Tema 7. Memòria Virtual

Joan Manuel Parcerisa

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Memòria Virtual

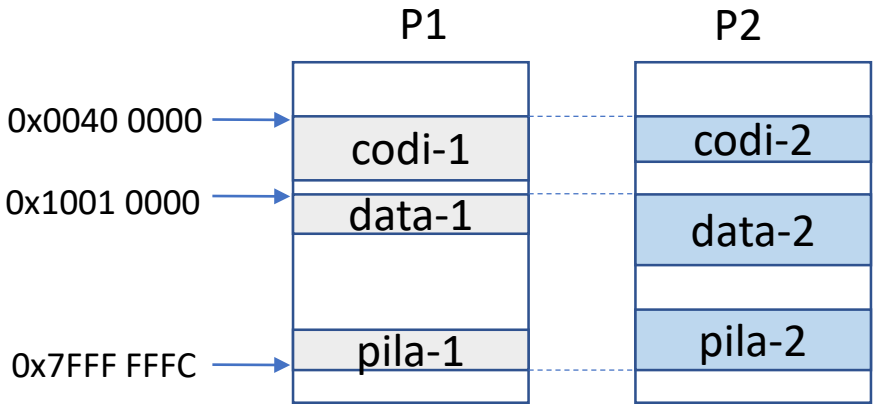
- Introducció
- Memòria virtual paginada
- Traducció ràpida amb TLB

Memòria Virtual

- Introducció
  - Requeriments de reubicació, protecció i capacitat
  - Traducció d'adreces
  - Jerarquia de memòria
- Memòria virtual paginada
- Traducció ràpida amb TLB

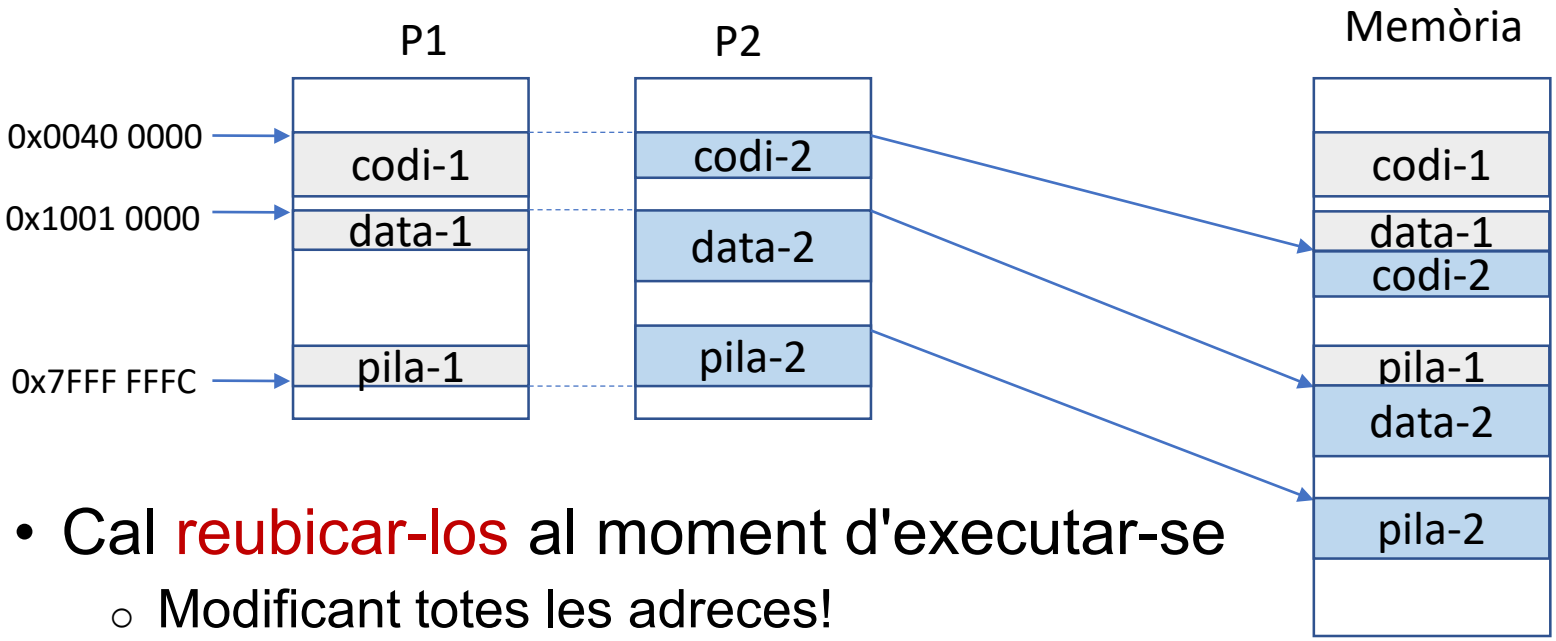
Necessitat de reubicació

- 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!

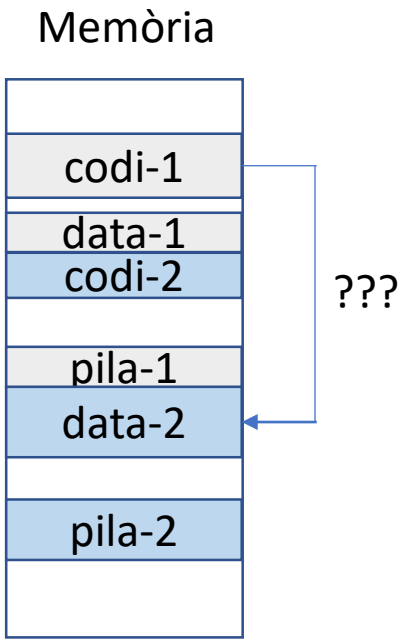


- Cal reubicar-los al moment d'executar-se
  - Modificant totes les adreces!

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 instal·lada?

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 instal·lada?

    → El programa ha d'estar parcialment carregat en memòria

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 instal·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

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 instal·lada?

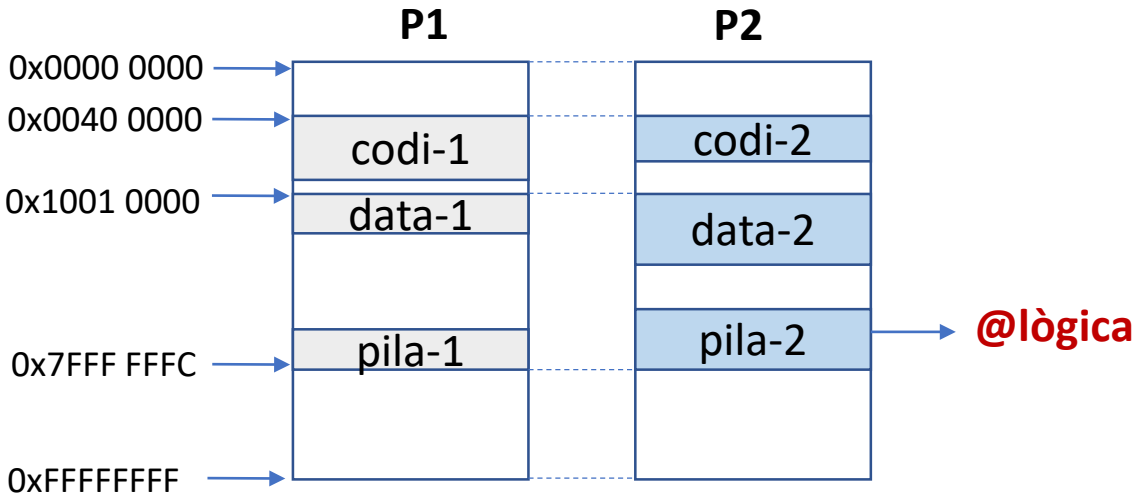
    → El programa ha d'estar parcialment carregat en memòria

- 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

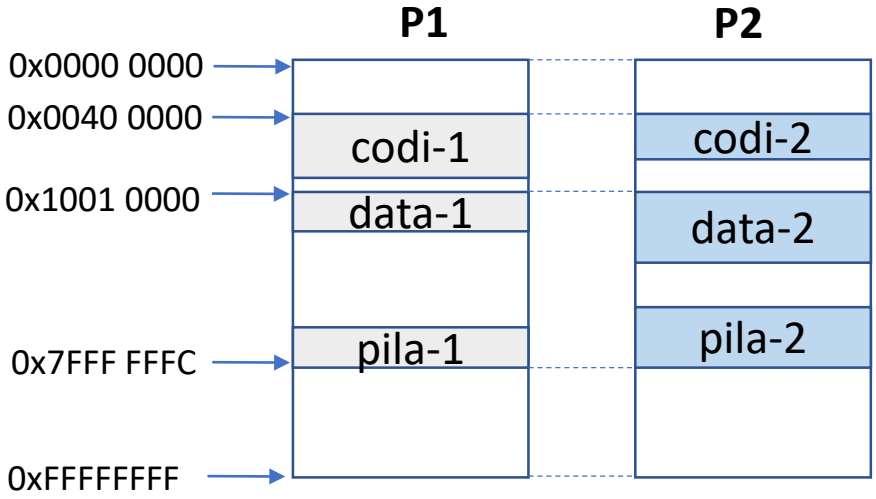
Solució MV: traducció d'adreces

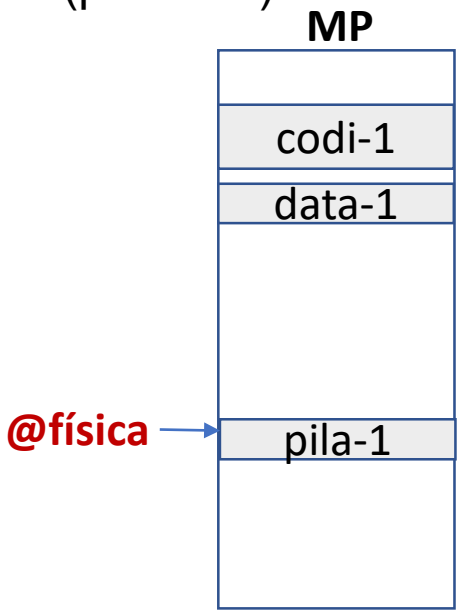
- Espai d'adreçament lògic
  - Són les adreces que assigna el compilador a cada programa, i que la CPU envia a la memòria en cada accés
  - Cada programa té el seu propi espai, mateixa mida (p.ex. 2^{32})



