Tema 7. Memòria Virtual

Curs 2019-20 Primavera

Grup 30

Joan Manuel Parcerisa

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

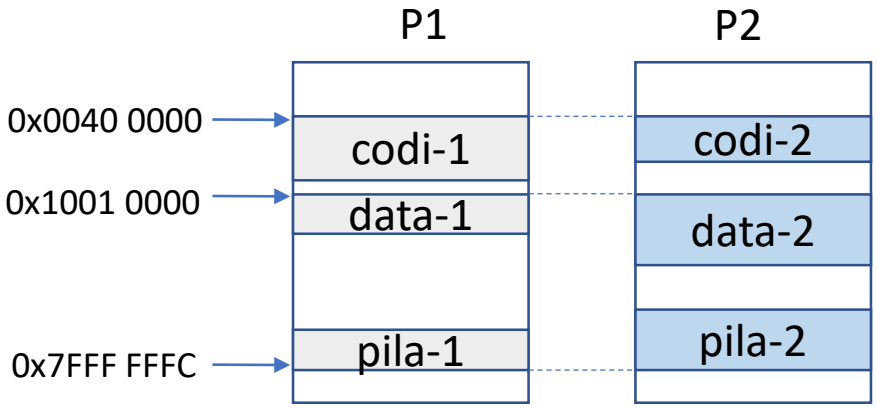
FIB

Introducció

- Múltiples programes simultanis
  - Necessitat de Reubicació
  - Necessitat de Protecció
  - Necessitat d'excedir la capacitat de MP
- Memòria Virtual
  - Traducció d'adreces
  - Programes parcialment carregats en memòria
  - Jerarquia de memòria

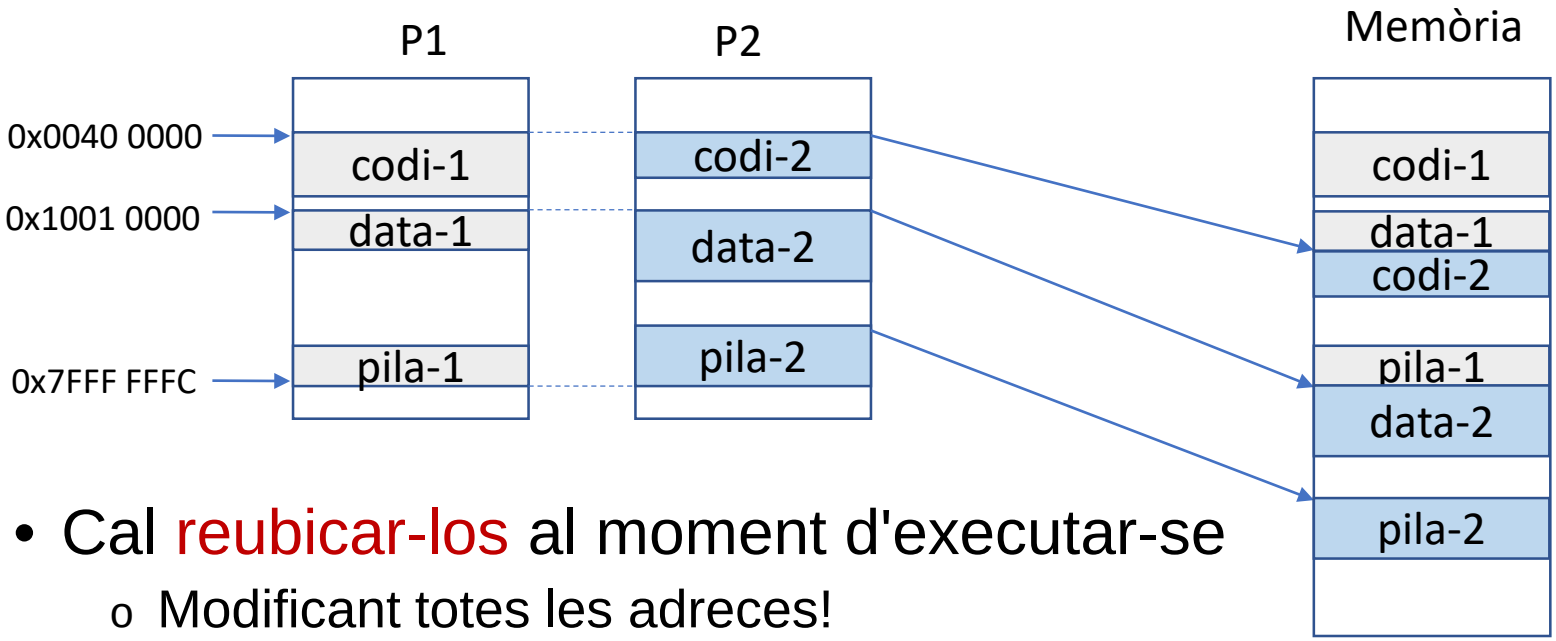
Necessitat de reubicació

- La compilació, assemblatge i enllaçat
  - Assignen adreces absolutes a instruccions i dades
  - Les mateixes a tots els programes!



Necessitat de reubicació

- La compilació, assemblatge i enllaçat
  - Assignen adreces absolutes a instruccions i dades
  - Les mateixes a tots els programes!

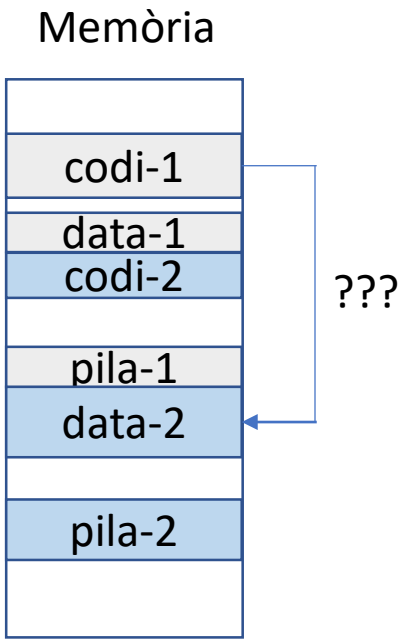


- Cal reubicar-los al moment d'executar-se
  - Modificant totes les adreces!

Necessitat de protecció

- Un programa podria accedir a dades d'un altre
  - Podria examinar tota la memòria...
  - ... i buscar passwords!
  - ... o modificar dades!

- Cal un mecanisme de protecció

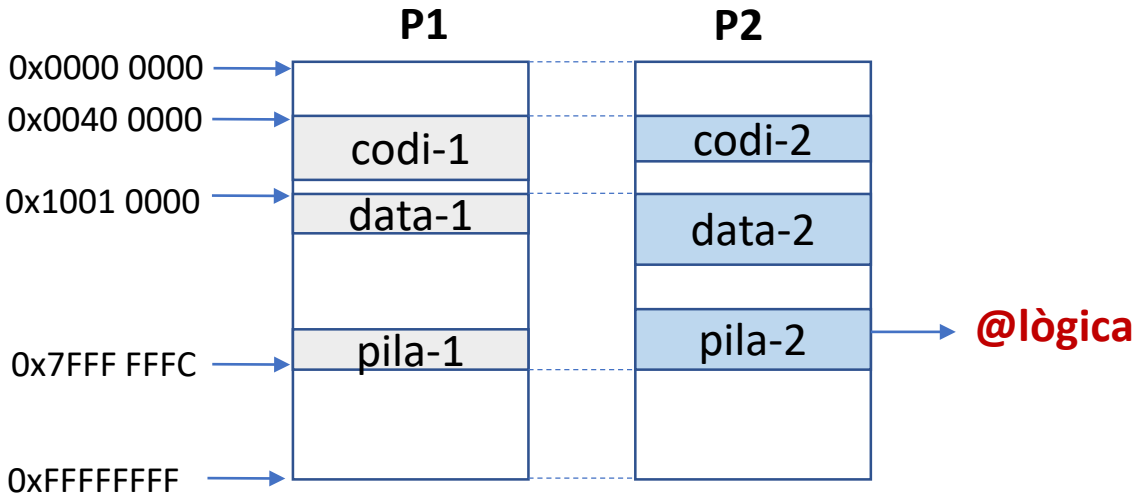


Necessitat d'excedir la capacitat de MP

- La capacitat de memòria RAM depèn del que instal·lem:
  4GB, 8GB, 16GB, 128GB, ...
- Què passa si la mida del programa excedeix la capacitat istal·lada?
  - El programa ha d'estar parcialment carregat en memòria
- Solució antiga: els *overlays*
  - El programador dividia el programa en mòduls
  - En executar-se, sols es carregaven en memòria els mòduls indispensables. La resta, sobre demanda
  - Gestió a càrrec del propi programa
  - → Farragós i ineficient

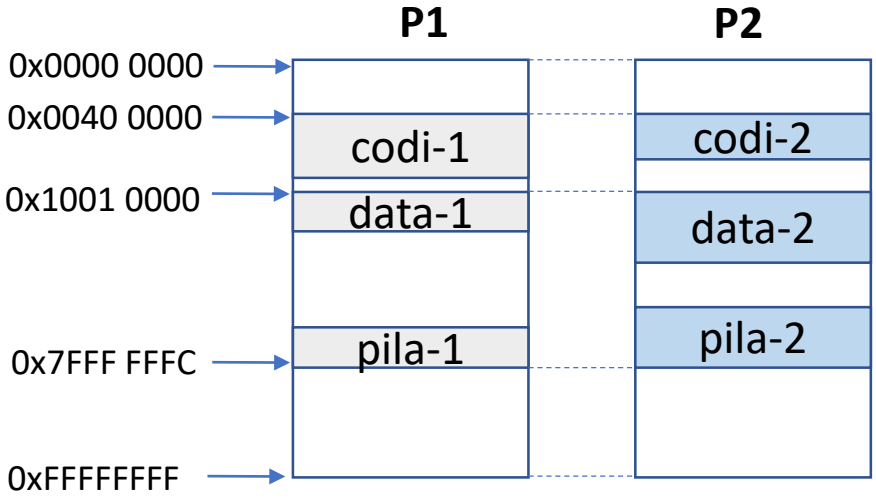
Solució MV: traducció d'adreces

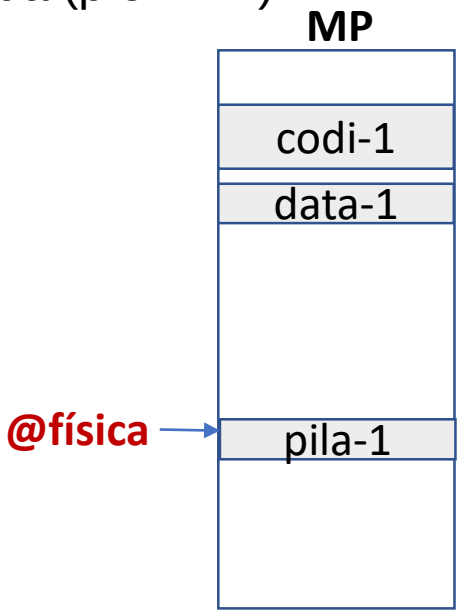
- Espai d'adreçament lògic
  - Són les adreces que assigna el compilador a cada programa, i que la CPU envia a la memòria en cada accés
  - Cada programa té el seu propi espai, mateixa mida (p.ex. 2^{32})



Solució MV: traducció d'adreces

- Espai d'adreçament lògic
  - Són les adreces que assigna el compilador a cada programa, i que la CPU envia a la memòria en cada accés
  - Cada programa té el seu propi espai, mateixa mida (p.ex. 2^{32})

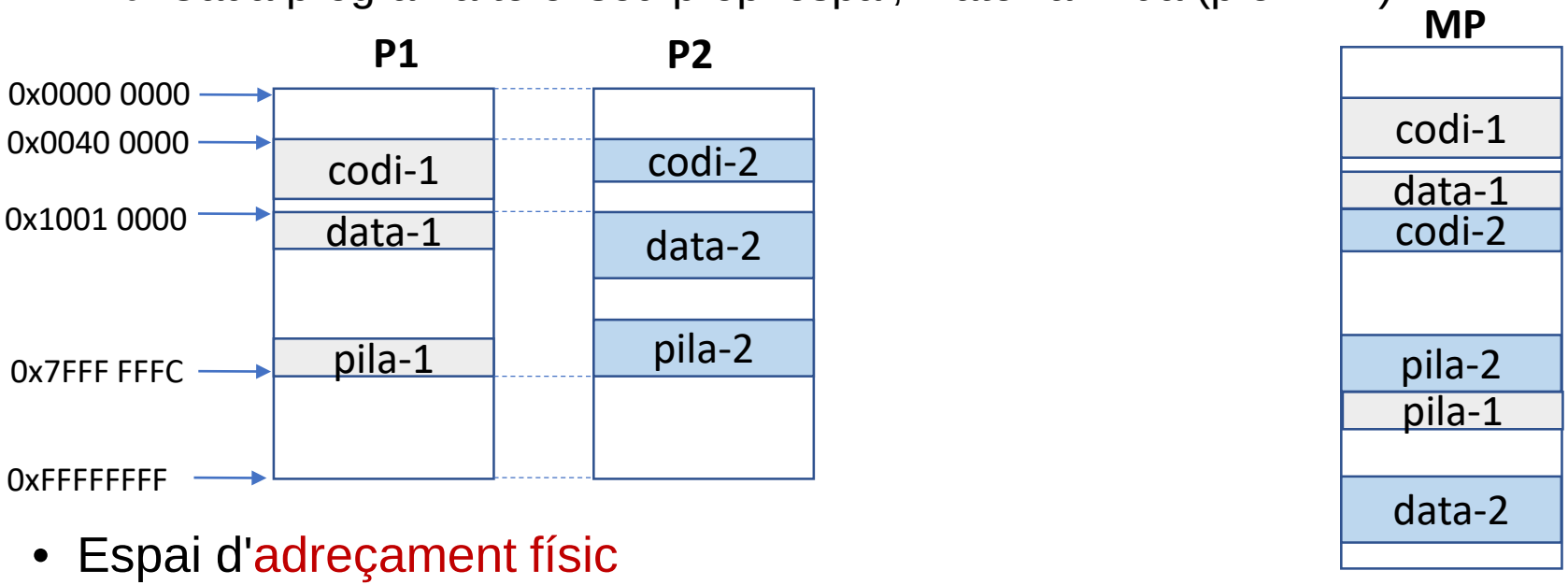




- Espai d'adreçament físic
  - Són les adreces "reals" que espera rebre la MP

Solució MV: traducció d'adreces

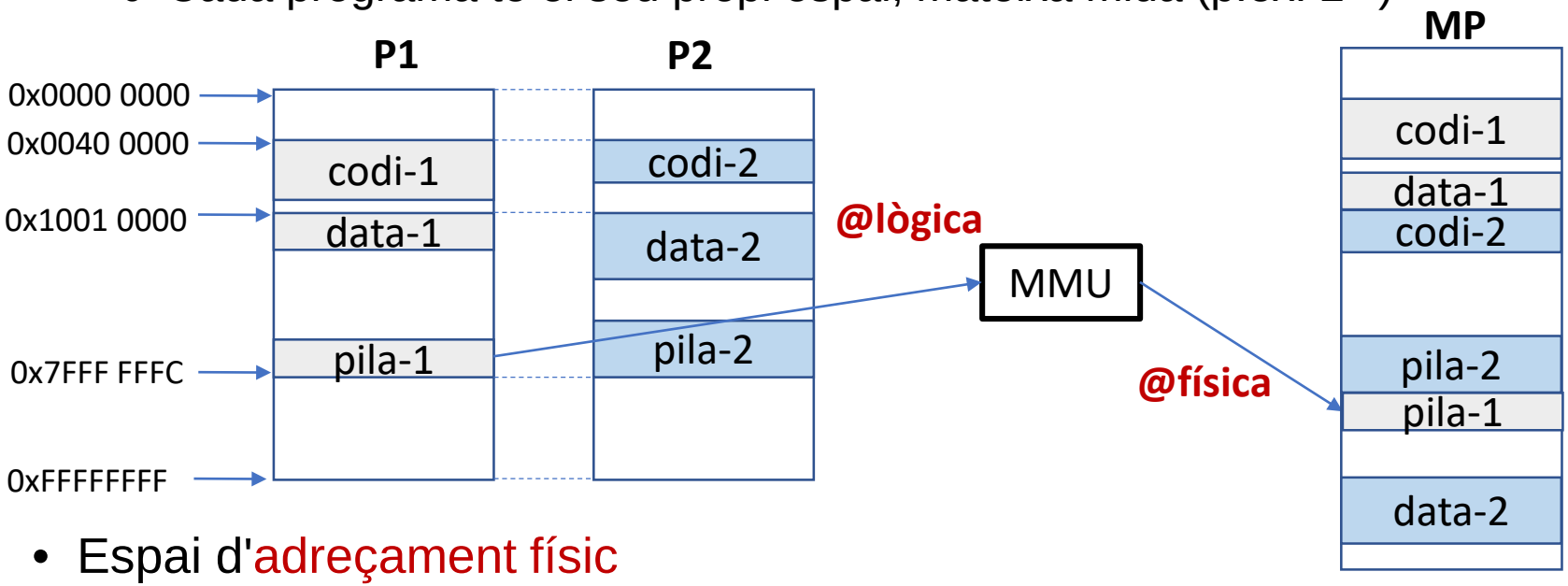
- Espai d'adreçament lògic
  - Són les adreces que assigna el compilador a cada programa, i que la CPU envia a la memòria en cada accés
  - Cada programa té el seu propi espai, mateixa mida (p.ex. 2^{32})



