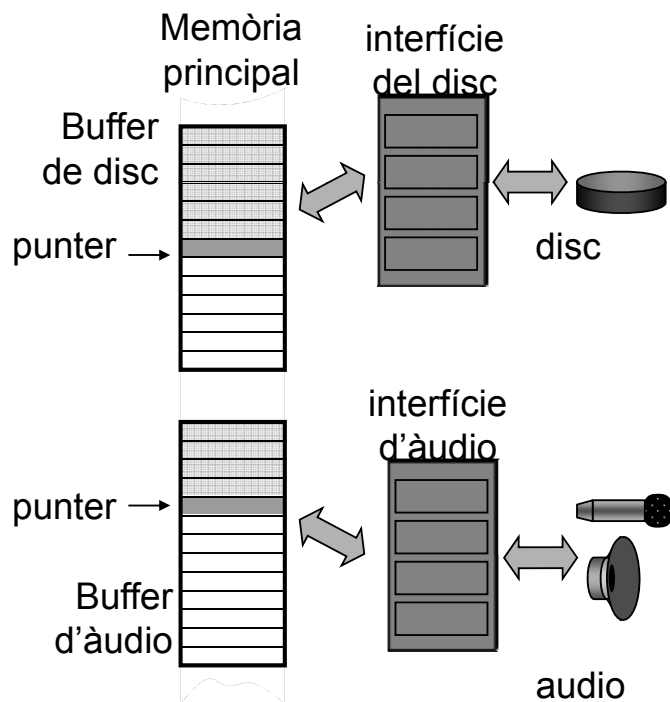
- Característiques
 - Cal transferir blocs = volums importants de dades, molt majors que la paraula del processador
 - Per transferir un bloc, cal fer moltes transferències elementals (d'un byte o d'una paraula) seguides
 - La naturalesa del perifèric imposa un mínima amplada de banda per a les transferències
 - Exemples
 - Disc: sectors de 512 bytes, a ~100 MBps
 - Targeta de xarxa: blocs a 10/100/1000 Mbps
 - Targeta de so: blocs de 512 o 1024 bytes, a ~200 KBps

94

Transferències de blocs

▪ Buffers

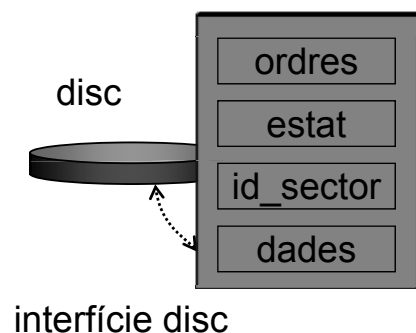
- La transferència es fa entre el dispositiu i un buffer de memòria amb capacitat d'un o més blocs
- Generalment, cada dispositiu de blocs té un buffer exclusiu
- Cada buffer té una adreça inicial i un punter que indica el punt actual de la transferència
- Fins que no es transfereix tot un bloc, no es dona per terminada l'operació d'entrada/eixida



95

Interfície exemple

- Controlador bàsic de disc
 - Permet la lectura i escriptura de sectors de fins a 512 bytes
- Interfície
 - *estat*: registre d'estat que informa de la disponibilitat del perifèric
 - *ordres*: registre d'ordres amb bits que determinen el sentit de la transferència (lectura: del controlador a la memòria o escriptura: de la memòria al controlador) i altres detalls
 - *id_sector*: coordenades del sector
 - *dades*: registre de dades (8 bits)