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})

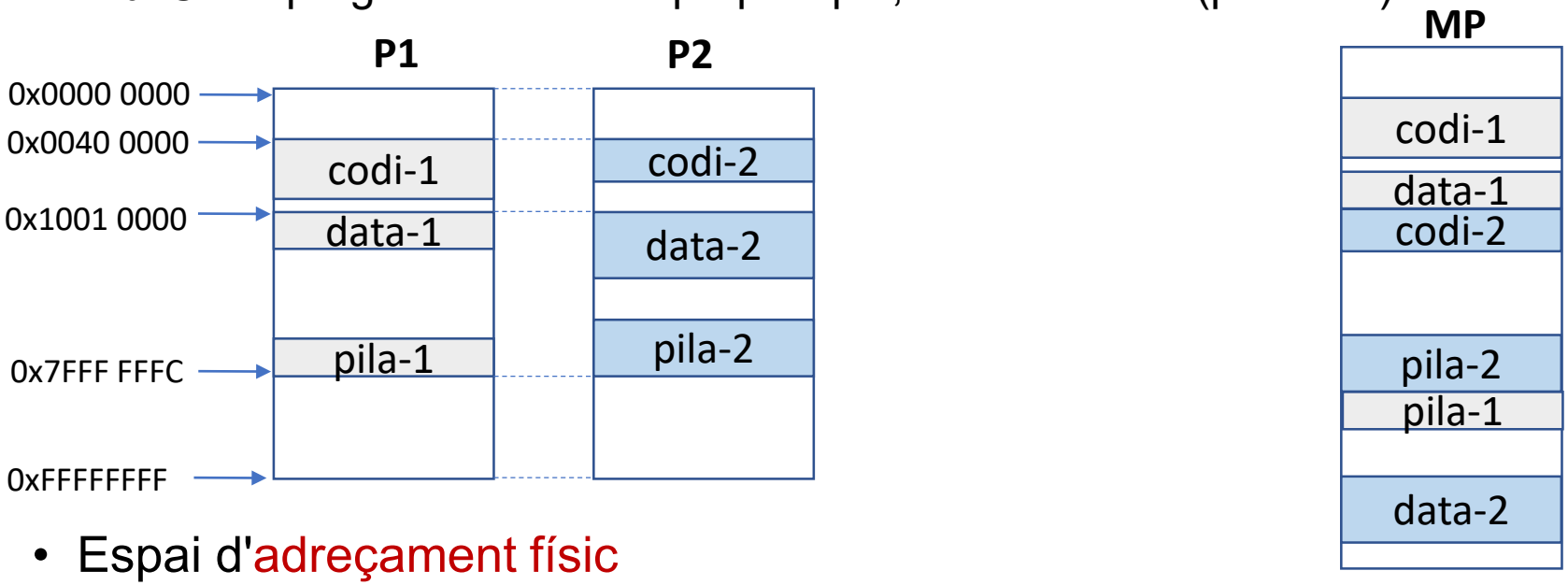




- Espai d'adreçament físic
  - Són les adreces "reals" que rep la MP

Solució MV: traducció d'adreces

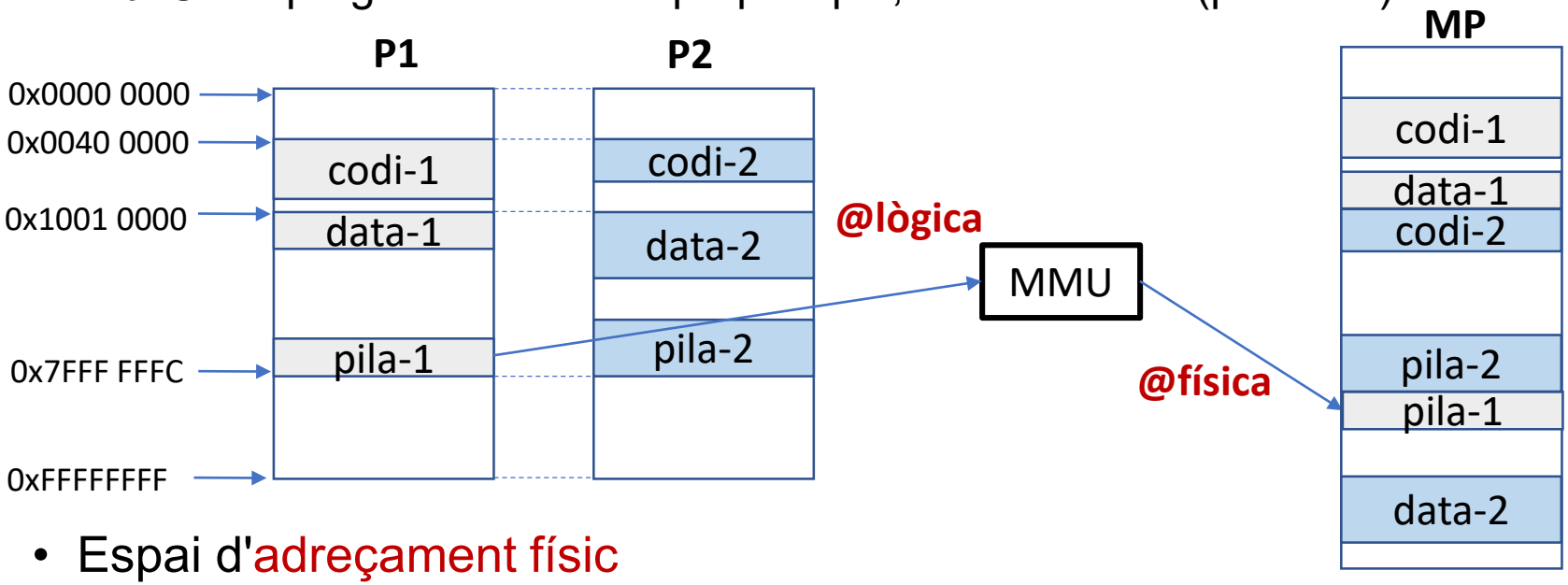
- Espai d'adreçament lògic
  - Són les adreces que assigna el compilador a cada programa, i que la CPU envia a la memòria en cada accés
  - Cada programa té el seu propi espai, mateixa mida (p.ex. 2^{32})



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

Solució MV: traducció d'adreces

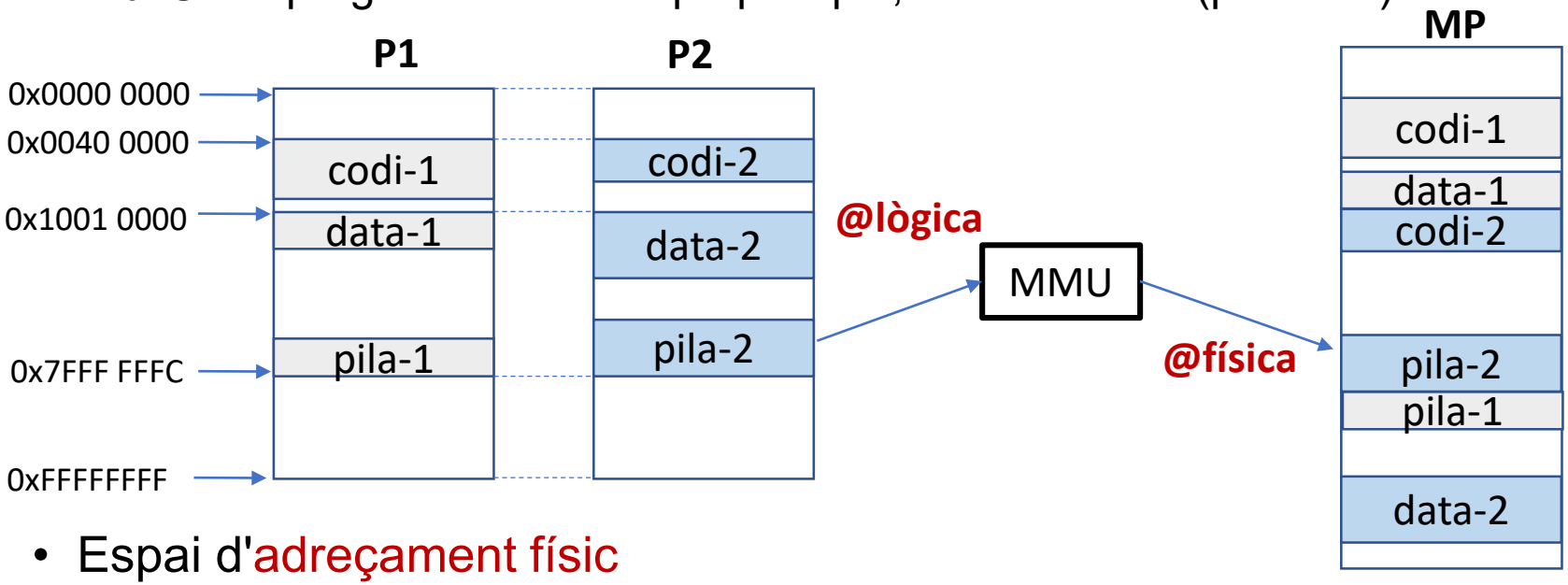
- Espai d'adreçament lògic
  - Són les adreces que assigna el compilador a cada programa, i que la CPU envia a la memòria en cada accés
  - Cada programa té el seu propi espai, mateixa mida (p.ex. 2^{32})