- Espai d'adreçament físic
  - Són les adreces "reals" que espera rebre la MP
  - Porcions de cada programa s'ubiquen en posicions lliures de MP

Solució MV: traducció d'adreces

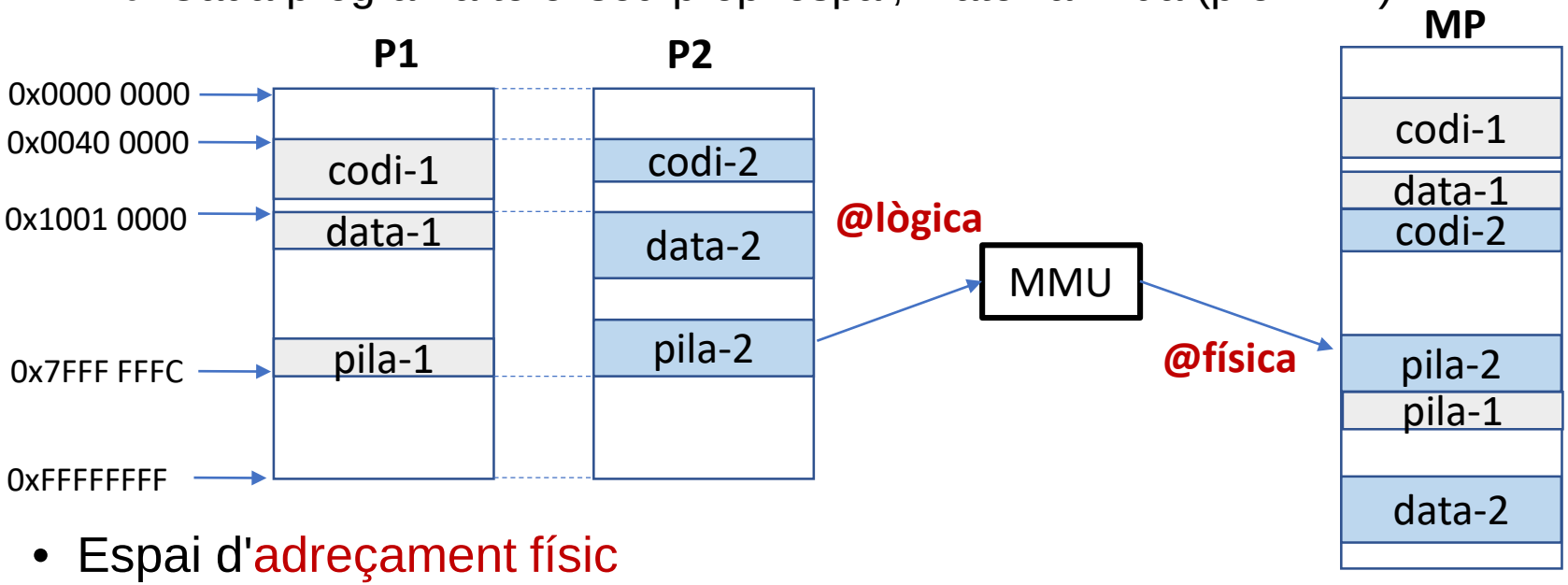
- Espai d'adreçament lògic
  - Són les adreces que assigna el compilador a cada programa, i que la CPU envia a la memòria en cada accés
  - Cada programa té el seu propi espai, mateixa mida (p.ex. 2^{32})



- Espai d'adreçament físic
  - Són les adreces "reals" que rep la MP
  - Porcions de cada programa s'ubiquen en posicions lliures de MP
  - El hardware **tradueix** automàticament cada adreça lògica a física
    - recorda en quines adreces físiques s'ha ubicat cada porció
    - Hardware MMU: Memory Management Unit

Solució MV: traducció d'adreces

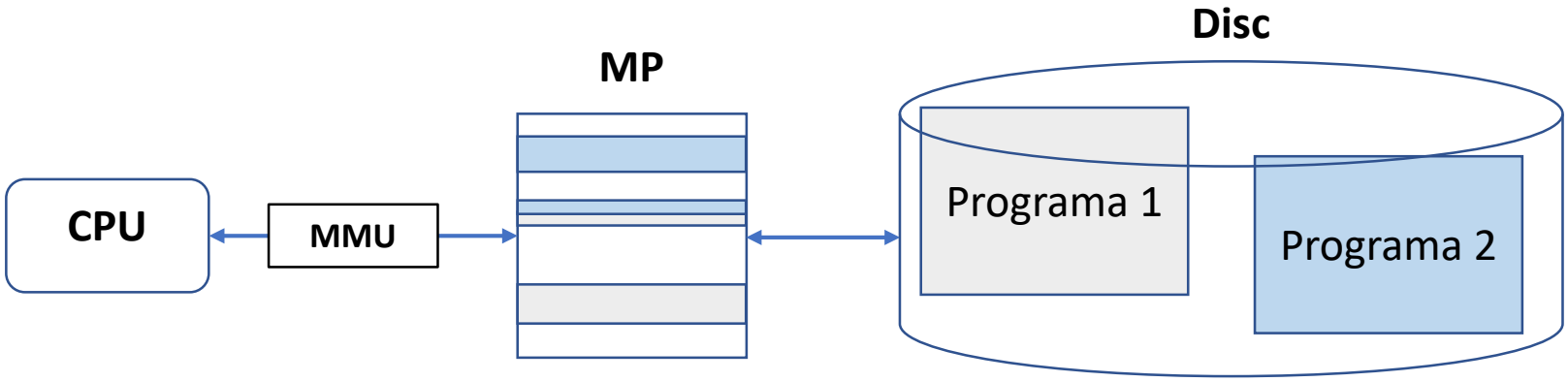
- Espai d'adreçament lògic
  - Són les adreces que assigna el compilador a cada programa, i que la CPU envia a la memòria en cada accés
  - Cada programa té el seu propi espai, mateixa mida (p.ex. 2^{32})



- Espai d'adreçament físic
  - Són les adreces "reals" que espera rebre la MP
  - Porcions de cada programa s'ubiquen en posicions lliures de MP
  - El hardware MMU fa la **traducció** de cada adreça lògica a física
    - recorda en quines adreces físiques s'ha ubicat cada porció

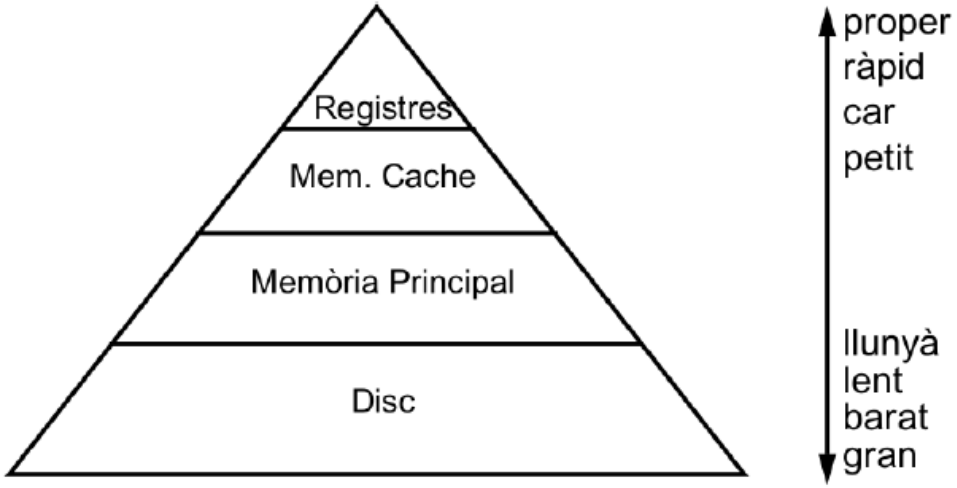
MV: Programes parcialment carregats en MP

- Els programes resideixen íntegrament al disc
- Les porcions accedides recentment es guarden en MP
- El mecanisme de memòria virtual s'encarrega de moure dades del Disc a la MP, explotant la localitat



Solució elegant amb Memòria Virtual

- Traducció d'adreces lògiques a físiques
  - Permet la reubicació automàtica
  - Proporciona protecció i compartició entre programes
- Usar la MP com una "cache" del disc, que esdevé un nivell més de la jerarquia de memòria
  - Permet excedir la capacitat de la memòria principal
- Gestió conjunta entre hardware i Sistema Operatiu
  - Transparent al programador

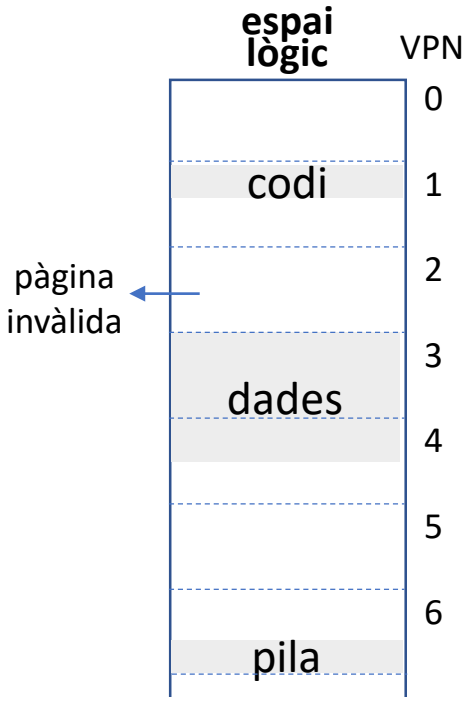


Memòria Virtual Paginada

- Organització en pàgines
- Traducció d'adreces
- Polítiques d'emplaçament, reemplaçament i escriptura
- Traducció amb Taula de Pàgines

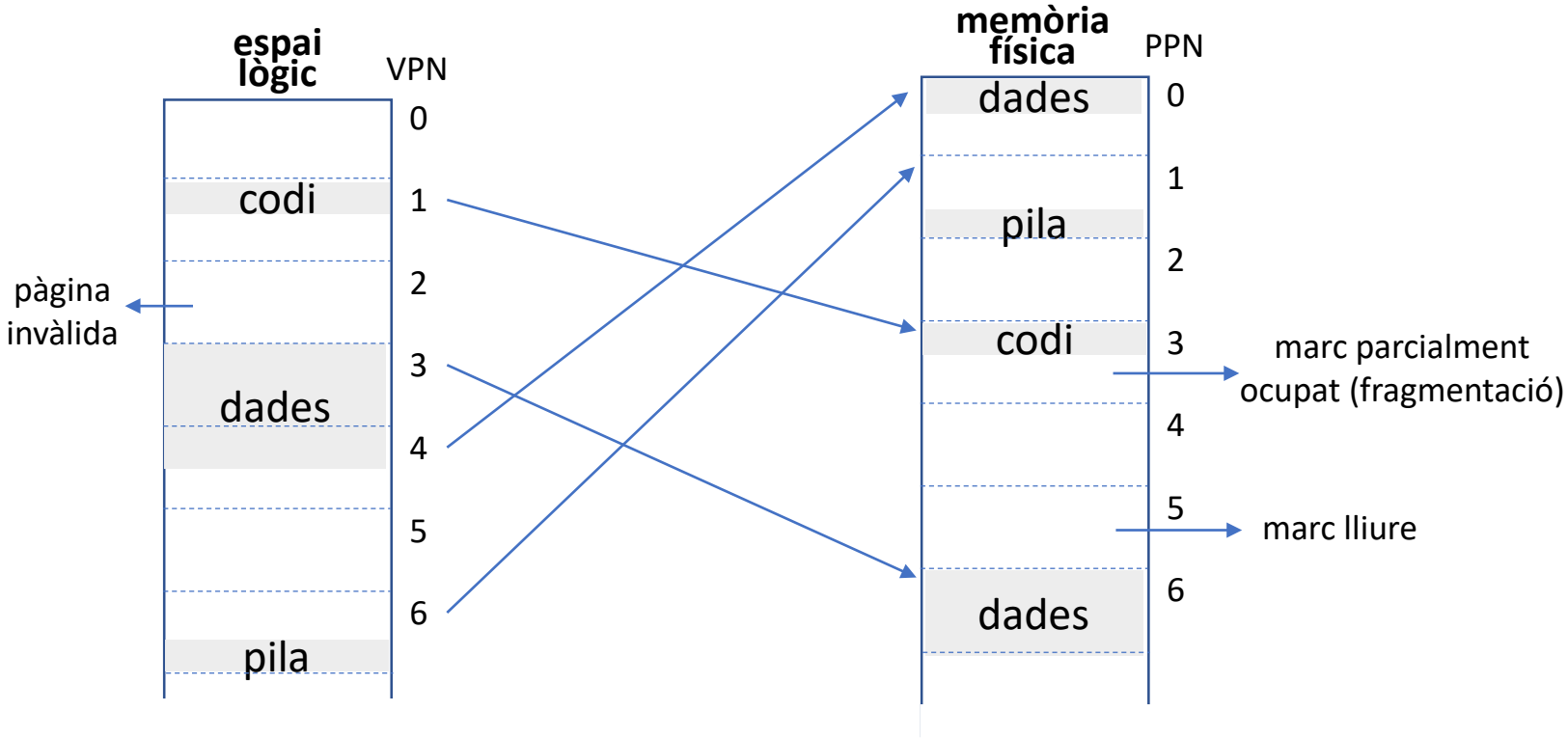
Organització en pàgines

- Espai d'adreçament **lògic o virtual**
  - És el format per les adreces determinades pel compilador i l'enllaçador
- Es divideix en **pàgines:** blocs contigus de mida fixa, potència de 2 (p.ex. 4KB, 4MB,...), anàlogues als "blocs de cache"
  - Es numeren amb el **VPN**, (Virtual Page Number), per identificar-les
  - Sols les pàgines usades ("vàlides") poden ser accedides pel programa