96

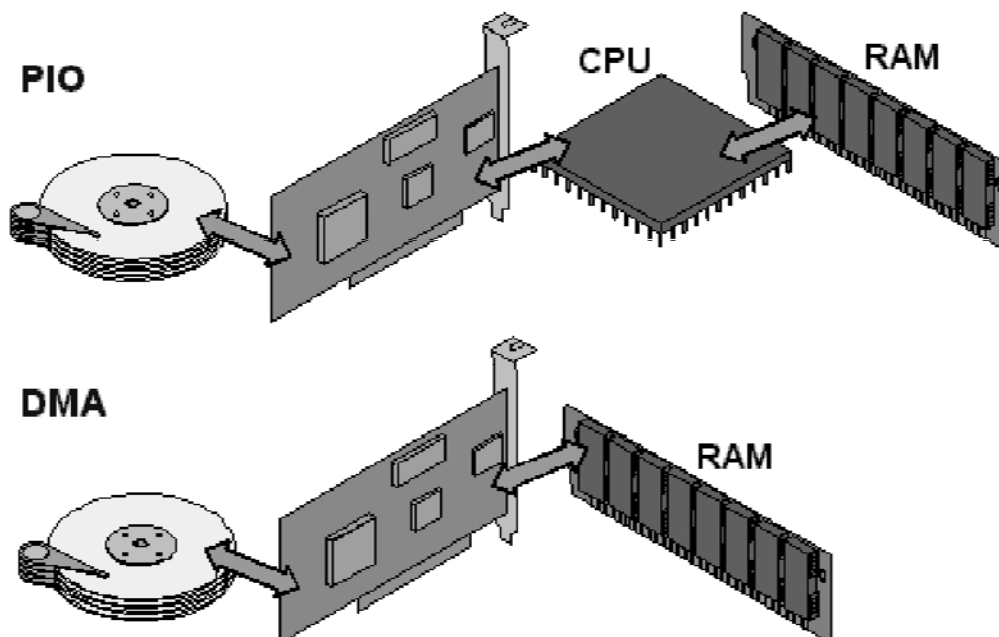
Mecanismes de transferència

- Transferència per programa
 - PIO: *programmed input/output*
 - La UCP du a terme la transferència mitjançant l'execució d'instruccions
 - `IN dada, port`
 - `OUT dada, port`
 - `lw $8, port($0)`
 - `sh $8, port($0)`
 - `lb $8, port($0)`
- Transferència per accés directe a memòria (ADM)
 - En anglés: DMA: *direct access memory*
 - La transferència es fa sense la intervenció de la UCP

97

PIO versus DMA

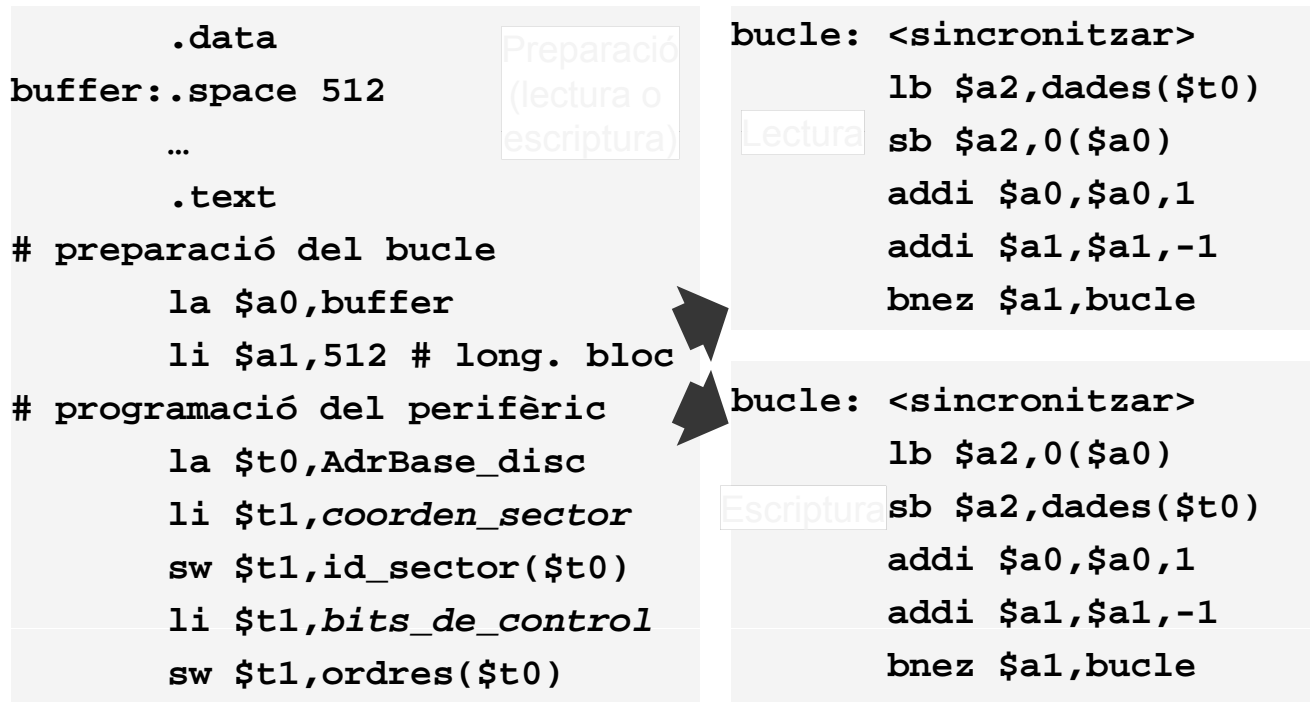
From Computer Desktop Encyclopedia
© 1998 The Computer Language Co. Inc.