- 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 MMU fa la **traducció** de cada adreça lògica a física
    - recorda en quines adreces físiques s'ha ubicat cada porció

Solució MV: traducció d'adreces

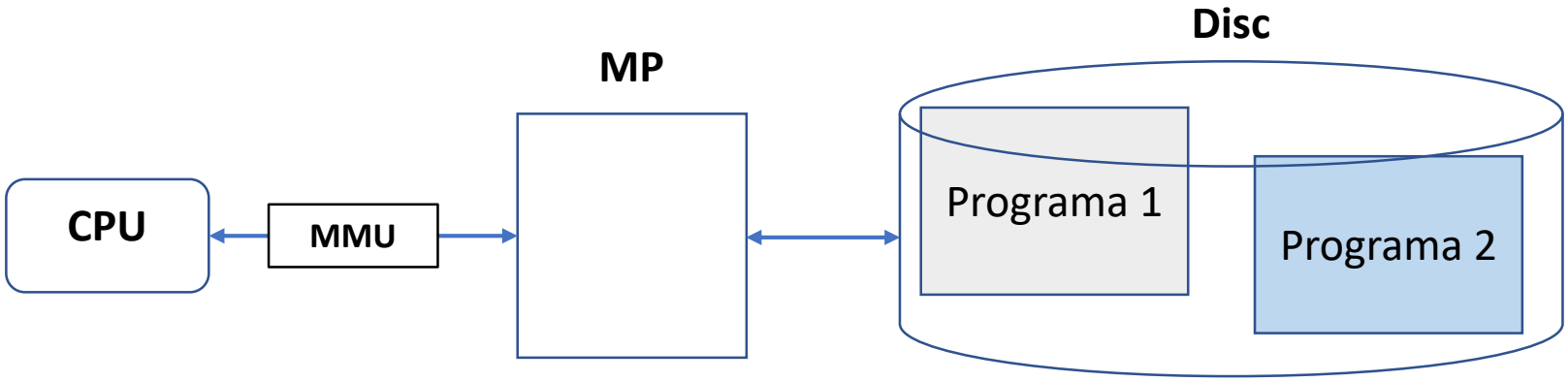
- Espai d'adreçament lògic
  - Són les adreces que assigna el compilador a cada programa, i que la CPU envia a la memòria en cada accés
  - Cada programa té el seu propi espai, mateixa mida (p.ex. 2^{32})



- 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 MMU fa la **traducció** de cada adreça lògica a física
    - recorda en quines adreces físiques s'ha ubicat cada porció

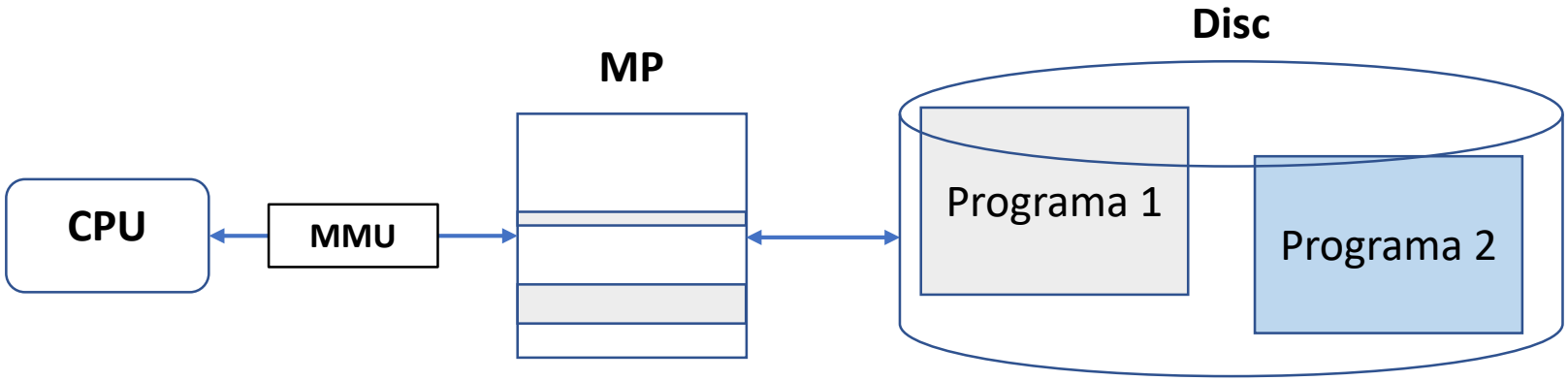
MV: Programes parcialment carregats en MP

- Els programes resideixen íntegrament al disc



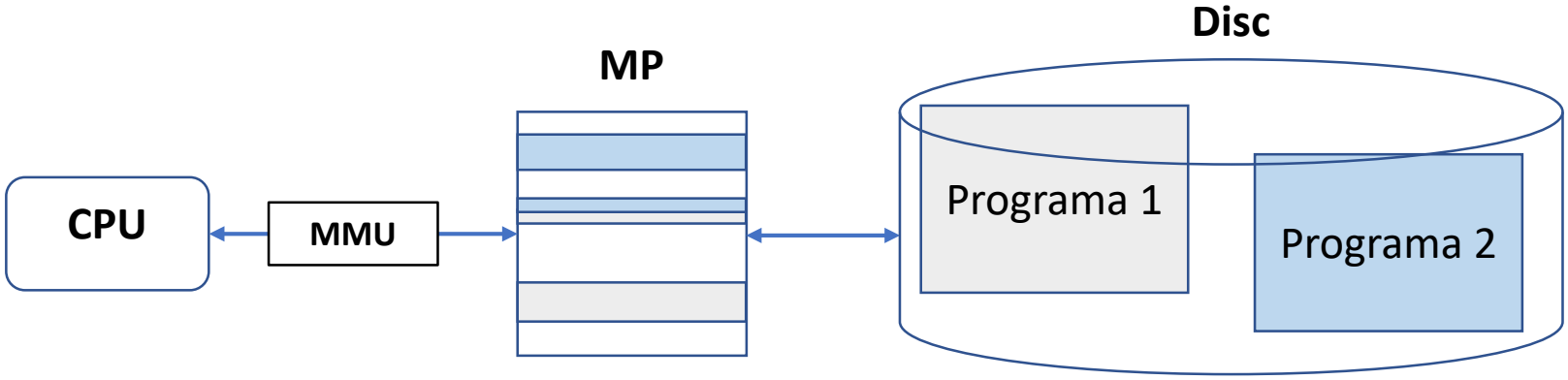
MV: Programes parcialment carregats en MP

- Els programes resideixen íntegrament al disc
- Les porcions accedides recentment es guarden en MP



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



Vist així, la MP sembla una cache del Disc!!

Memòria Virtual: una solució elegant

- Traducció d'adreces lògiques a físiques
  - Permet reubicació automàtica
  - Proporciona protecció i compartició entre programes

Memòria Virtual: una solució elegant

- Traducció d'adreces lògiques a físiques
  - Permet reubicació automàtica
  - Proporciona protecció i compartició entre programes

- Usa la MP com una "cache" del disc
  - Però amb gestió conjunta de hardware (MMU) i Sistema Operatiu

Memòria Virtual: una solució elegant

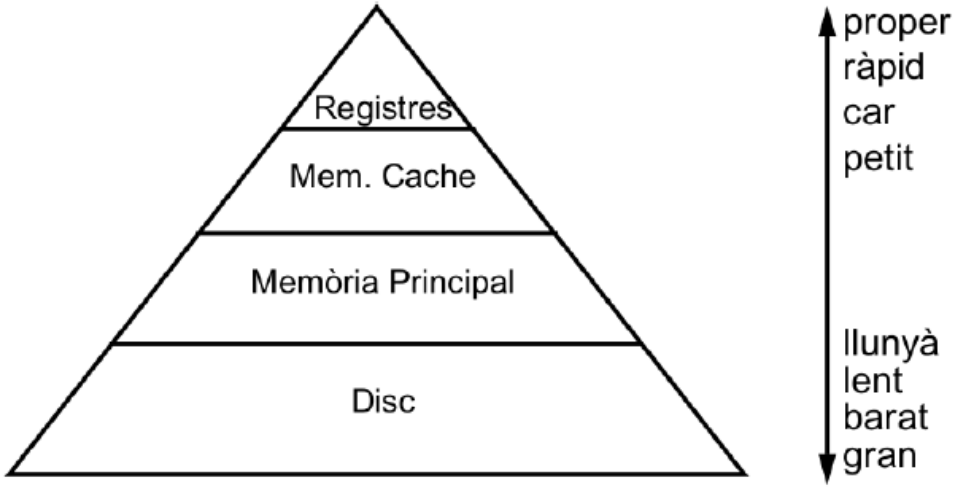
- Traducció d'adreces lògiques a físiques
  - Permet reubicació automàtica
  - Proporciona protecció i compartició entre programes

- Usa la MP com una "cache" del disc
  - Però amb gestió conjunta de hardware (MMU) i Sistema Operatiu
  - Permet excedir la capacitat de la MP de forma transparent al programador