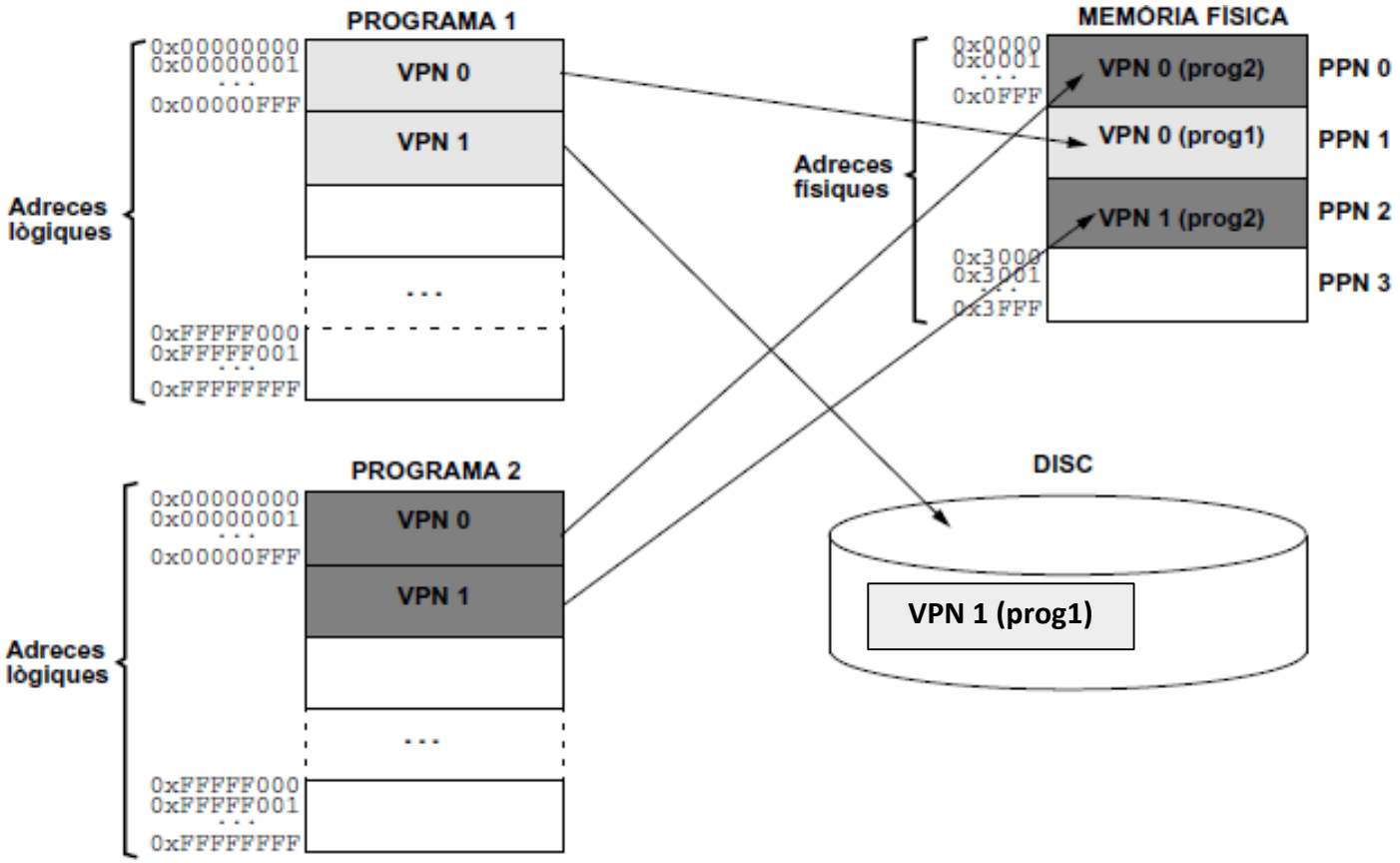
Organització en pàgines

- Espai d'adreçament **físic**
  - Depèn de la memòria RAM instal·lada (p.ex. 256MB, 4GB, 32GB, etc.)
- Es divideix en **marcs de pàgina**
  - De la mateixa mida que les pàgines
    - Un marc és com un "contenidor", que pot allotjar una pàgina
  - Es numeren amb el **PPN**, (Physical Page Number), per identificar-los



Exemple

- Exemple:
  - Pàgines de 4KB (2^{12} bytes), memòria física de 16KB (4 marcs)
  - Programa 1 i Programa 2 fan servir sols 2 pàgines lògiques (VPN0 i VPN1)
  - La pàgina VPN1 del programa 1 no està carregada en memòria

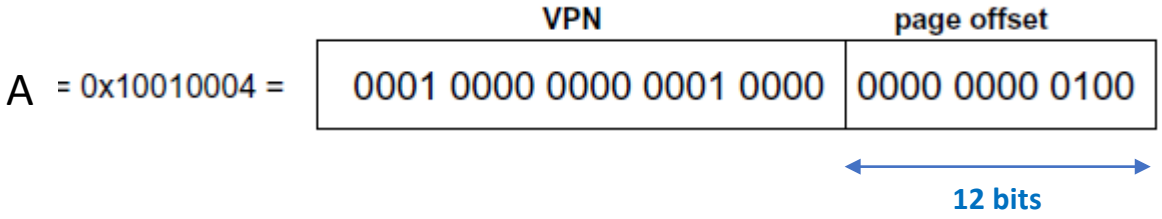


Subdivisió de l'adreça lògica

- A quina pàgina (VPN) pertany l'adreça lògica A?
  - Per a una adreça A, si la mida de pàgina és T

    VPN = A div T
  - I la posició relativa de A dins la pàgina (offset) és

    offset = A mod T
  - Per exemple, si **A=0x10010004** i la mida és **T=4KB** (2^{12} bytes)

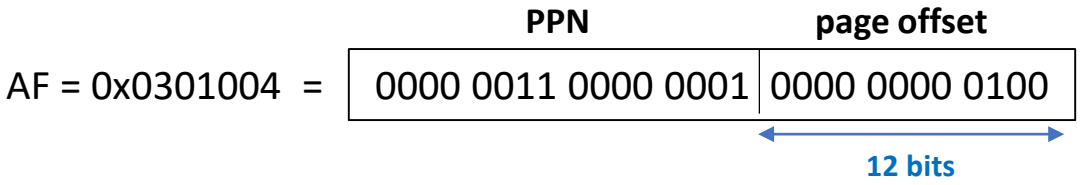


Subdivisió de l'adreça física

- A quin marc de pàgina (PPN) pertany l'adreça física AF?
  - Per a una adreça física AF, si la mida de pàgina és T

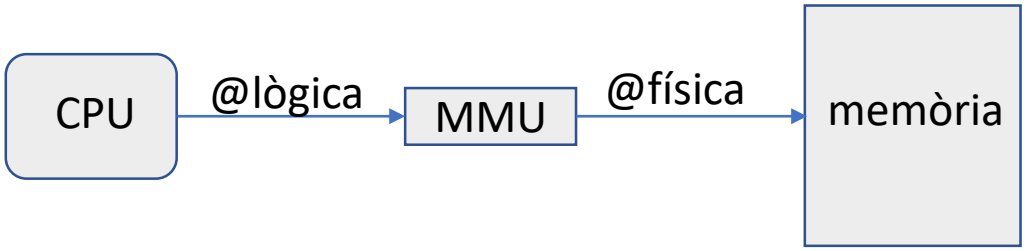
    PPN = AF div T
  - I la posició relativa de AF dins el marc de pàgina (offset) és

    offset = AF mod T
  - P.ex.: Memòria de 2^{28} bytes, **T=** 2^{12} bytes, **AF = 0x0301004**

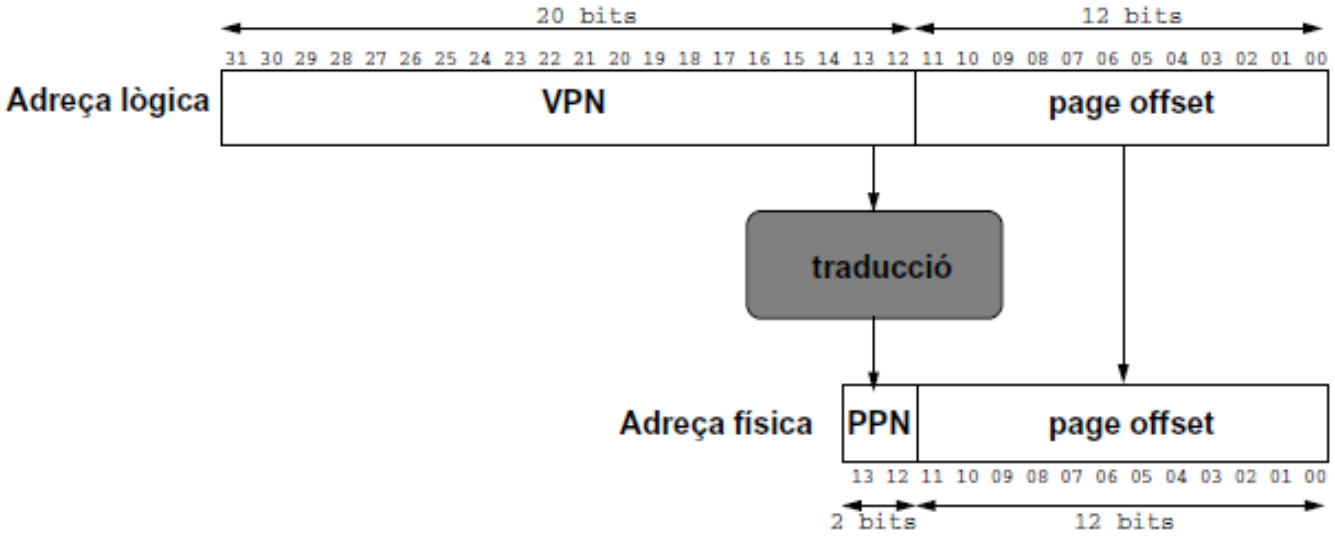


Traducció d'adreces

- El processador treballa amb adreces lògiques
- Cada adreça (codi o dades) s'ha de traduir a una adreça física
- Ho fa la MMU (Memory Management Unit)

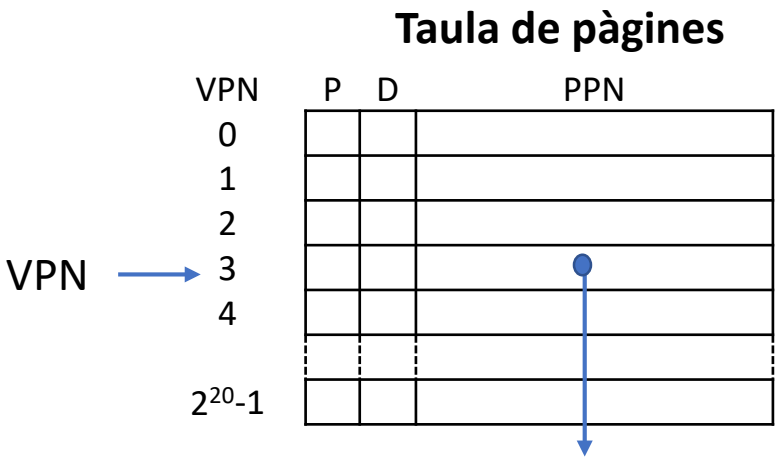


  - Tradueix el VPN al corresponent PPN
  - L'offset no canvia



Traducció amb Taula de Pàgines

- Com sap la MMU en quin marc (PPN) està cada pàgina (VPN)?
- Consultant la **Taula de Pàgines** que manté el S.O.
  - Té tantes entrades com pàgines té l'espai lògic (s'indexa amb el VPN)
- Cada entrada (PTE o Page Table Entry) conté
  - **PPN**: marc de pàgina on es troba la pàgina
  - Bit de presència (**P**): val 1 si la pàgina **és vàlida** i **està present** en MP
  - Bit de modificada (**D**): val 1 si **ha estat modificada** (per un Store)
  - Altres bits: Permisos diversos (Lectura/ Escriptura/ Execució), i bit de Referència (per gestionar l'algorisme LRU)
- Si la pàgina no està present en memòria
  - La PTE conté la ubicació en disc



Encert i Fallada de pàgina

- Memòria i Disc són 2 nivells de la jerarquia de memòria
  - En principi, totes les pàgines resideixen al disc
  - A mesura que s'accedeixen, es van carregant en MP
- Encert de pàgina
  - Quan la pàgina accedida està present en MP (bit P=1)
  - La MMU obté el PPN de la PTE corresponent
- Fallada de pàgina
  - Quan la pàgina accedida no està present en MP (bit P=0)
  - El S.O. la copia del disc a un marc en MP (milions de cicles!)
  - Seleciona *qualsevol* marc lliure: emplaçament totalment associatiu, per minimitzar fallades
  - El S.O. Actualitza la PTE corresponent (bits P, D i PPN)
  - Es reexecuta la instrucció que ha causat la fallada

Reemplaçament LRU

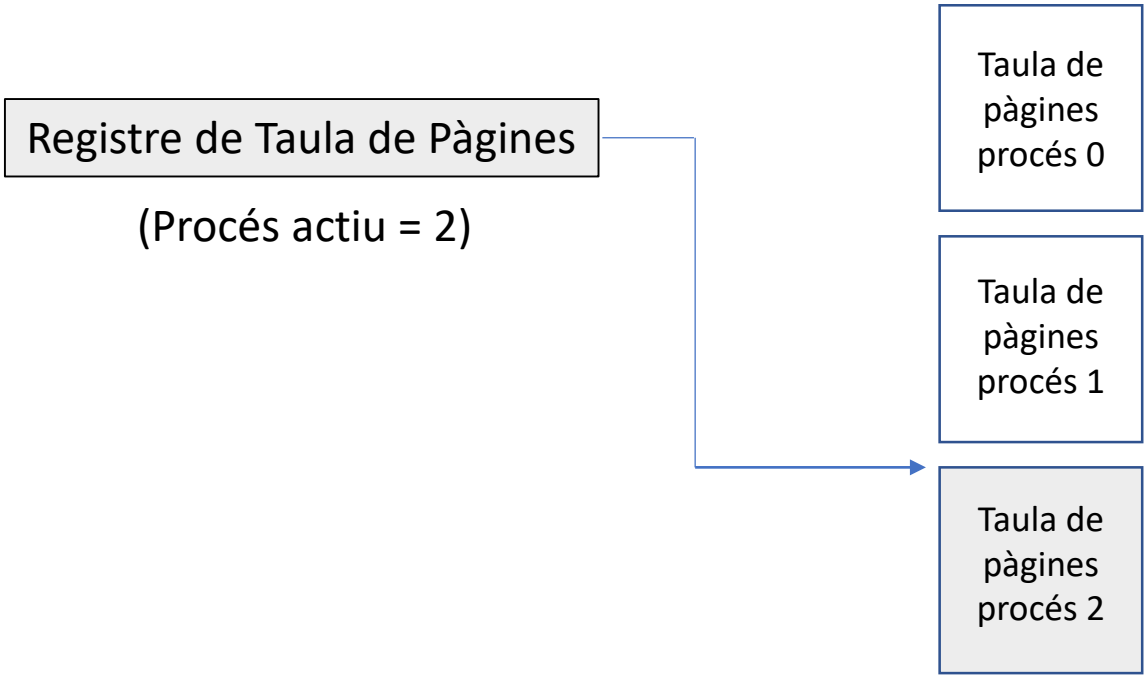
- Reemplaçament de pàgina
  - Quan no queden marcs lliures, el S.O. en reemplaça un d'ocupat
  - Algorisme de reemplaçament LRU, per minimitzar fallades
  - Abans de reemplaçar-la, verifica si està modificada (bit D=1)
    - Si està modificada, l'escriu al disc
  - En Linux, la zona de disc on s'emmagatzemen les pàgines s'anomena *espai d'intercanvi* o *swap*