98

Transferències de bloc per programa

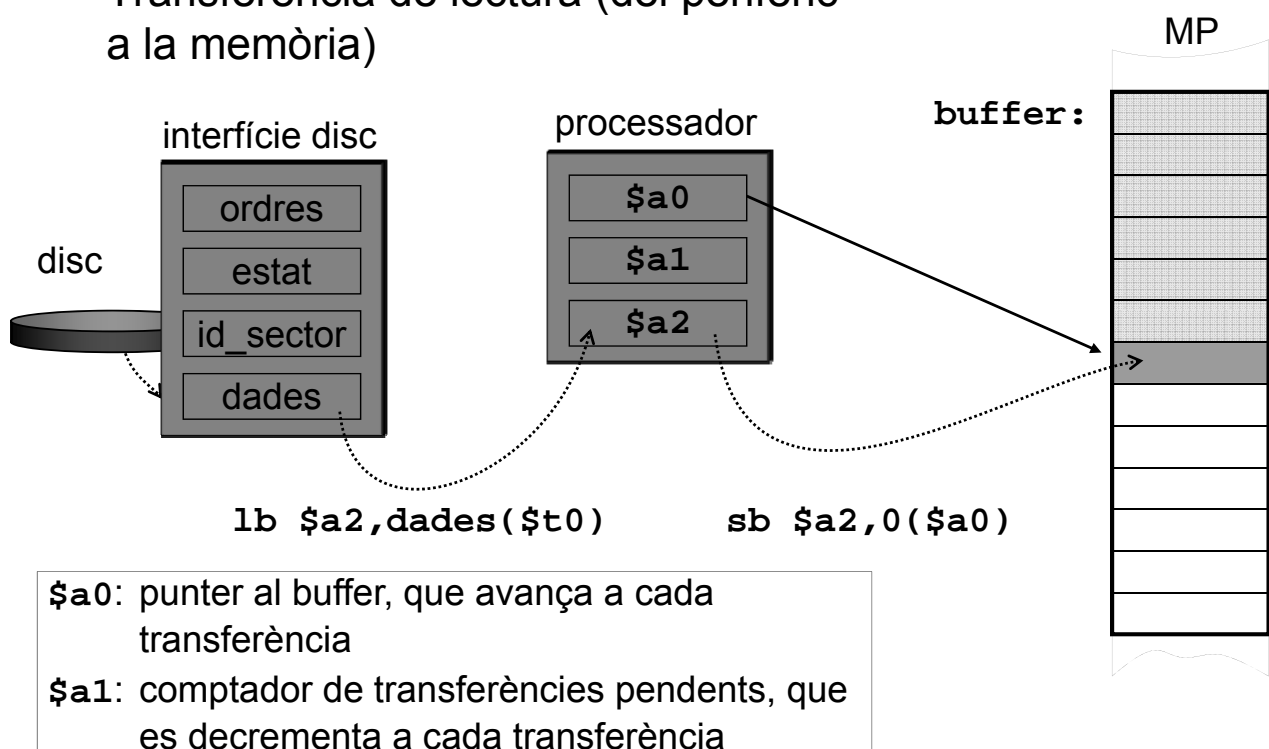
- Esquema de programa



99

Transferències de bloc per programa

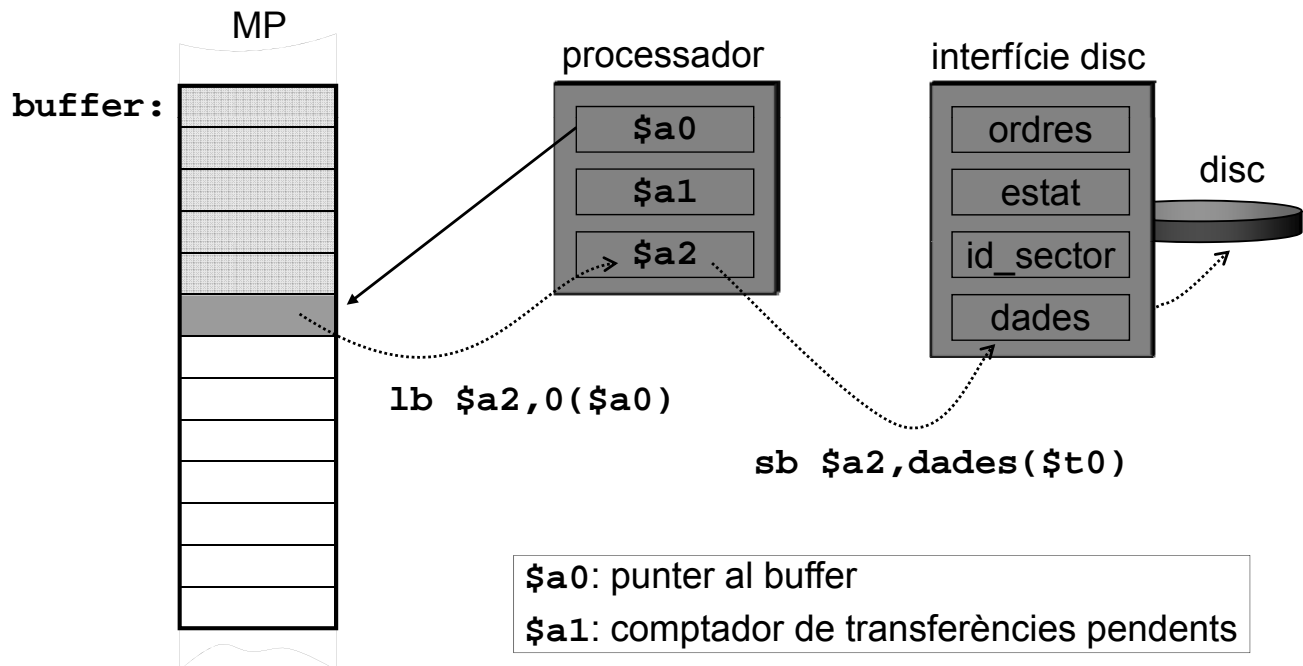
- Transferència de lectura (del perifèric a la memòria)



100

Transferències de bloc per programa

- Transferència d'escriptura (de la memòria al perifèric)