Memòria Virtual: una solució elegant

- Traducció d'adreces lògiques a físiques
  - Permet reubicació automàtica
  - Proporciona protecció i compartició entre programes

- Usa la MP com una "cache" del disc
  - Però amb gestió conjunta de hardware (MMU) i Sistema Operatiu
  - Permet excedir la capacitat de la MP de forma transparent al programador
  - MP esdevé un nivell més de la jerarquia de memòria



Memòria Virtual

- Introducció
- Memòria virtual paginada
- Traducció ràpida amb TLB

Memòria Virtual

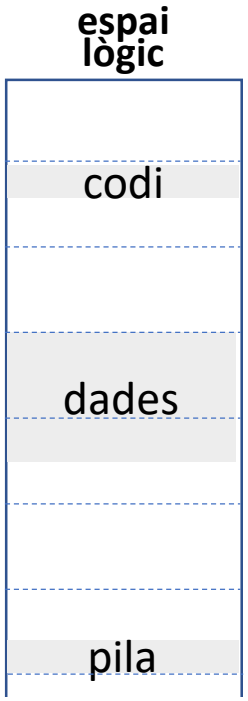
- Introducció
- Memòria virtual paginada
  - Organització en pàgines
  - Traducció d'adreces
  - Polítiques d'emplaçament, reemplaçament i escriptura
  - Traducció amb Taula de Pàgines
- Traducció ràpida amb TLB

Organització en pàgines

- Espai d'adreçament **lògic o virtual**
  - És el format per les adreces determinades pel compilador i l'enllaçador

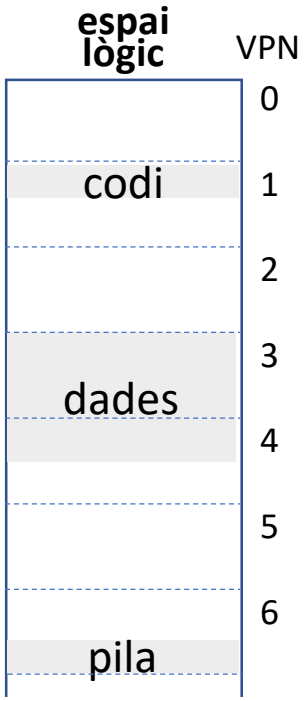
Organització en pàgines

- 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"



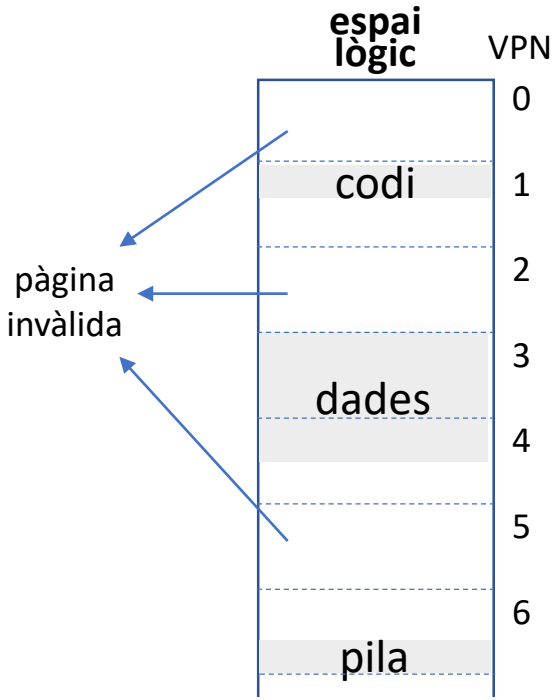
Organització en pàgines

- 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



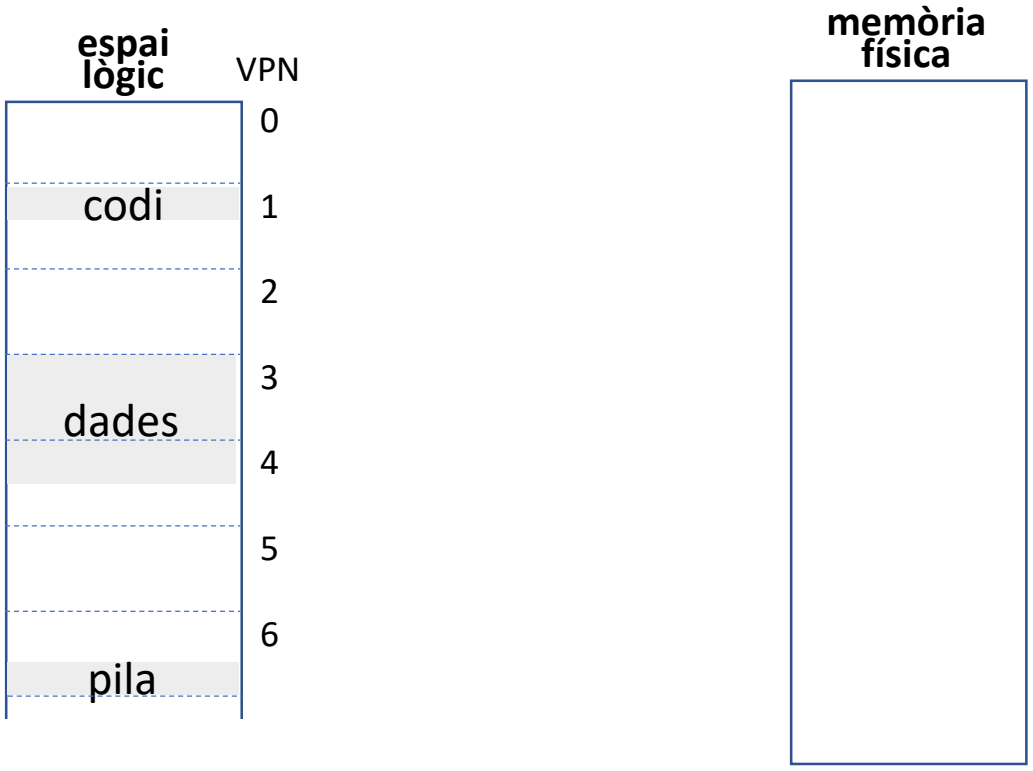
Organització en pàgines

- 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



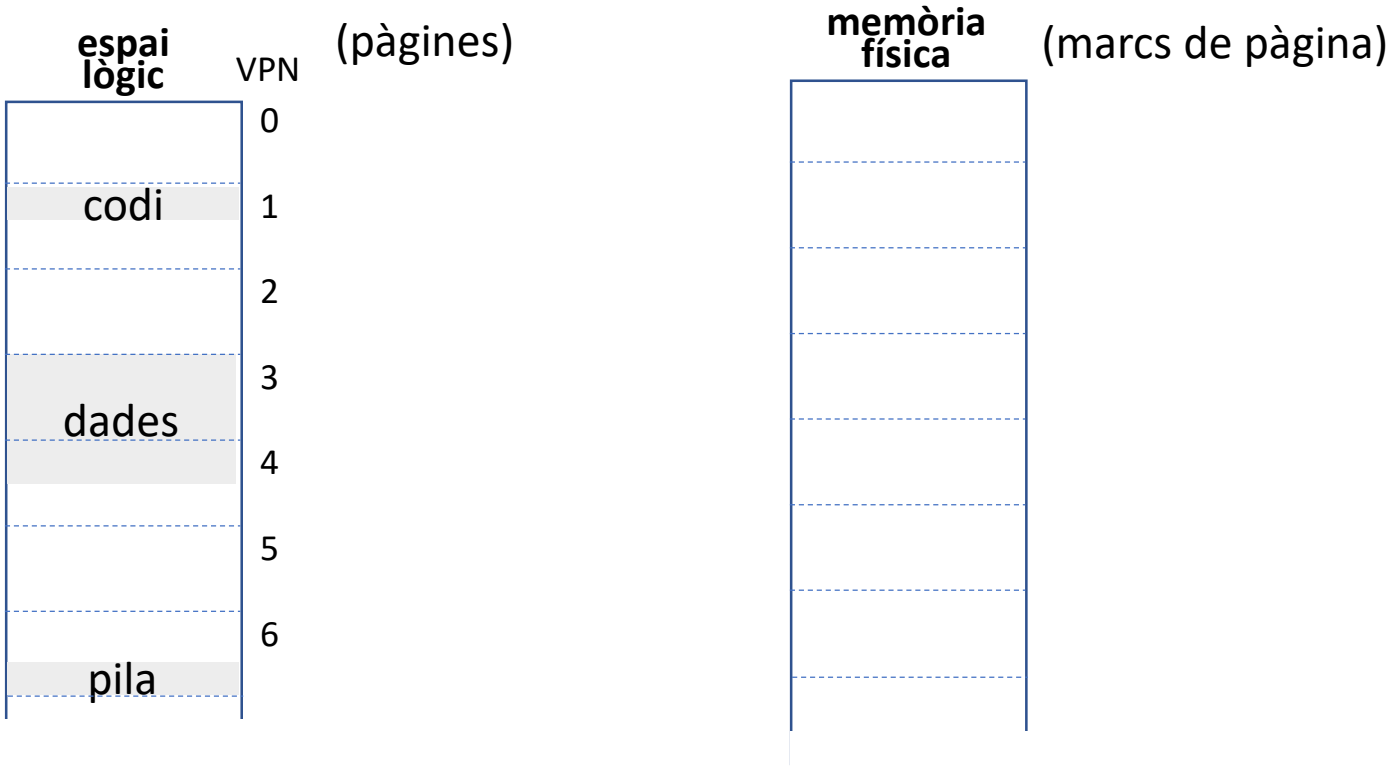
Organització en pàgines

- Espai d'adreçament **físic**
  - Depèn de la memòria RAM instal·lada (p.ex. 4GB, 32GB, etc.)