Escriptura retardada

- Escriptures (instruccions store)
  - L'escriptura immediata és totalment impràctica (l'accés al disc tarda milions de cicles!)
  - Escriptura retardada amb assignació
  - Els stores només escriuen a MP (no al disc)
  - Quan un store escriu a memòria, la pàgina es marca "modificada" posant a 1 el bit Dirty (D=1)
    - Si més tard és reemplaçada, caldrà escriure-la al disc

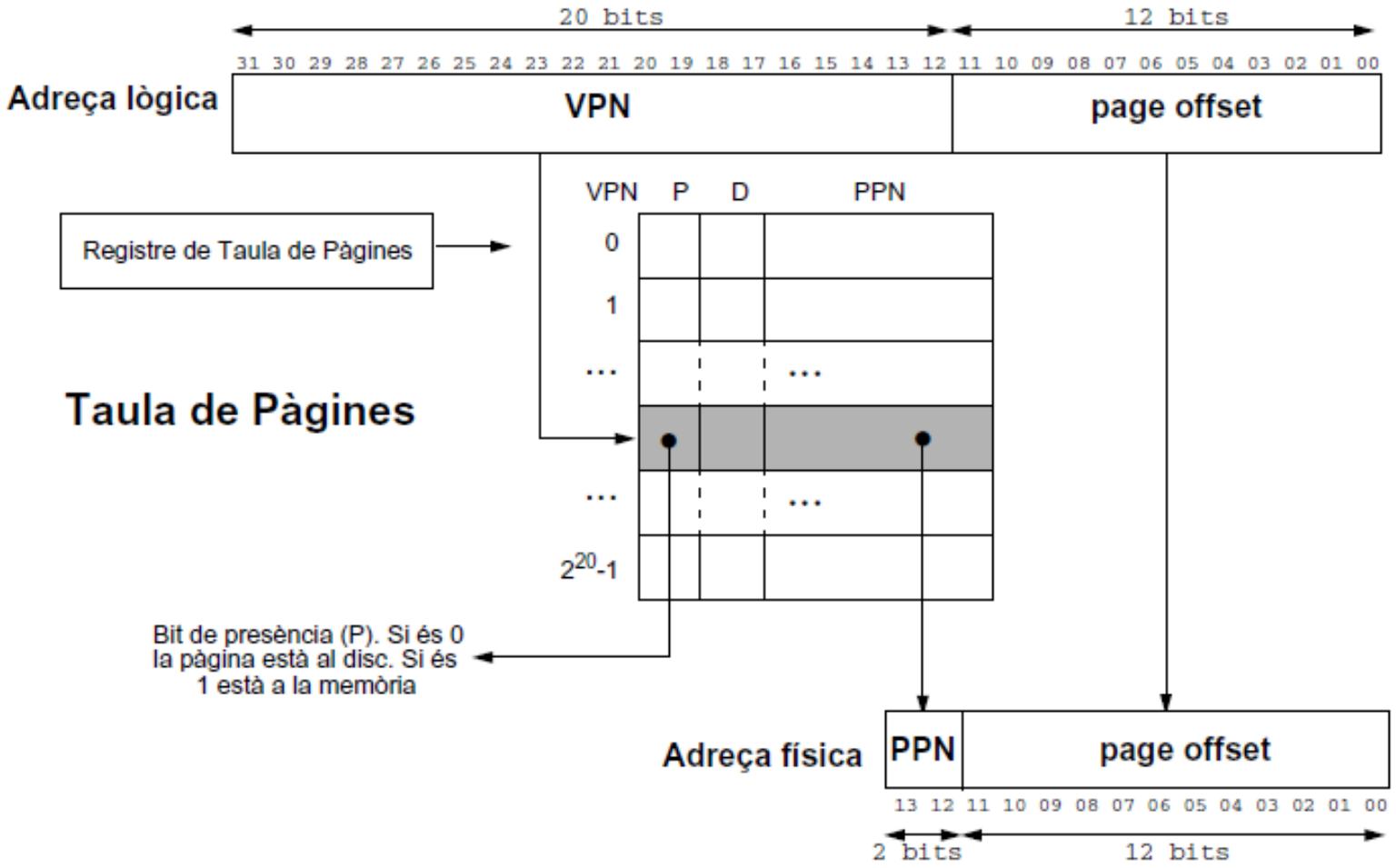
Traducció amb Taula de Pàgines

- Múltiples processos poden executar-se concurrentment
  - Cada un té el seu propi espai lògic → Té la seva Taula de Pàgines
  - S'alternen en l'ús de CPU al llarg del temps, fent canvis de context
  - Però sols el *procés actiu* s'executa en un instant donat
- La CPU manté un **Registre de Taula de Pàgines**
  - Apunta a la Taula de Pàgines del *procés actiu*
  - Es modifica cada cop que el S.O. fa un *canvi de context*



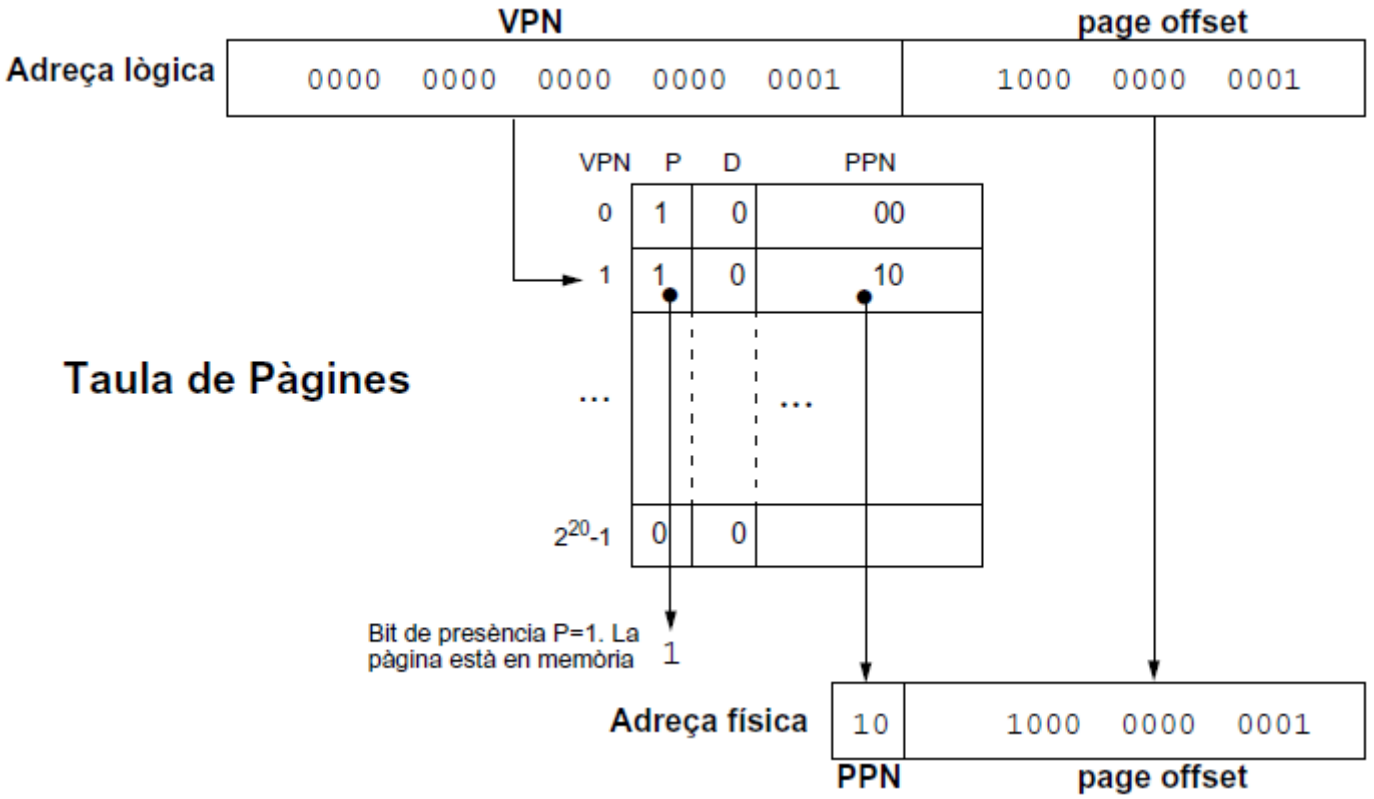
Traducció amb Taula de Pàgines

- **Diagrama de flux** de la traducció
  - Suposem pàgines de 4KB (2^{12} bytes), MP de 16KB (4 marcs)



Exemple

- **Exemple**:
  - La MMU tradueix l'adreça lògica A = 0x00001801
  - La pàgina està Present en memòria (P=1) i resideix al marc PPN = 2
  - L'adreça física resultant és: 0x2801

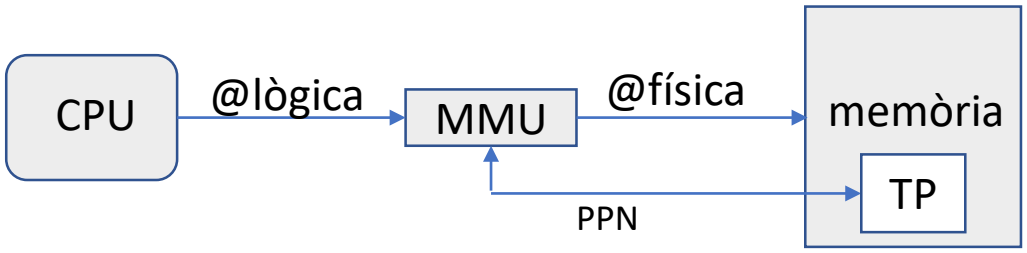


Aspectes de disseny

- Traducció ràpida amb TLB
- Gestió de les fallades de pàgina i de TLB
- Protecció i Compartició
- Integració de TLB i Cache

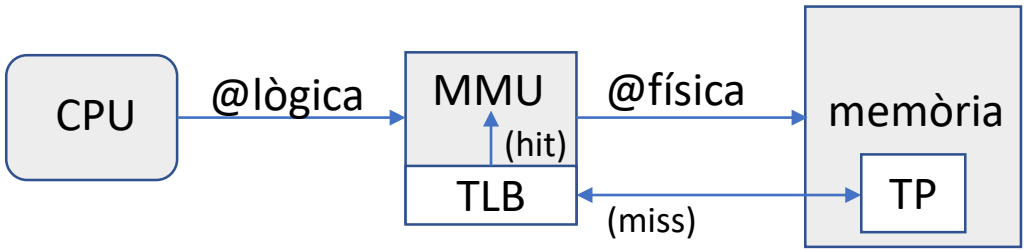
Traducció ràpida amb TLB

- Problema
  - Les TP resideixen en MP
  - Cada accés a memòria que genera la CPU requereix 2 accessos a memòria física:
    - 1 accés a la TP per traduir l'adreça
    - 1 accés per llegir/escriure la dada
  - L'accés a la TP augmenta molt el temps d'accés a MP !



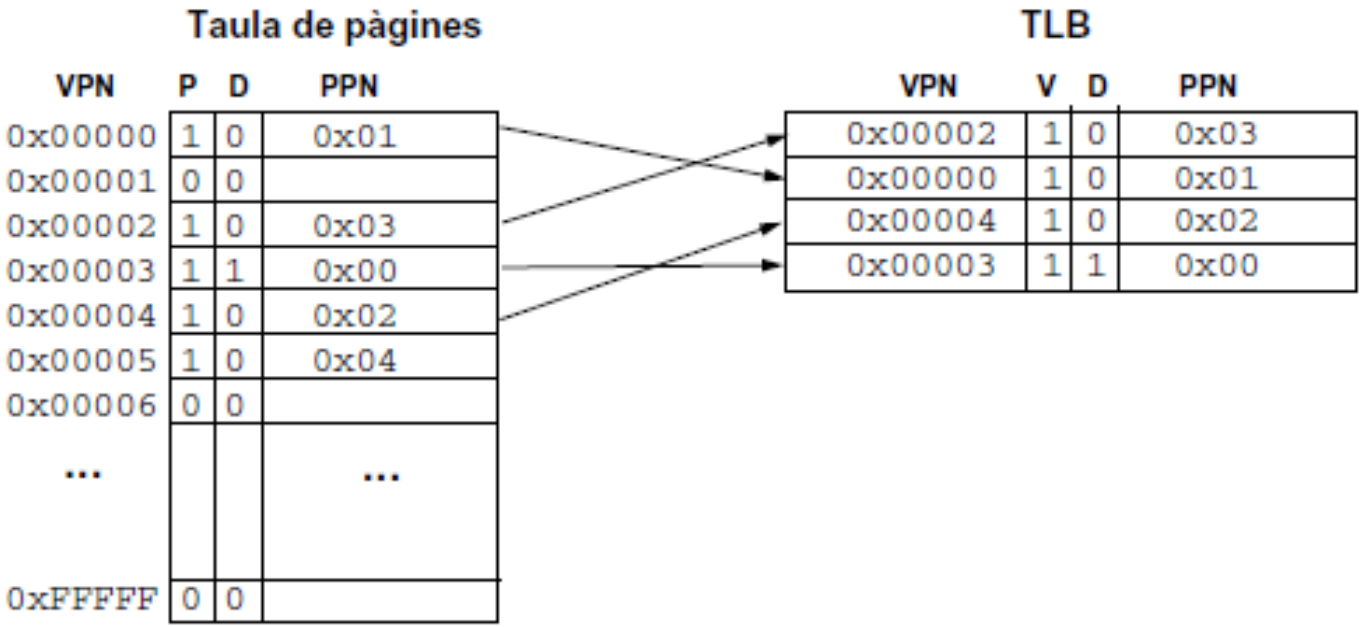
Traducció ràpida amb TLB

- Solució: el TLB (Translation-Lookaside Buffer)
  - És una cache de traduccions: guarda les entrades (PTE) de la TP que s'han accedit més recentment
  - Aprofita que els accessos a la TP tenen molta localitat
  - Típicament: 16 - 512 PTEs, 0,5 - 1 cicle si hit, i taxa de fallades de 0,01% - 1%