101

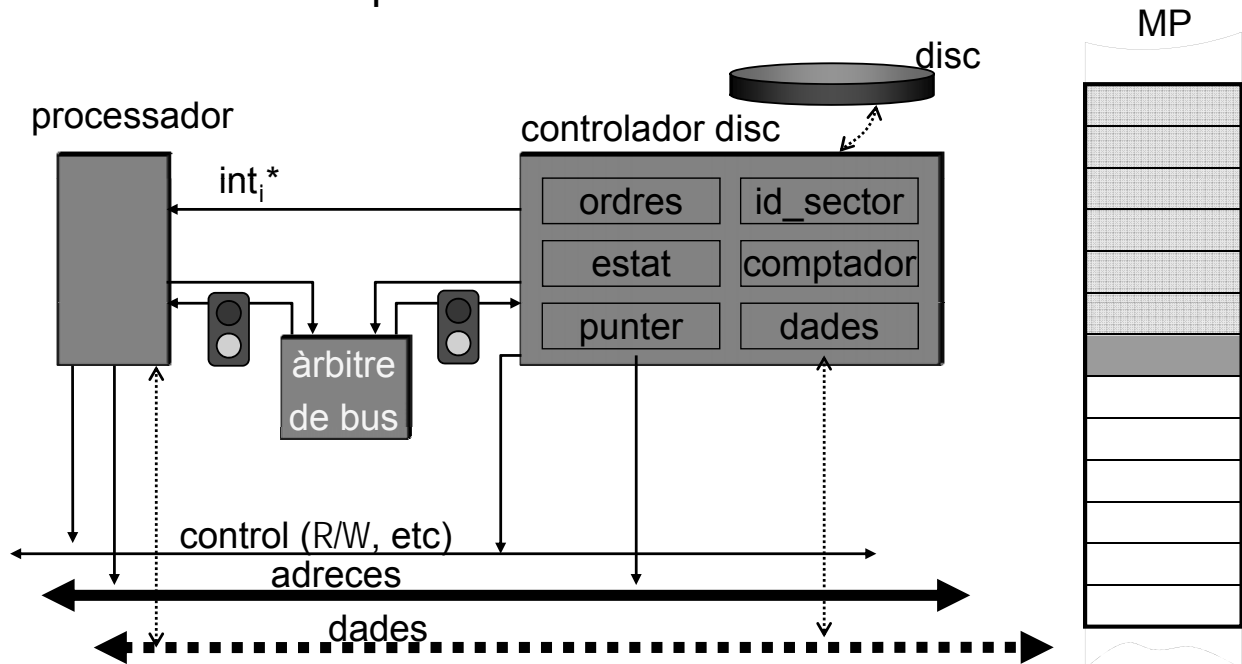
Suport a transferències per ADM

- Noves funcions del controlador de perifèric
 - El controlador del perifèric pot accedir a la memòria per a llegir i escriure manejant les línies del bus (com si fóra el processador)
- Novetats en la interfície
 - La interfície incorpora registres d'adreça (per fer el paper de \$a0) i de compte de bytes (per fer de \$a1)
- Sincronització
 - El controlador del perifèric indica que està preparat (i provoca la interrupció si escau) quan acaba la **transferència completa**.
- Compartició del bus
 - Cal preveure un arbitratge per a distribuir l'ús del bus entre diversos dispositius: el(s) processador(s) i el(s) controlador(s) de perifèric amb capacitat d'ADM

102

Suport a transferències per ADM

- Connexions importants:



103

Funcionament bàsic de l'ADM

- Programació del controlador
 - Tipus d'operació:
 - Entrada: Lectura de perifèric / escriptura en la memòria
 - Eixida: Lectura de memòria / escriptura en perifèric
 - Registre d'adreces: adreça inicial del buffer
 - Comptador de dades: defineix la longitud del bloc a transferir