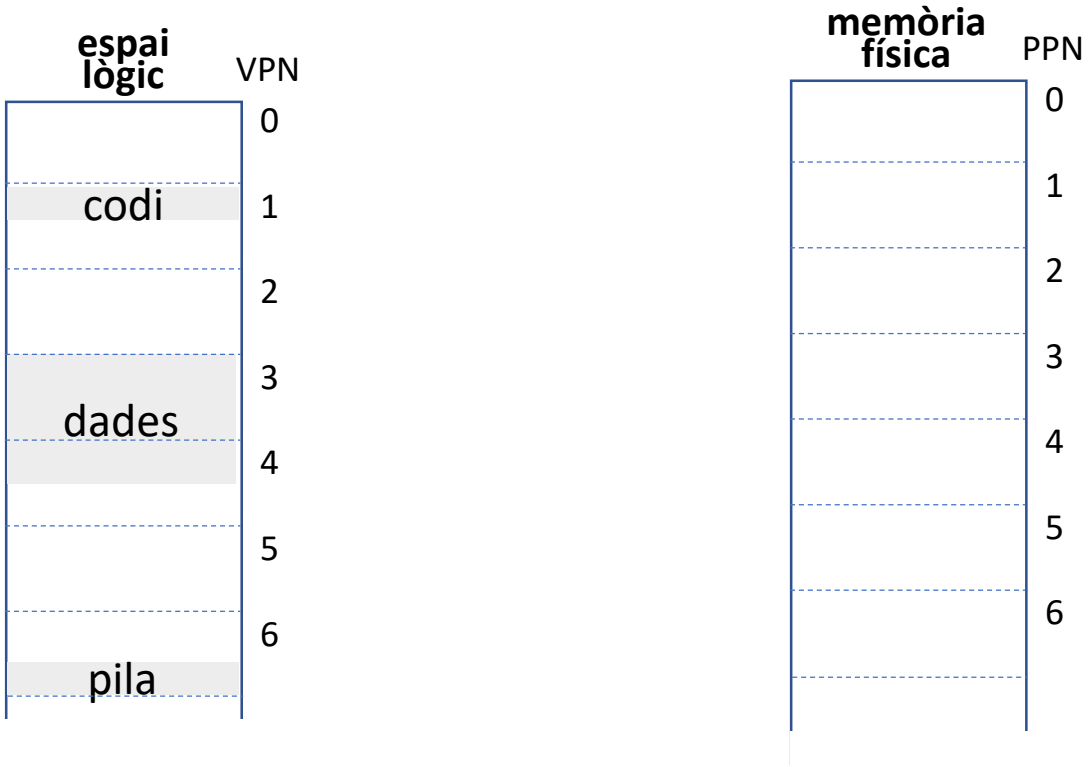
Organització en pàgines

- Espai d'adreçament **físic**
  - Depèn de la memòria RAM instal·lada (p.ex. 4GB, 32GB, etc.)
- Es divideix en **marcs de pàgina**
  - De la mateixa mida que les pàgines



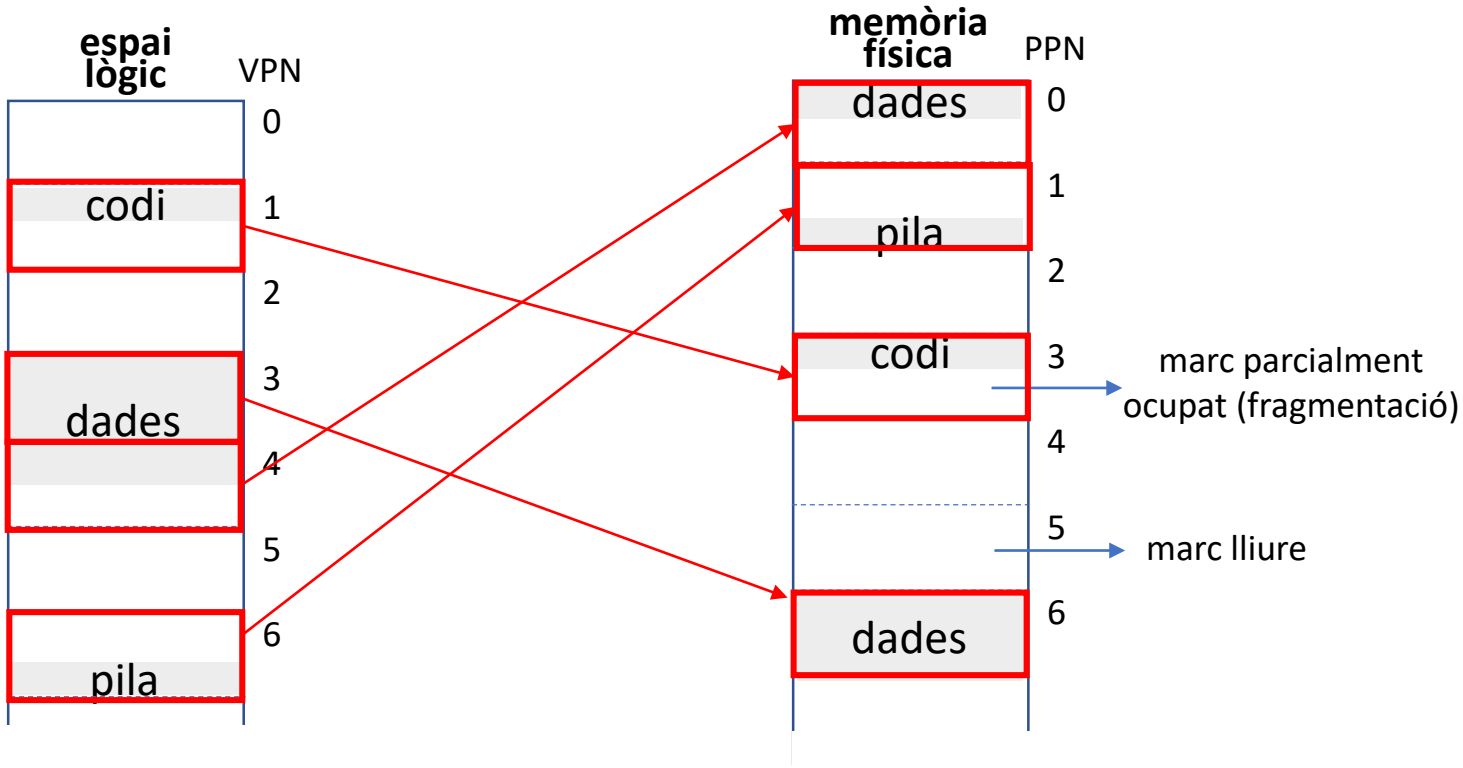
Organització en pàgines

- Espai d'adreçament **físic**
  - Depèn de la memòria RAM instal·lada (p.ex. 4GB, 32GB, etc.)
- Es divideix en **marcs de pàgina**
  - De la mateixa mida que les pàgines
  - Es numeren amb el **PPN** (Physical Page Number), per identificar-los



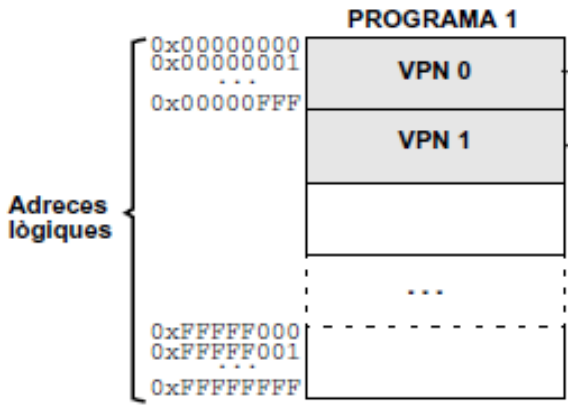
Organització en pàgines

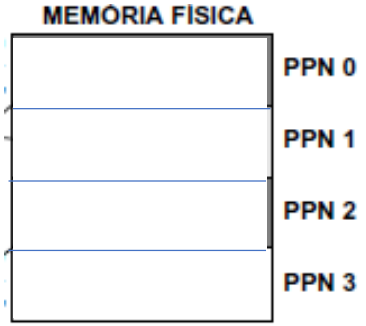
- Espai d'adreçament **físic**
  - Depèn de la memòria RAM instal·lada (p.ex. 4GB, 32GB, etc.)
- Es divideix en **marcs de pàgina**
  - De la mateixa mida que les pàgines
  - Es numeren amb el **PPN** (Physical Page Number), per identificar-los
    - Un marc és com un "contenidor", que pot allotjar una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

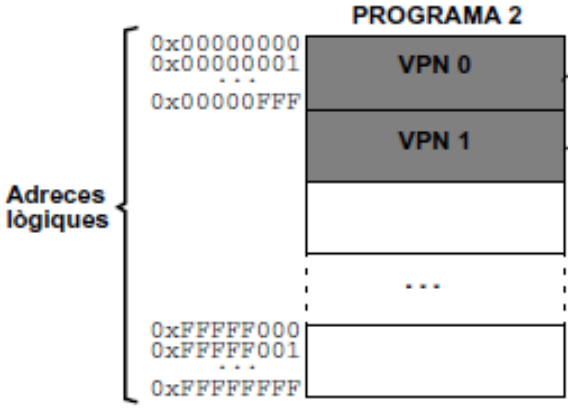


Exemple

- 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)