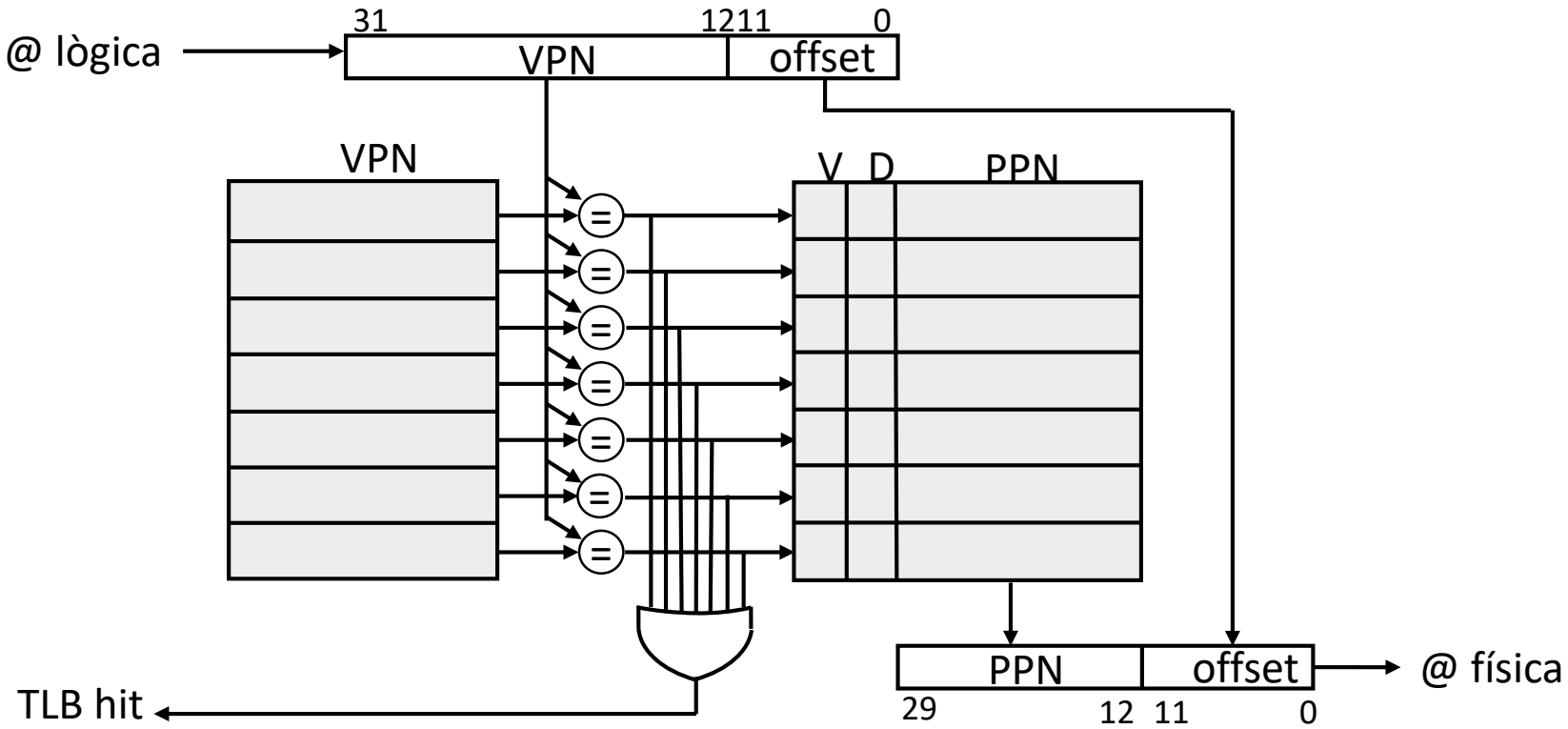
Traducció ràpida amb TLB

- Organització del TLB
  - Mapeig completament associatiu
    - Una PTE es pot guardar en qualsevol entrada del TLB
  - Cada entrada conté (exemple de MIPS)
    - Còpia de la PTE: PPN, bit V (còpia del bit P), bit D
    - VPN (és l'*etiqueta* que identifica la pàgina)



Traducció ràpida amb TLB: encert

- Quan la MMU busca un VPN al TLB, compara tots els VPN (etiquetes)
- Si algun VPN coincideix, tenim un encert de TLB (TLB hit)
  - El camp PPN és la traducció buscada
  - És un encert de TLB inclús si el bit V=0



Traducció ràpida amb TLB: fallada

- Si cap VPN coincideix, és una fallada de TLB (TLB miss)
- Gestió d'un TLB miss: anar a la TP i copiar la PTE al TLB
  - Ocupar una entrada lliure del TLB (amb el bit V=0)
  - Si no hi ha cap entrada lliure, fer un reemplaçament (LRU, o Random)

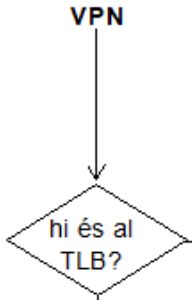
Traducció ràpida amb TLB: fallada

- Si cap VPN coincideix, és una fallada de TLB (TLB miss)
- Gestió d'un TLB miss: anar a la TP i copiar la PTE al TLB
  - Ocupar una entrada lliure del TLB (amb el bit V=0)
  - Si no hi ha cap entrada lliure, fer un reemplaçament (LRU, o Random)
- Alternatives
  - MIPS: gestió per software, la MMU genera una excepció i invoca el S.O.
  - Sparc v8, x86, PowerPC: gestió per hw, la pròpia MMU accedeix a la TP
- Nota 1: el bit V d'una entrada del TLB pot valdre 0 per 2 raons:
  - Entrada lliure, sense inicialitzar
  - L'entrada s'acaba de copiar de la PTE, i el bit de presència era 0 (encara cal resoldre la fallada de pàgina)

Traducció ràpida amb TLB: fallada

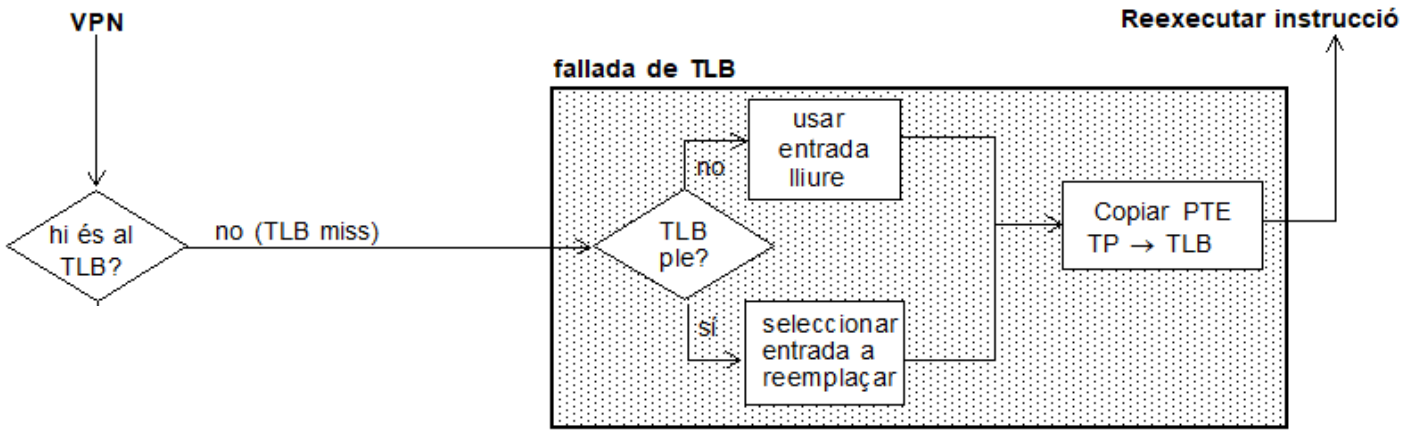
- Si cap VPN coincideix, és una fallada de TLB (TLB miss)
- Gestió d'un TLB miss: anar a la TP i copiar la PTE al TLB
  - Ocupar una entrada lliure del TLB (amb el bit V=0)
  - Si no hi ha cap entrada lliure, fer un reemplaçament (LRU, o Random)
- Alternatives
  - MIPS: gestió per software, la MMU genera una excepció i invoca el S.O.
  - Sparc v8, x86, PowerPC: la pròpia MMU accedeix a la TP
- Nota 1: el bit V d'una entrada del TLB pot valdre 0 per 2 raons:
  - Entrada lliure, sense inicialitzar
  - L'entrada s'acaba de copiar de la PTE, i el bit de presència era 0 (encara cal resoldre la fallada de pàgina)
- Nota 2: el bit D és l'únic del TLB que pot ser modificat pel programa (es posa a 1 quan s'executa un Store)
  - Usem una política d'escriptura immediata (s'escriu D=1 al TLB i a la TP)
  - Però sols cal accedir a la TP en MP el primer cop que escrivim a la pàgina, quan D encara val 0 al TLB