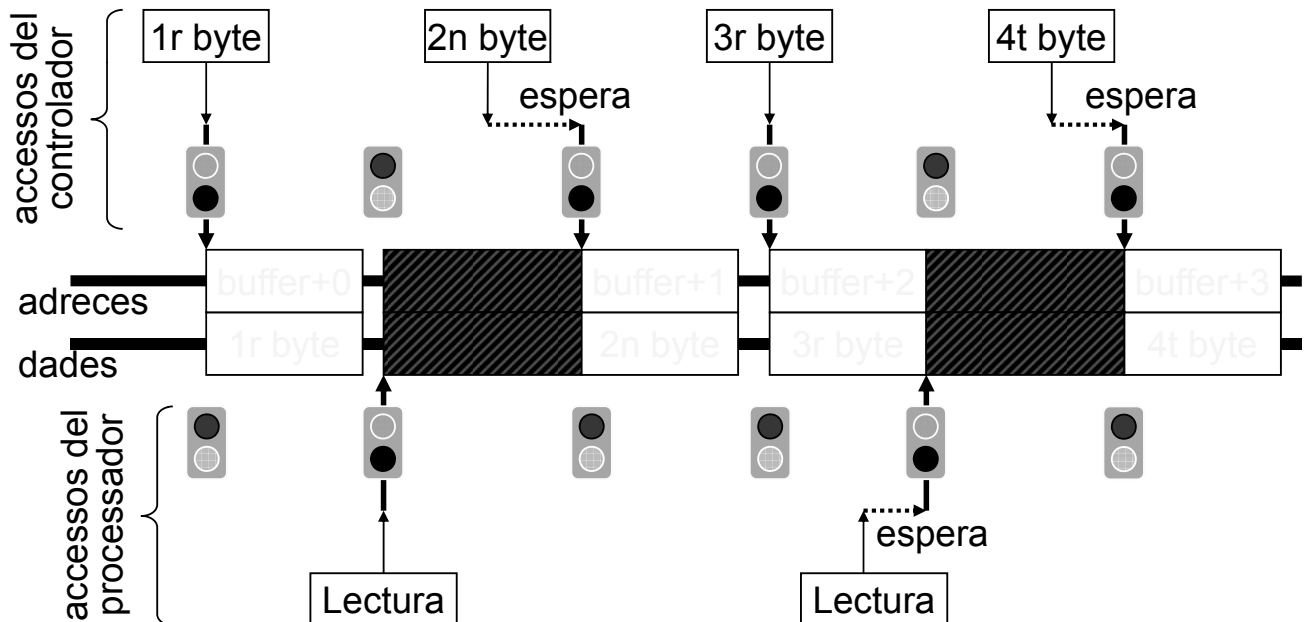
```
.data
buffer: .space 512
...
.text
...
la $t0,AdrBase_disc
# preparació de l'ADM
la $t1,buffer
sw $t1,punter($t0)
li $t1,512
sw $t1,comptador($t0)
# paràmetres de l'operació
li $t1,coordenades
sw $t1,id_sector($t0)
li $t1,bits_de_control
sw $t1,ordres($t0)
```

104

Arbitratge del bus

- Cronograma exemple

- accés del controlador de disc al bus mentre el processador fa accessos de lectura



105

Perifèrics amb ADM dins del sistema operatiu

- Excepcions bàsiques:

- Petició d'E/S feta pel procés d'usuari
 - La funció del SO programa la interfície del perifèric amb tots els paràmetres
 - Identificació de l'operació (lectura o escriptura)
 - Coordenades del sector en el disc
 - Adreça en la memòria i longitud del buffer
 - El procés que l'ha feta pot quedar en espera
- Interrupció del perifèric quan acaba l'operació
 - El controlador de disc provoca la interrupció quan el comptador arriba al zero
 - Si hi ha un procés que espera, passa a estat de llest

106

Exemple de funcions

- Especificació i ús de les funcions de disc

Funció	Codi	Arguments	Resultat
Read_Disk	\$v0 = 666	\$a0 = punter al <i>buffer</i> \$a1 = coordenades del sector	
Write_Disk	\$v0 = 667	\$a0 = punter al buffer \$a1 = coordenades del sector	

```
                .data
buffer:         .space 512
                ...
                .text
                ...
# petició de lectura del disc
                li $v0,666
                la $a0,buffer
                li $a1,coordenades
                syscall
```

107

Exemple de funcions

- Tractament de la funció en el manejador

```
                .ktext
fun666:         la $t0,adreça_base_disc
                sw $a0,punter($t0)
                li $t1,512
                sw $t1,comptador($t0)
                sw $a1,id_sector($t0)
                li $t1,bits_de_control
                sw $t1,control($t0)
                jal suspén_aquest_procés
                j retexc
```

- Tractament de la interrupció del controlador de disc

```
intn:          .ktext
                jal activar_proc_en_espera
                j retexc
```

108

FIN