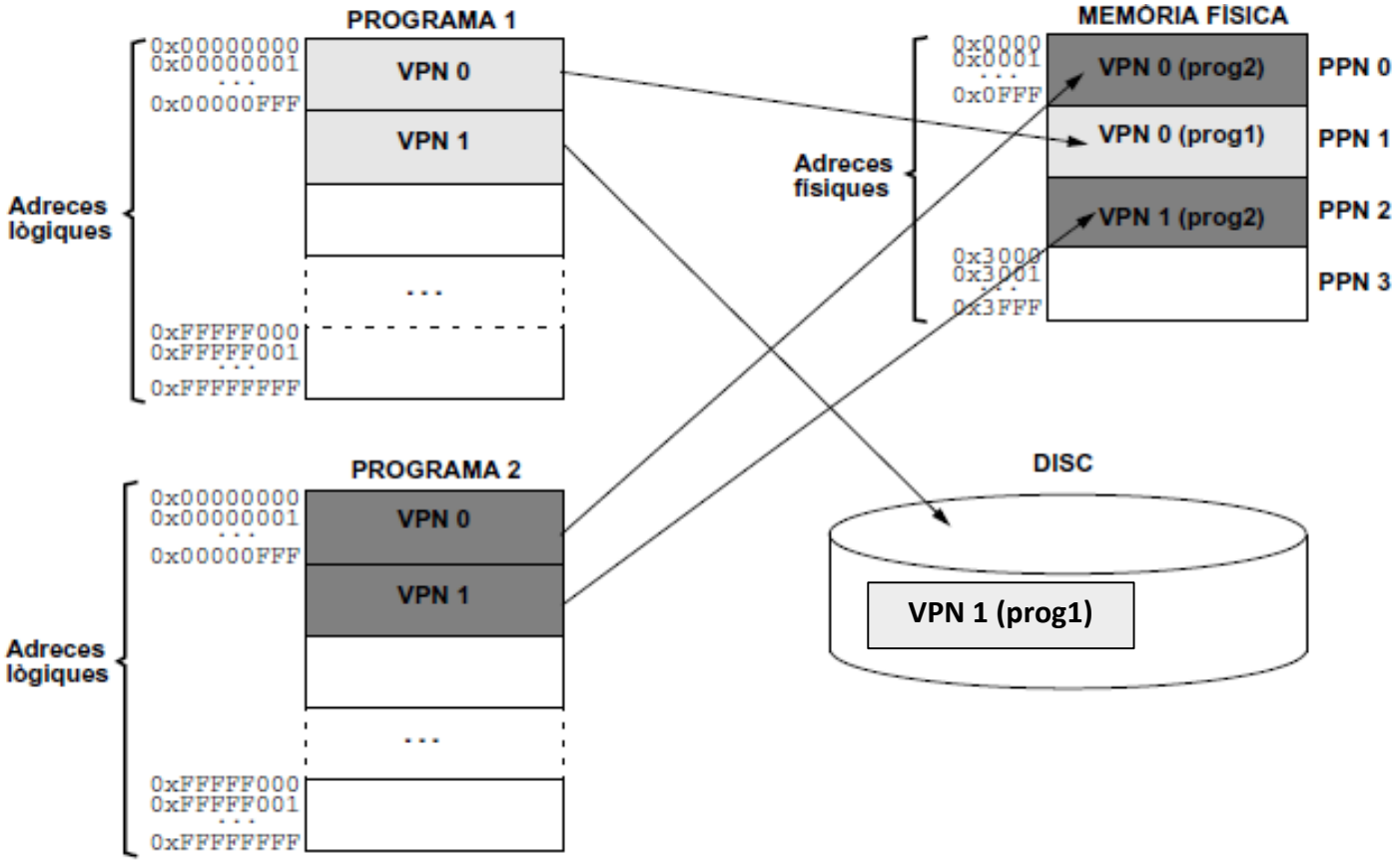






Exemple

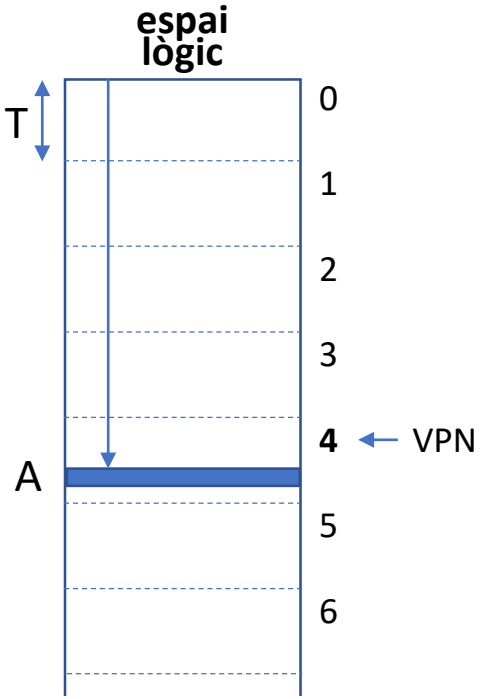
- Exemple:
  - Pàgines de 4KB (2^{12} bytes), memòria física de 16KB (4 marcs)
  - Programa 1 i Programa 2 fan servir sols 2 pàgines lògiques (VPN0 i VPN1)
  - La pàgina VPN1 del programa 1 no està carregada en memòria



Adreça lògica: com determinar el VPN

- A quina pàgina (VPN) pertany l'adreça lògica A?
  - Si la mida de pàgina és T

VPN = A div T



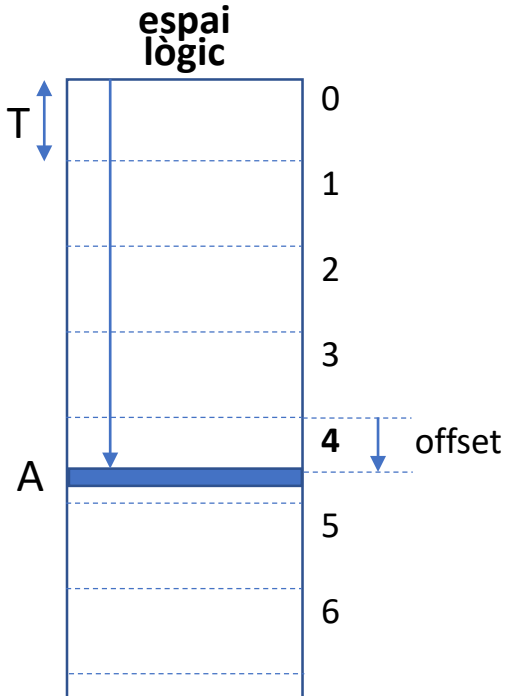
Adreça lògica: com determinar el VPN

- A quina pàgina (VPN) pertany l'adreça lògica 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



Adreça lògica: com determinar el VPN

- A quina pàgina (VPN) pertany l'adreça lògica 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

- Exemple
  - Adreces lògiques de 32 bits
  - T = 4KB = 2^{12} bytes
  - A = 0x10010004

A = | 0001 0000 0000 0001 0000 0000 0000 0100 |

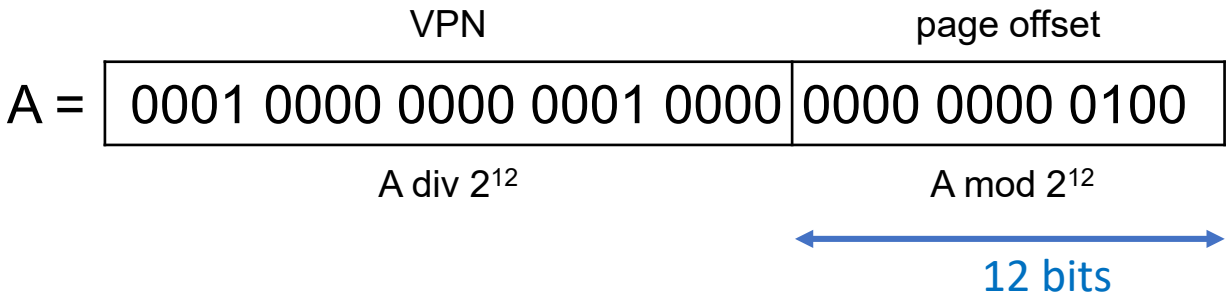
Adreça lògica: com determinar el VPN

- A quina pàgina (VPN) pertany l'adreça lògica 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

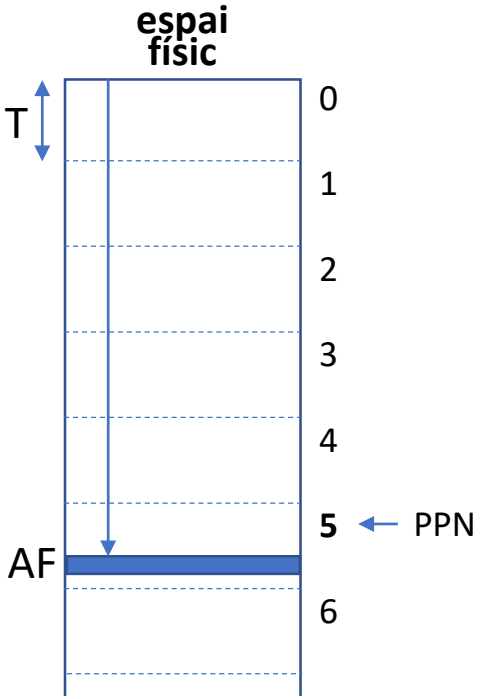
- Exemple
  - Adreces lògiques de 32 bits
  - T = 4KB = 2^{12} bytes
  - A = 0x10010004