Fallades de TLB i de página: diagrama

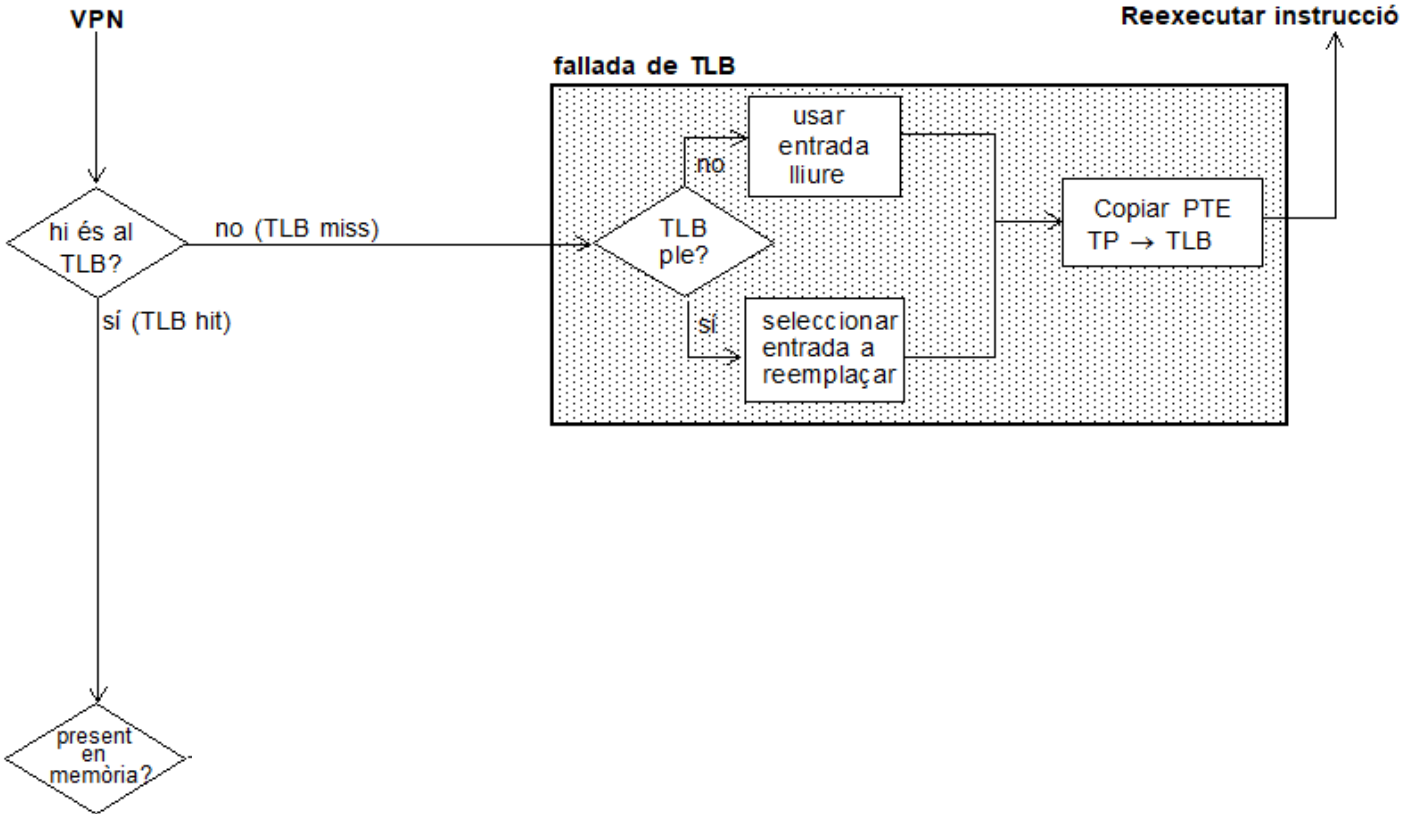


La CPU inicia un accés a memòria

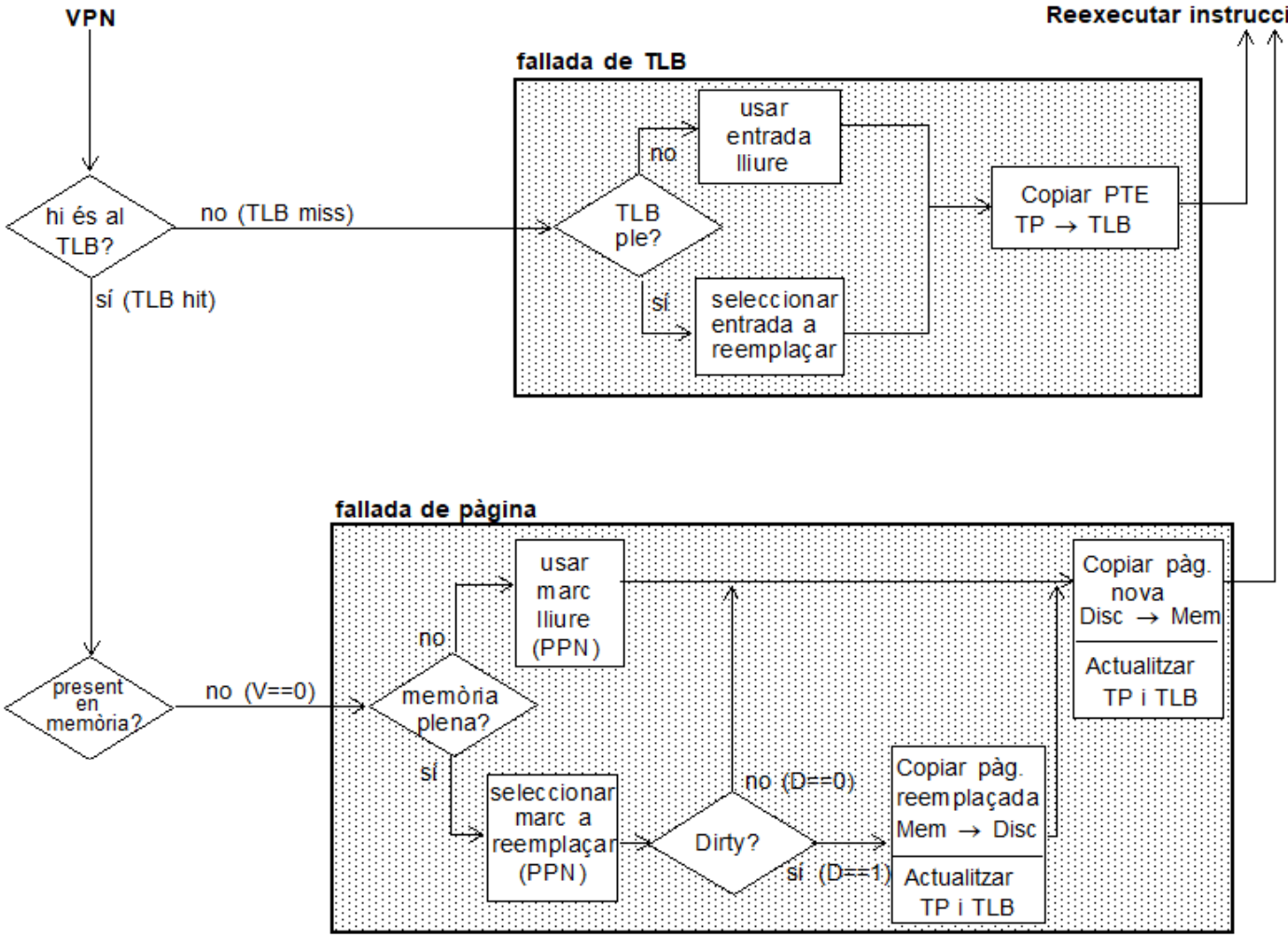
Fallades de TLB i de página: diagrama



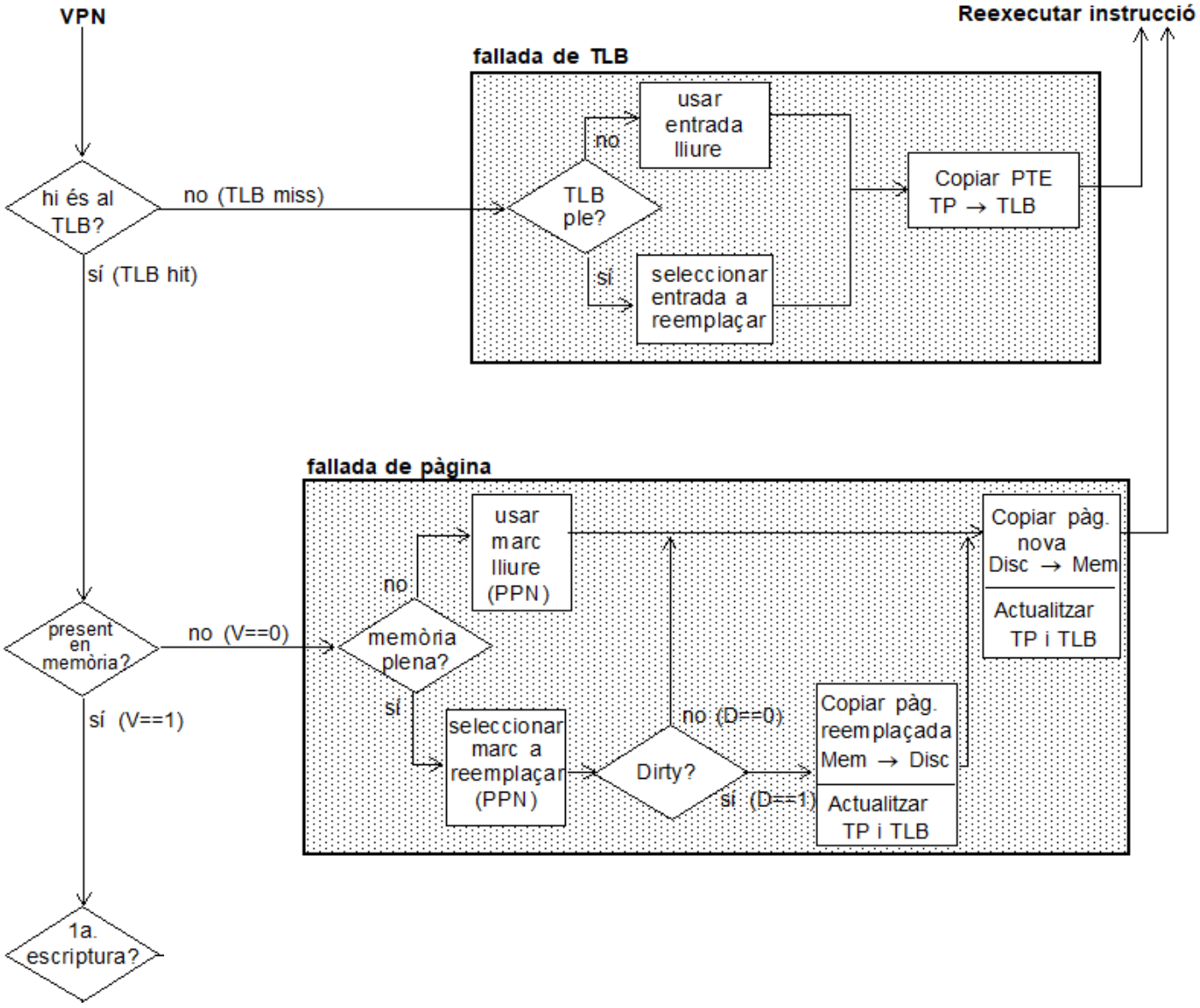
Fallades de TLB i de pàgina: diagrama



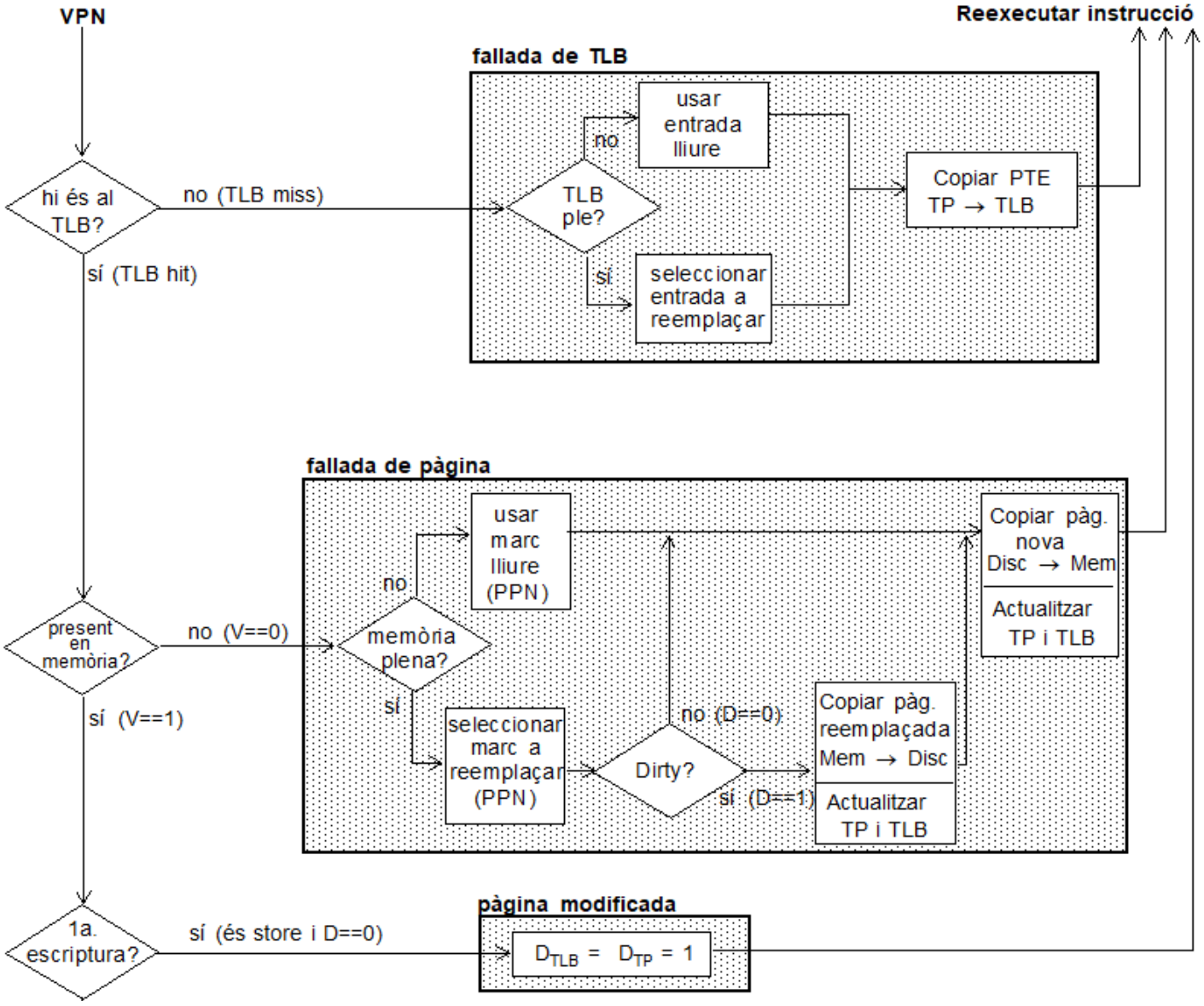
Fallades de TLB i de pàgina: diagrama



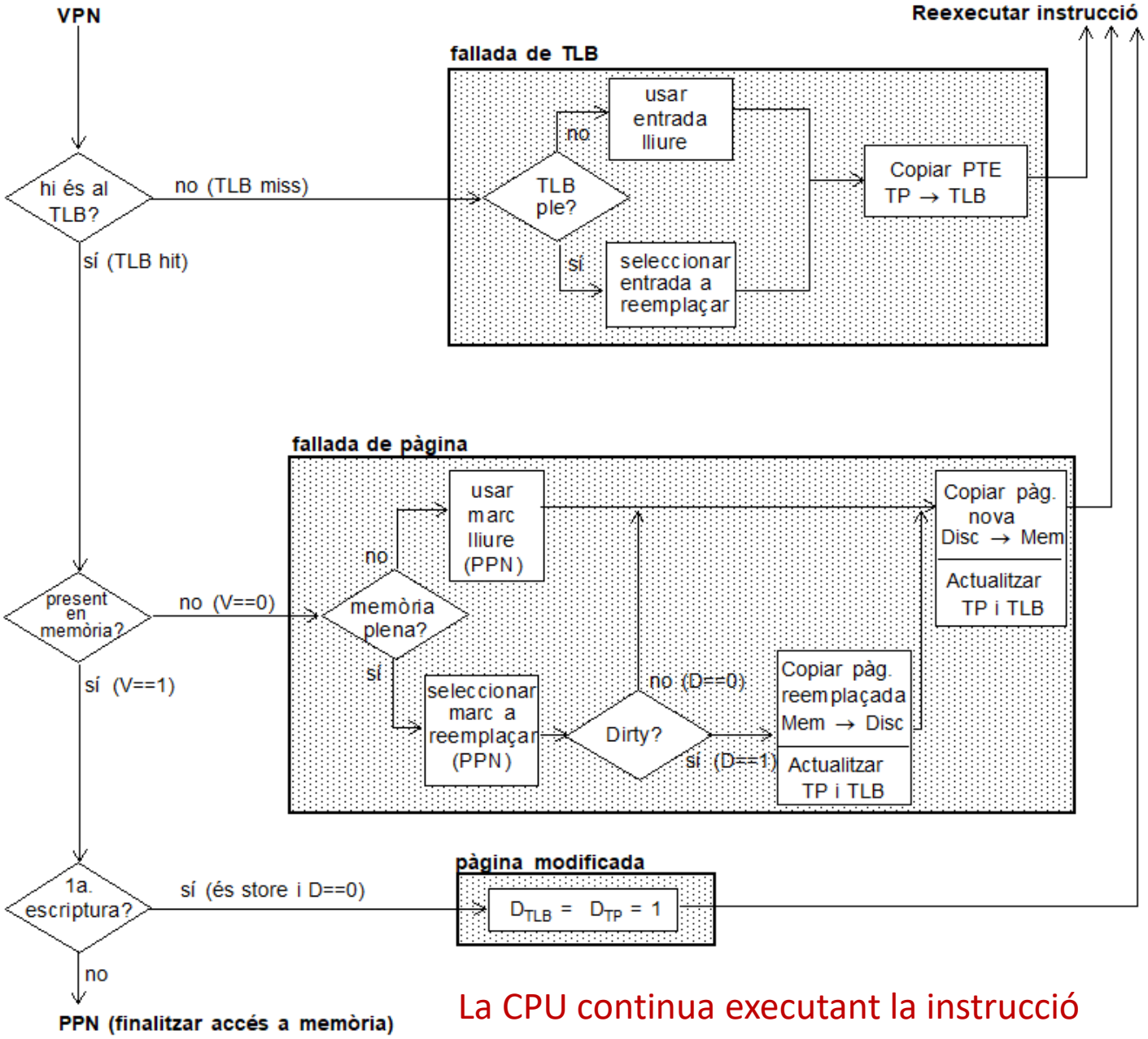
Fallades de TLB i de pàgina: diagrama



Fallades de TLB i de pàgina: diagrama



Fallades de TLB i de pàgina: diagrama



La CPU continua executant la instrucció

Fallades de TLB i de página

- En una fallada de pàgina cal l'accés al disc (milions de cicles)
  - Ho gestiona el Sistema Operatiu (software), i mentre es resol pot donar pas a executar un altre programa (canvi de context)
    → Un cop resolt, la instrucció causant s'ha de reexecutar des d'inici

- En una fallada de TLB cal accés a la TP en MP (*page table walk*)
  - Ho pot gestionar el S.O. (software), com fa MIPS
    → Un cop resolt, la instrucció causant s'ha de reexecutar des d'inici
  - En alguns sistemes (x86, PowerPC), ho resol el hardware
    → Un cop resolt, l'accés a memòria s'ha de reintentar

- Les TP resideixen en memòria
  - Han de ser accessibles per al codi del SO sense traducció d'adreces

Protecció amb Memòria Virtual

- Cada página física está assignada a un únic procés, i no apareix a les taules de pàgines de cap altre procés
  - El mecanisme de traducció fa imposible que un procés accedeixi a pàgines d'un altre procés

Protecció amb Memòria Virtual

- Cada página física está assignada a un únic procés, i no apareix a les taules de pàgines de cap altre procés
  - El mecanisme de traducció fa imposible que un procés accedeixi a pàgines d'un altre procés

- Però ¿què passaria si un procés prova de modificar la seva TP?
  - Les TP es guarden en un espai d'adreces reservat al S.O.
  - En MIPS, són les adreces lògiques amb el bit 31 igual a 1

Protecció amb Memòria Virtual

- Cada página física está assignada a un únic procés, i no apareix a les taules de pàgines de cap altre procés
  - El mecanisme de traducció fa imposible que un procés accedeixi a pàgines d'un altre procés

- Però què passaria si un procés prova de modificar la seva TP?
  - Les TP es guarden en un espai d'adreces reservat al S.O.
  - En MIPS, són les adreces lògiques amb el bit 31 igual a 1

- I com es reserva aquest espai?
  - El procesador té 2 modes de funcionament: usuari i sistema
  - Solament quan funciona en mode sistema té accés al TLB i la TP

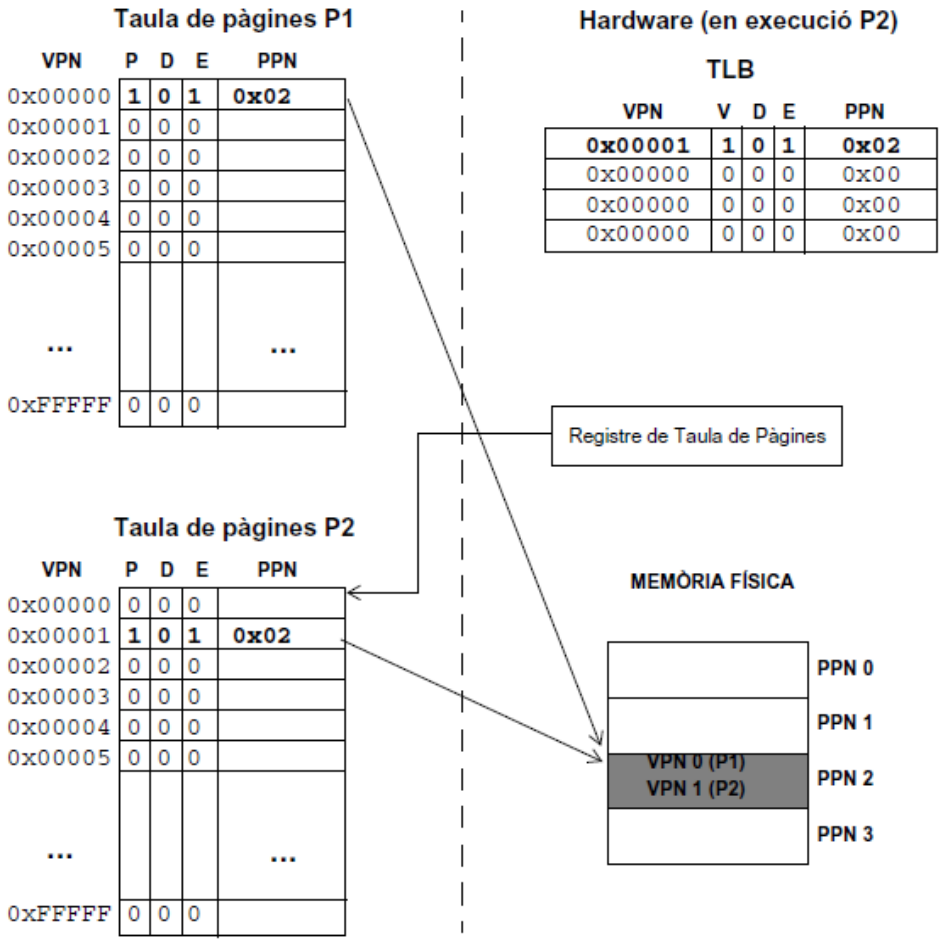
Protecció contra escriptura

- Resulta convenient prohibir l'escriptura en determinades pàgines
  - P. ex., les pàgines de codi: no es permet el codi automodificable
  - S'inclou un bit de permís d'escriptura (bit E) a la TP i al TLB