Adreça física: com determinar el PPN

- A quin marc de pàgina (PPN) pertany l'adreça física AF?
  - Si la mida de pàgina és T

PPN = AF div T



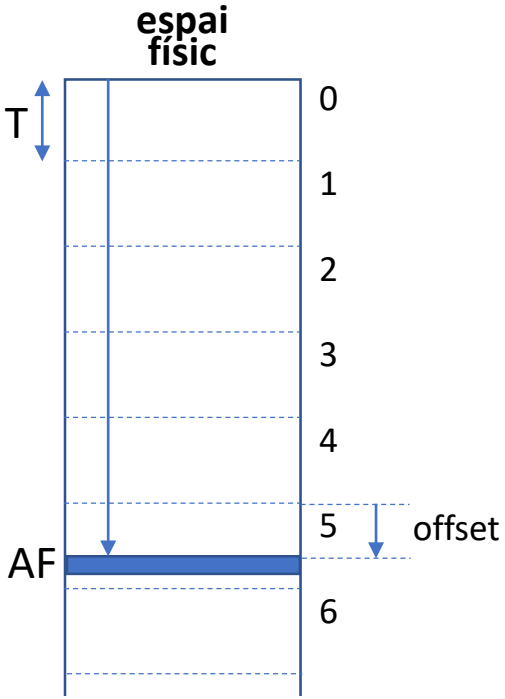
Adreça física: com determinar el PPN

- A quin marc de pàgina (PPN) pertany l'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



Adreça física: com determinar el PPN

- A quin marc de pàgina (PPN) pertany l'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

- Exemple
  - Adreces físiques de 28 bits
  - T = 4KB = 2^{12} bytes
  - AF = 0x0301004

AF = | 0000 0011 0000 0001 0000 0000 0100 |

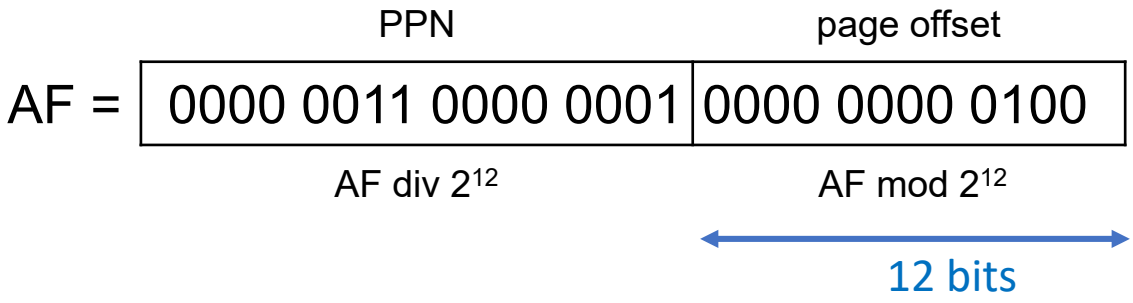
Adreça física: com determinar el PPN

- A quin marc de pàgina (PPN) pertany l'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

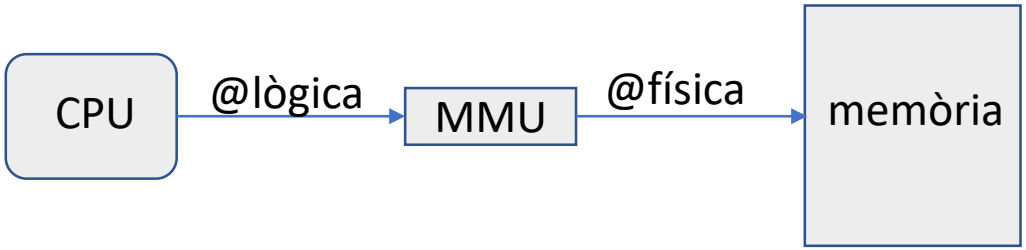
    offset = AF mod T

- Exemple
  - Adreces físiques de 28 bits
  - T = 4KB = 2^{12} bytes
  - AF = 0x0301004



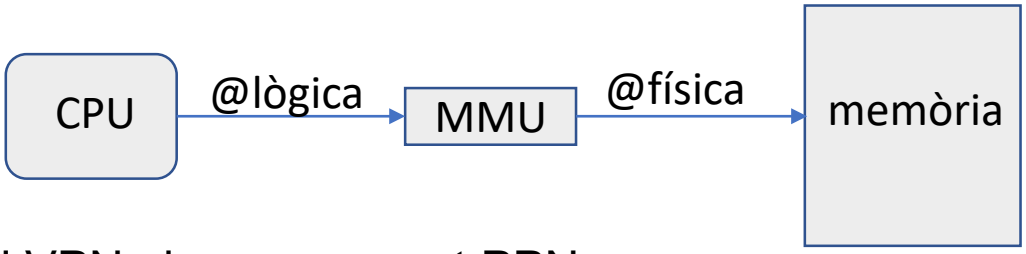
Traducció d'adreces

- 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)

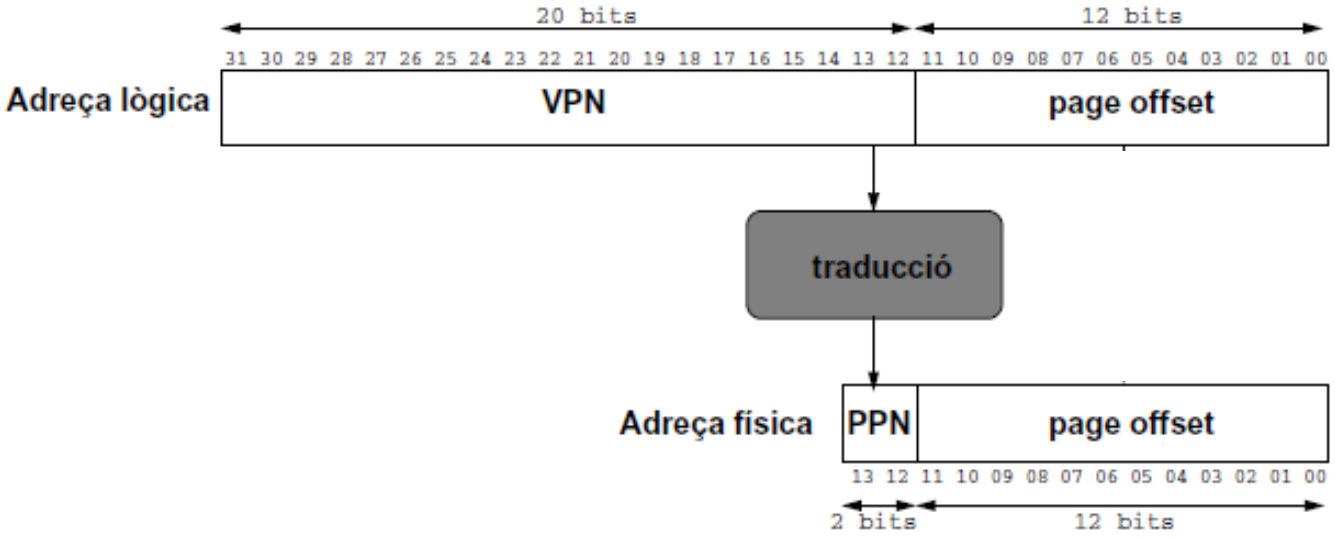


Traducció d'adreces

- 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)

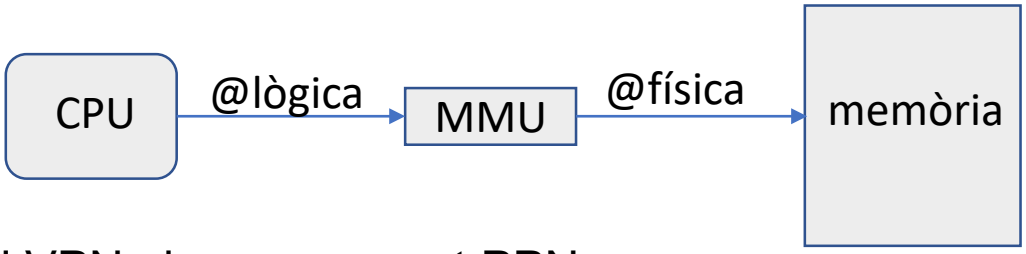


  - Tradueix el VPN al corresponent PPN

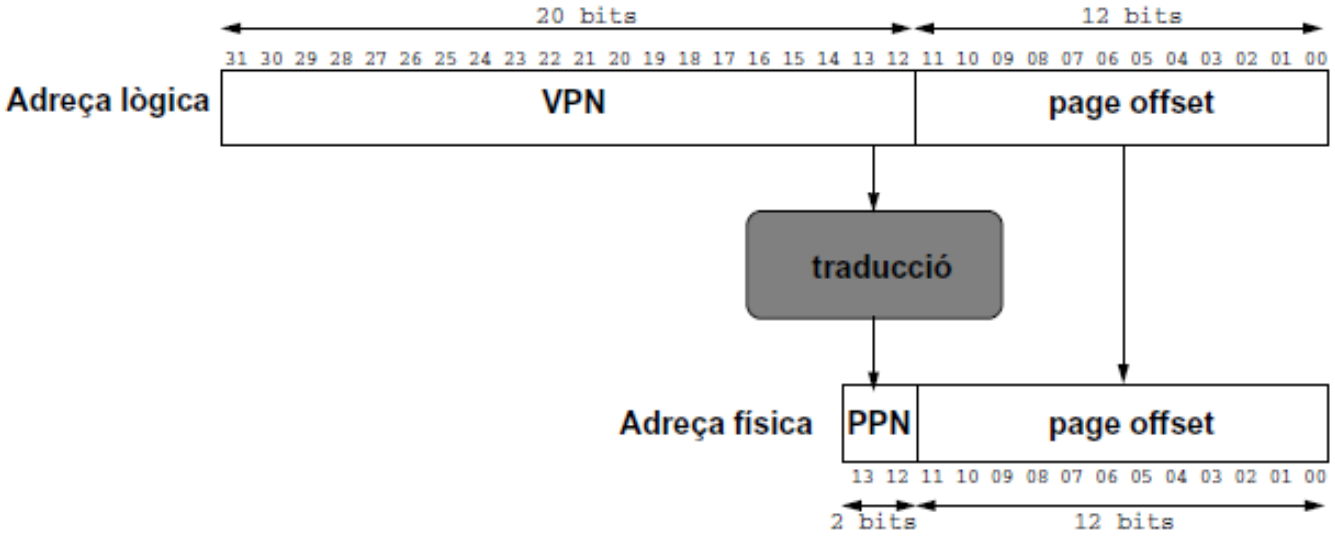


Traducció d'adreces

- 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)