- I si un procés prova d'escriure en una página amb el bit E=0?
  - Es produeix una excepció, i el S.O. avorta el procés

Compartició de memòria

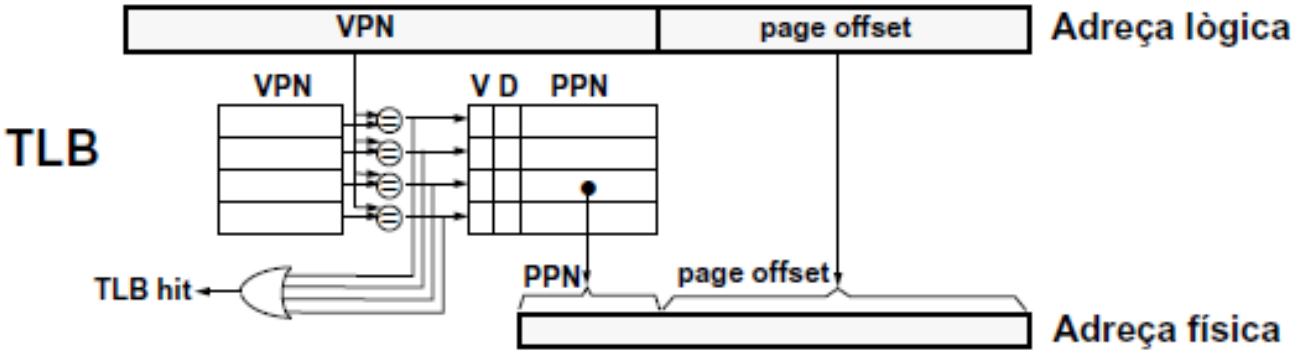
- Un procés P1 pot demanar al S.O. compartir una página amb P2
- El S.O. pot habilitar la compartició
  - S'escriu el PPN de la página compartida a les dues TPs



Exemple:
PPN2 apareix a les 2 TPs

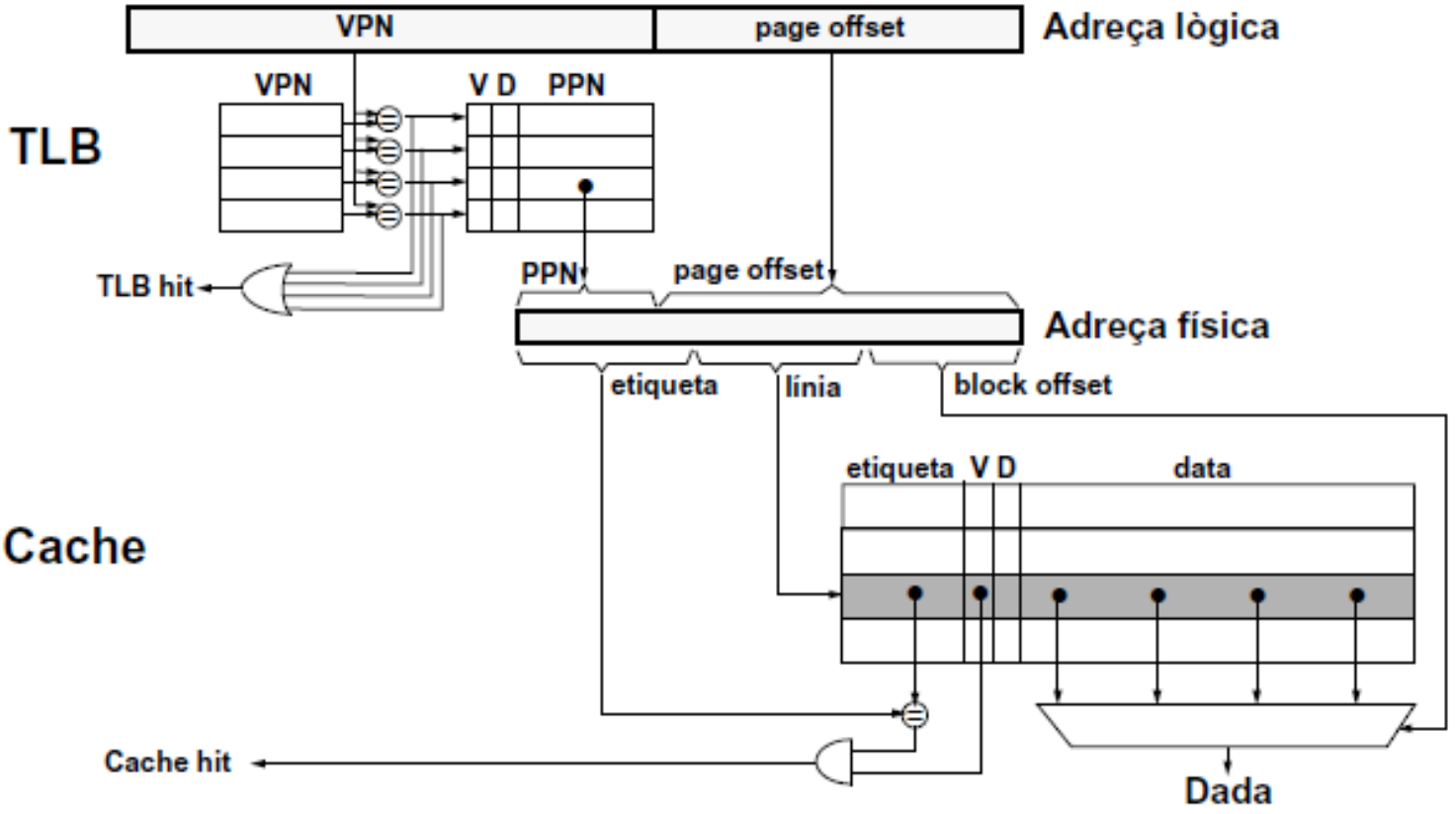
Integració del TLB i la memòria cache

- Memòria cache **indexada físicament**
  - Simple, però té un temps d'accés elevat (s'accedeix seqüencialment a TLB i cache)



Integració del TLB i la memòria cache

- Memòria cache **indexada físicament**
  - Simple, però té un temps d'accés elevat (s'accedeix seqüencialment a TLB i cache)



Test de repàs: Veritat o Fals?

1. Si l'accés a dades d'una instrucció produeix un encert al TLB, però el bit V val 0, llavors la instrucció causarà una excepció de fallada de pàgina

2. Una mateixa instrucció pot causar durant la seva execució 2 fallades de pàgina

3. Una fallada al TLB no implica que hi hagi una fallada de pàgina

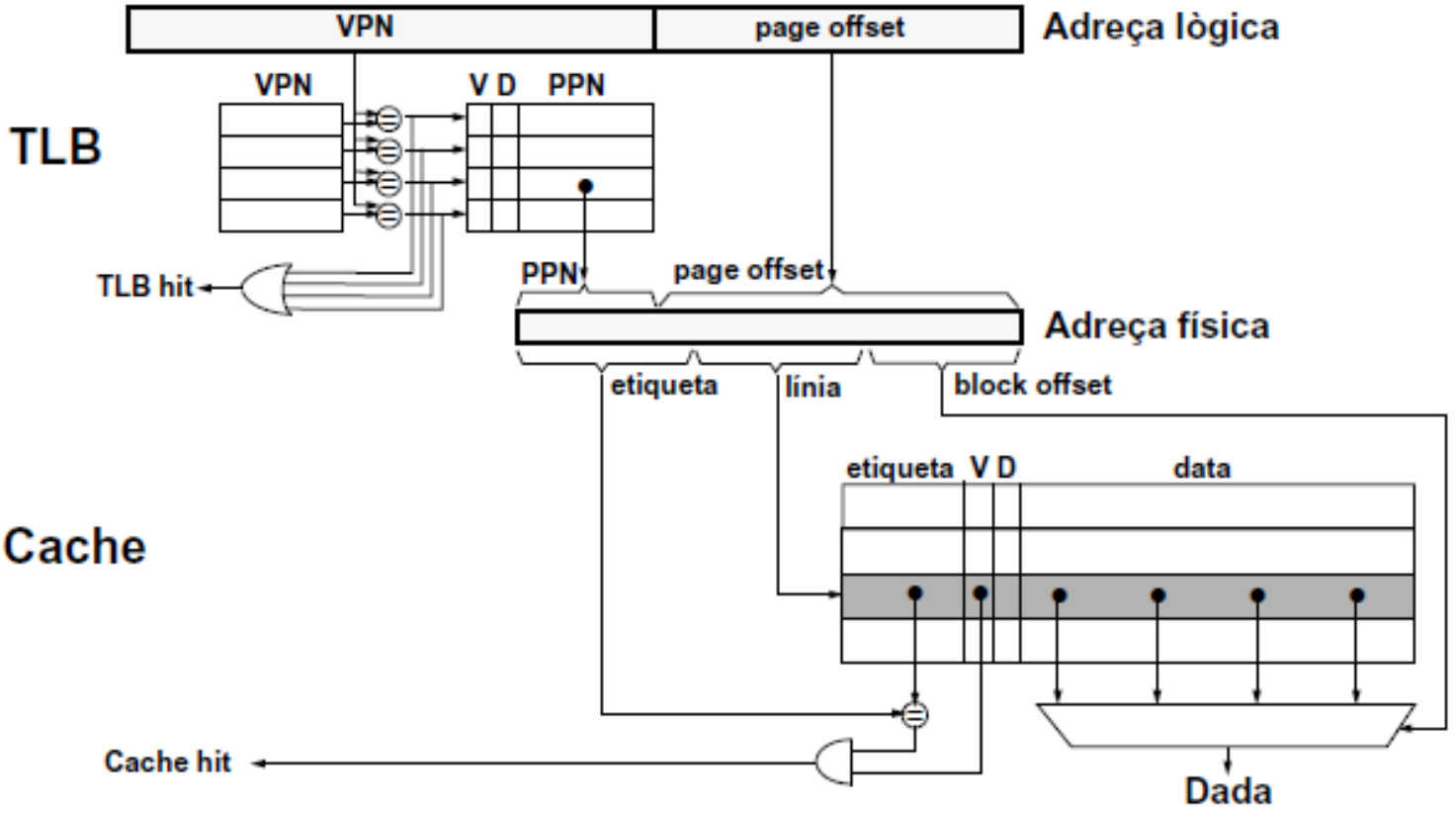
4. En memòria virtual paginada, sempre que reemplacem una pàgina de la memòria física cal escriure-la en disc

Annex (no entra)

- Integració de TLB i caches
- Taula de Pàgines multinivell

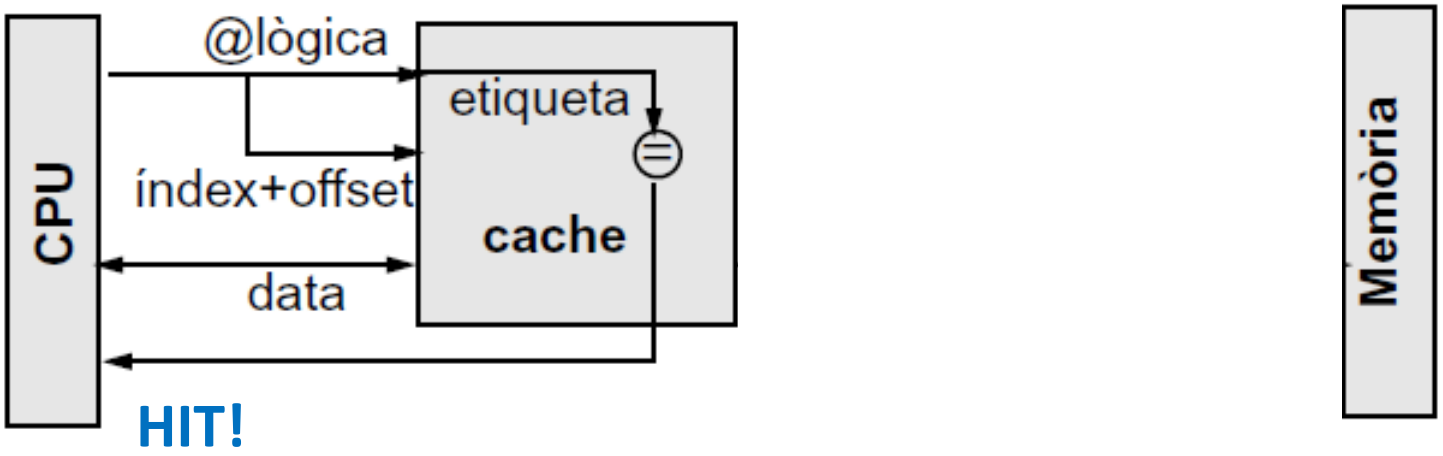
Integració del TLB i la memòria cache

- Memòria cache **indexada físicament**
  - Simple, però té un temps d'accés elevat (s'accedeix seqüencialment a TLB i cache)



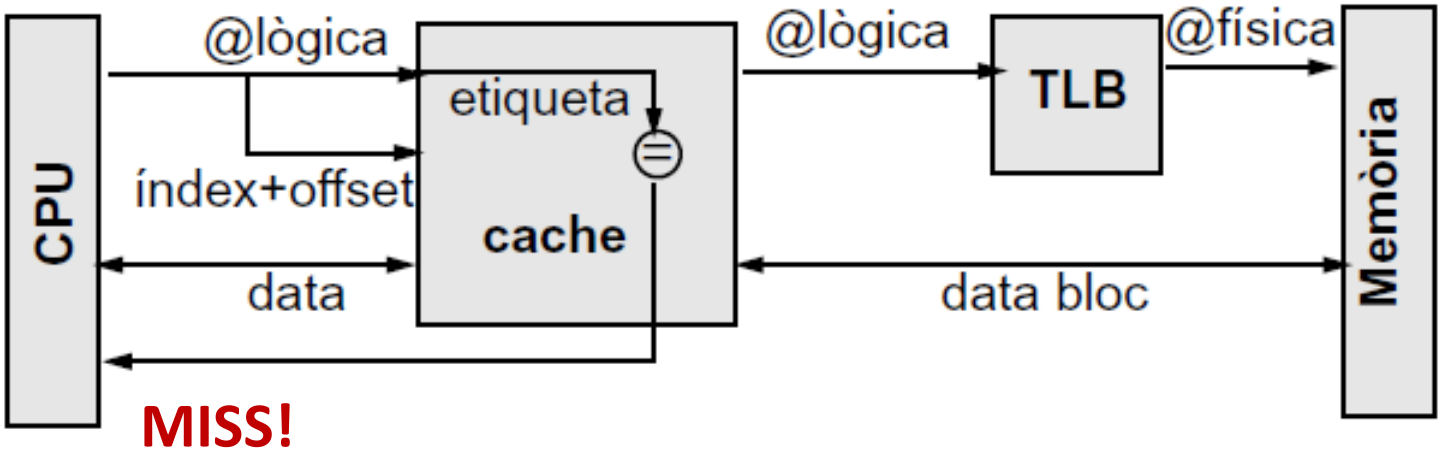
Integració del TLB i la memòria cache

- Memòria cache **indexada virtualment**
  - Temps d'accés menor: no cal traduir l'adreça en cas d'encert



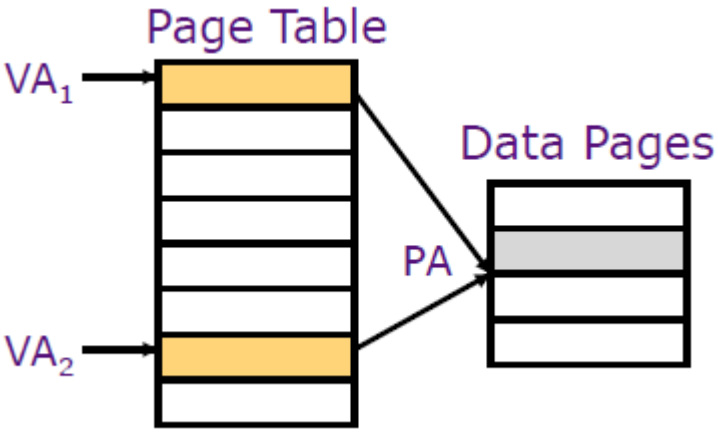
Integració del TLB i la memòria cache

- Memòria cache **indexada virtualment**
  - Temps d'accés menor: no cal traduir l'adreça en cas d'encert, només en cas de fallada



Integració del TLB i la memòria cache

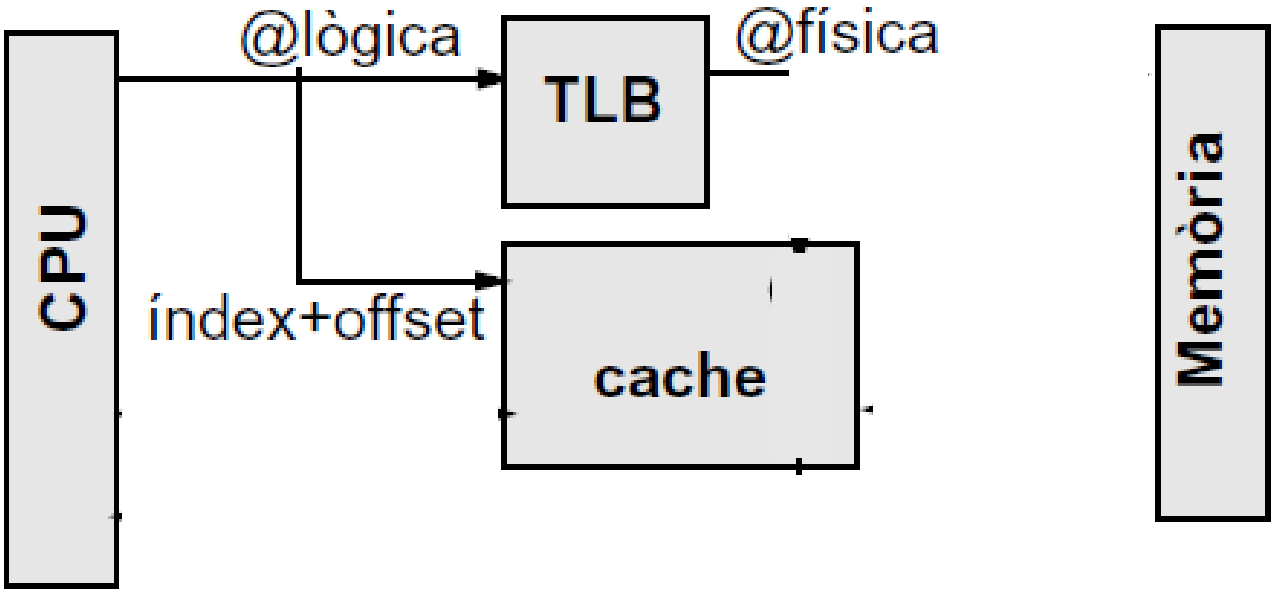
- Memòria cache **indexada virtualment**
  - Problema de l'*aliasing*: pàgines compartides poden quedar duplicades en línies diferents
    - el que s'escriu en una còpia no és visible a les lectures de l'altra



Etiqueta	blocs de dades
VA_1	1st Copy of Data at PA
VA_2	2nd Copy of Data at PA

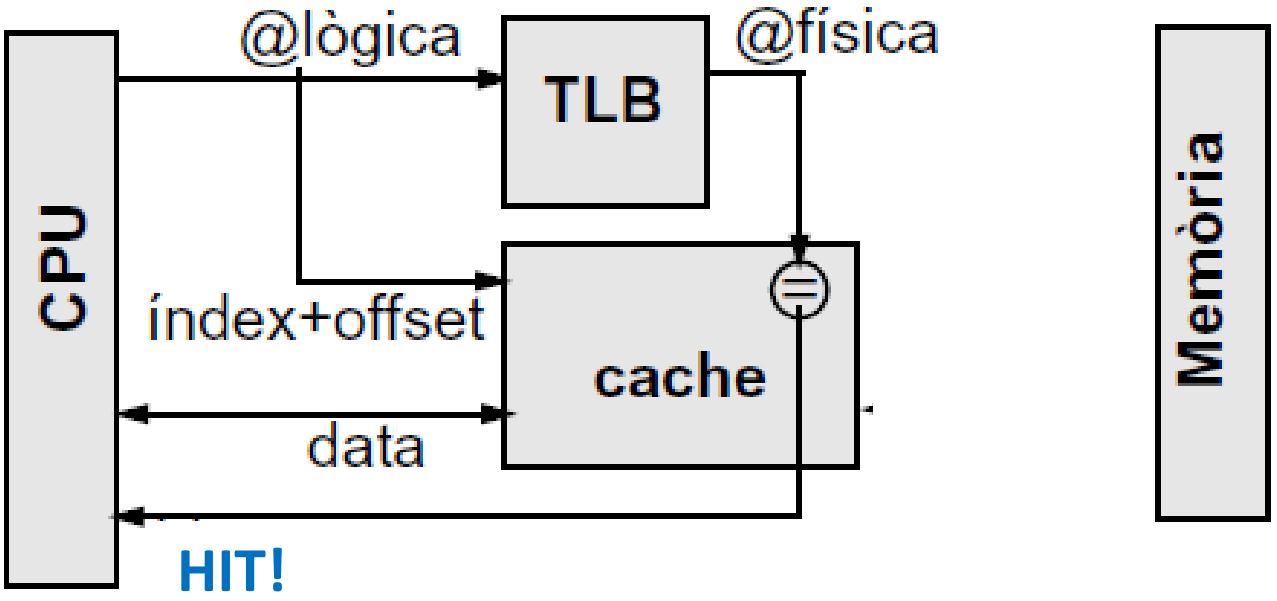
Integració del TLB i la memòria cache

- Memòria cache **indexada virtualment i etiquetada físicament**
  - S'inicia l'accés amb l'adreça virtual: l'índex de línea o conjunt s'obté dels bits de page-offset, que no s'han de traduir
  - En paral·lel, s'accedeix al TLB i s'obté l'adreça física



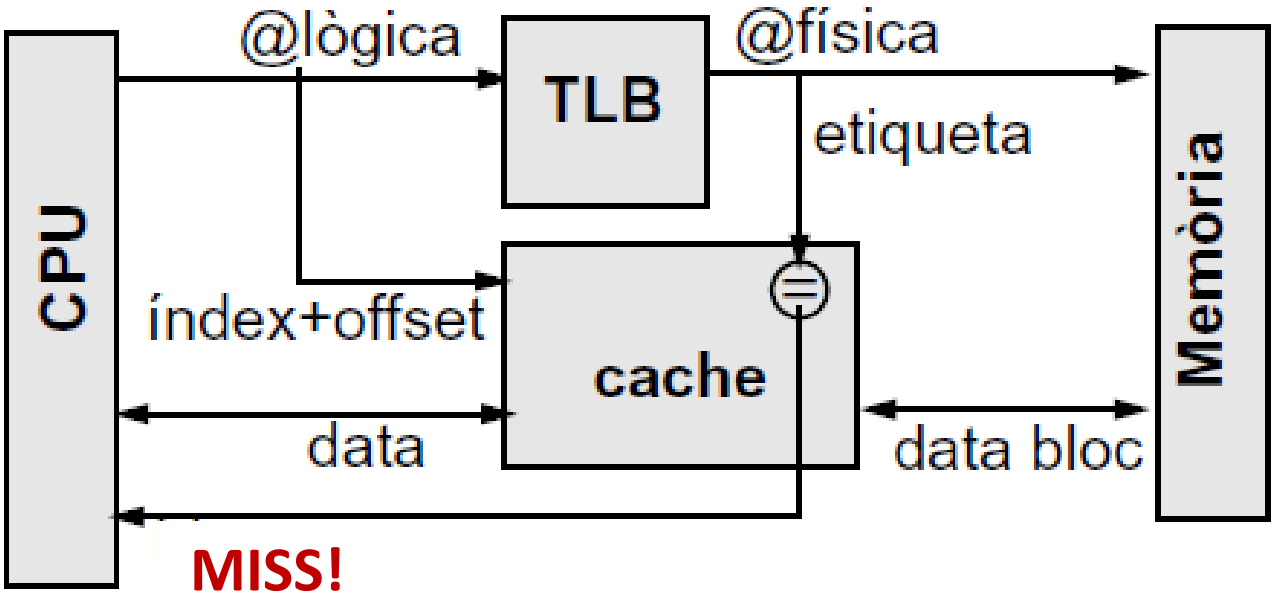
Integració del TLB i la memòria cache

- Memòria cache **indexada virtualment i etiquetada físicament**
  - S'inicia l'accés amb l'adreça virtual: l'índex de línea o conjunt s'obté dels bits de page offset, que no s'han de traduir
  - En paral·lel s'accedeix al TLB, i s'obté l'adreça física
  - Finalment es comproven les etiquetes amb l'adreça física



Integració del TLB i la memòria cache

- Memòria cache **indexada virtualment i etiquetada físicament**
  - S'inicia l'accés amb l'adreça virtual: l'índex de línea o conjunt s'obté dels bits de page offset, que no s'han de traduir
  - En paral·lel s'accedeix al TLB, i s'obté l'adreça física
  - Finalment es comproven les etiquetes amb l'adreça física

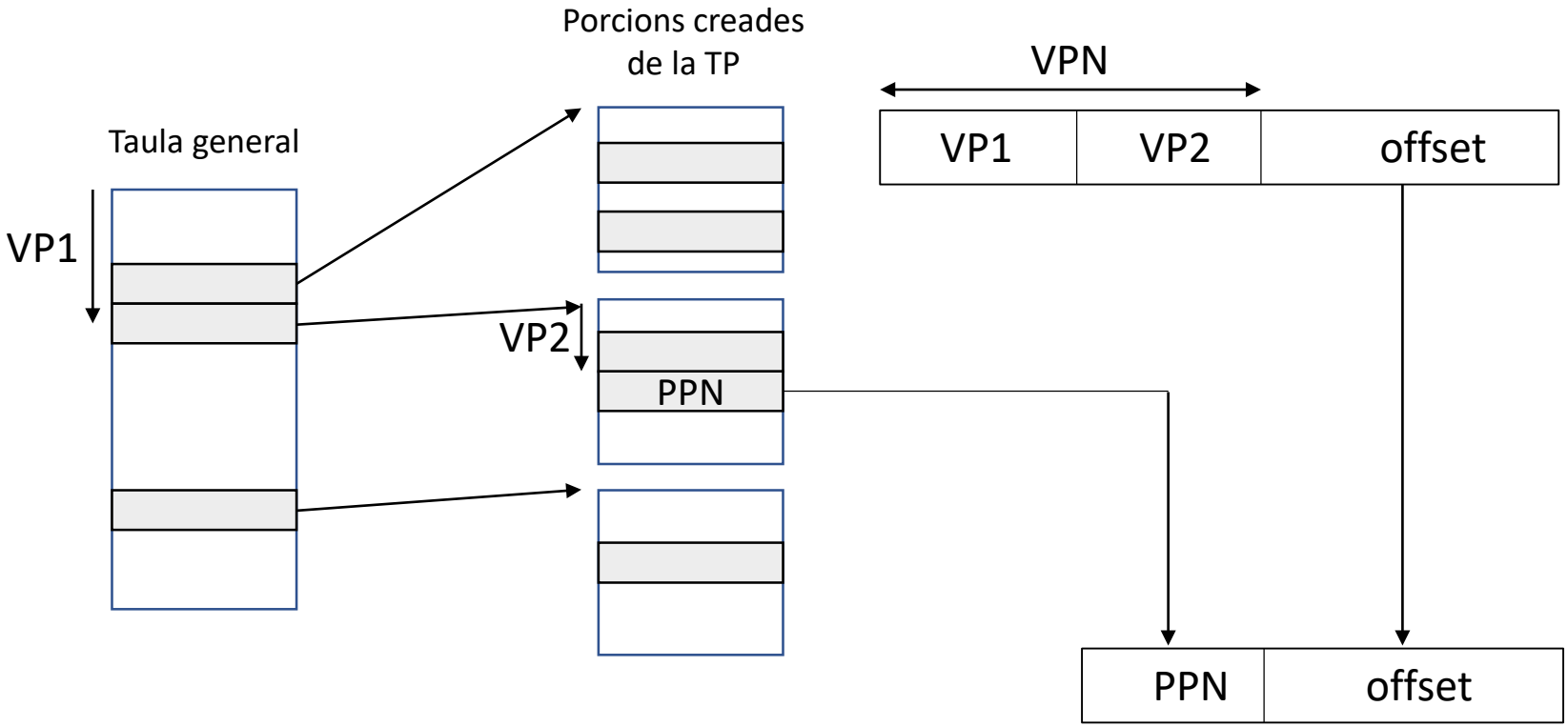


TP multinivell

- Vegem el **problema,** amb quatre números
- Suposem:
  - Espai lògic de 64 bits
  - Memòria física de 43 bits
  - Pàgines de 8KB = 2^{13} bytes
- Quanta memòria cal per implementar la TP?
  - Offset de 13 bits
  - VPN de 64-13 bits = 51 bits → TP de 2^{51} entrades
  - PPN de 43-13 = 30 bits
  - Cada entrada té: PPN (30 bits) + P (1 bit) + D (1 bit) = 32 bits = 4 bytes
- En total: 2^{53} bytes = 8 Petabytes (!!!)

TP multinivell

- Calen tècniques especials: taules multinivell
  - Observació: cada procés sol tenir unes poques pàgines vàlides
  - Dividim la TP en pàgines o porcions, i sols aquelles amb alguna entrada vàlida es poden crear (i potser mantenir en MP)
  - Una taula general conté punters a les porcions creades de la TP



PT multinivell: exemple