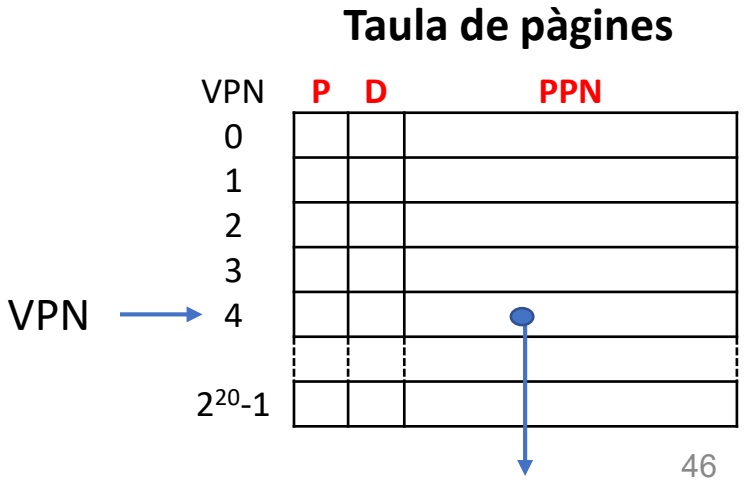


  - Tradueix el VPN al corresponent PPN
  - L'offset no canvia



Traducció amb Taula de Pàgines

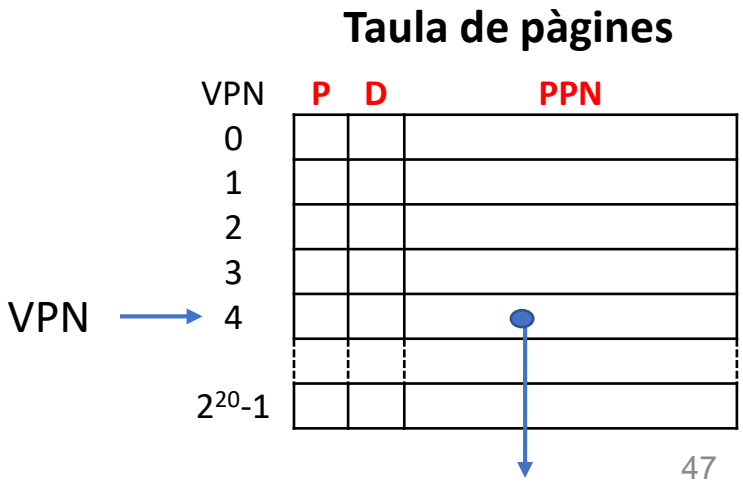
- Com sap la MMU en quin marc (PPN) està cada pàgina (VPN)?
  - Consultant la **Taula de Pàgines** que manté el S.O.
  - Té tantes entrades com pàgines té l'espai lògic (s'indexa amb el VPN)



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
  - **P:** Bit de presència. Val 1 si la pàgina **és vàlida** i **està present** en MP
  - **D:** Bit de modificada. Val 1 si **ha estat modificada** (per un Store)
  - **PPN:** Marc de pàgina
  - Altres bits
    - Permisos (p.ex. Lectura/ Escriptura/ Execució)
    - Bit de Referència (per gestionar l'algorisme LRU)

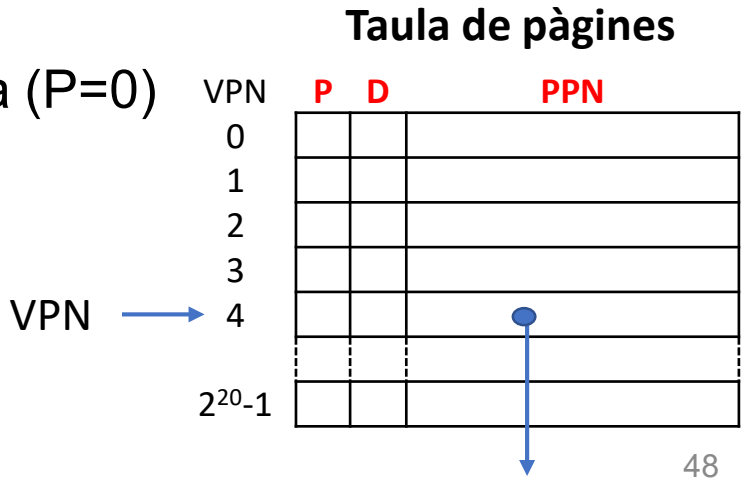


Traducció amb Taula de Pàgines

- 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
  - **P:** Bit de presència. Val 1 si la pàgina **és vàlida** i **està present** en MP
  - **D:** Bit de modificada. Val 1 si **ha estat modificada** (per un Store)
  - **PPN:** Marc de pàgina
  - Altres bits
    - Permisos (p.ex. Lectura/ Escriptura/ Execució)
    - Bit de Referència (per gestionar l'algorisme LRU)

- Si la pàgina no està present en memòria (P=0)
  - La PTE conté la ubicació en disc



Encert i Fallada de pàgina

- Memòria i Disc són 2 nivells de la jerarquia de memòria
  - Al principi, totes les pàgines resideixen al disc
  - A mesura que s'accedeixen, es van carregant en MP

Encert i Fallada de pàgina

- Memòria i Disc són 2 nivells de la jerarquia de memòria
  - Al 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

Encert i Fallada de pàgina

- Memòria i Disc són 2 nivells de la jerarquia de memòria
  - Al 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!)
  - El S.O. Actualitza la PTE corresponent (bits P, D i PPN)
  - Es reexecuta la instrucció que ha causat la fallada

Emplaçament i reemplaçament

- Algorisme d'emplaçament: on es pot allotjar una pàgina?
  - Mapeig totalment associatiu: en qualsevol marc
  - Evita *fallades per conflicte*

Emplaçament i reemplaçament

- Algorisme d'emplaçament: on es pot allotjar una pàgina?
  - Mapeig totalment associatiu: en qualsevol marc
  - Evita *fallades per conflicte*

- Algorisme de reemplaçament: en cas de fallada, com seleccionem un marc?
  - Si en queden, en un marc lliure
  - Si no en queden, reemplaçar-ne un d'ocupat
  - Reemplaçament LRU

Emplaçament i reemplaçament

- Algorisme d'emplaçament: on es pot allotjar una pàgina?
  - Mapeig totalment associatiu: en qualsevol marc
  - Evita *fallades per conflicte*

- Algorisme de reemplaçament: en cas de fallada, com seleccionem un marc?
  - Si en queden, en un marc lliure
  - Si no en queden, reemplaçar-ne un d'ocupat
  - Reemplaçament LRU
  - Si la pàgina reemplaçada està modificada (bit D=1), s'escriu al disc

Emplaçament i reemplaçament

- Algorisme d'emplaçament: on es pot allotjar una pàgina?
  - Mapeig totalment associatiu: en qualsevol marc
  - Evita *fallades per conflicte*

- Algorisme de reemplaçament: en cas de fallada, com seleccionem un marc?
  - Si en queden, en un marc lliure
  - Si no en queden, reemplaçar-ne un d'ocupat
  - Reemplaçament LRU
  - Si la pàgina reemplaçada està modificada (bit D=1), s'escriu al disc
  - La zona de disc on s'emmagatzemen les pàgines reemplaçades s'anomena *fitxer d'intercanvi*, *espai d'intercanvi* o *swap*

Política d'escriptura (stores)

- Escriptura immediata?
  - Que cada store escrigui a MP i també al Disc?

Política d'escriptura (stores)

- Escriptura immediata?
  - Que cada store escrigui a MP i també al Disc?
  - Però cada accés al disc tarda milions de cicles
  - És totalment impràctica!!

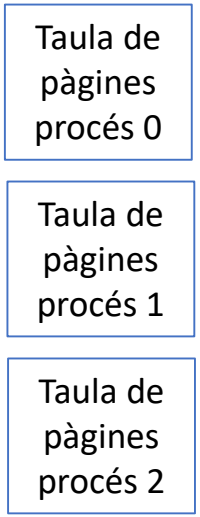
Política d'escriptura (stores)

- Escriptura immediata?
  - Que cada store escrigui a MP i també al Disc?
  - Però cada accés al disc tarda milions de cicles
  - És totalment impràctica!!

- Escriptura retardada amb assignació
  - Els stores només escriuen a MP (no al disc)
  - El store marca la pàgina "modificada" posant el bit Dirty (D=1)
  - Si més tard és reemplaçada, caldrà escriure-la al disc

El registre de Taula de Pàgines

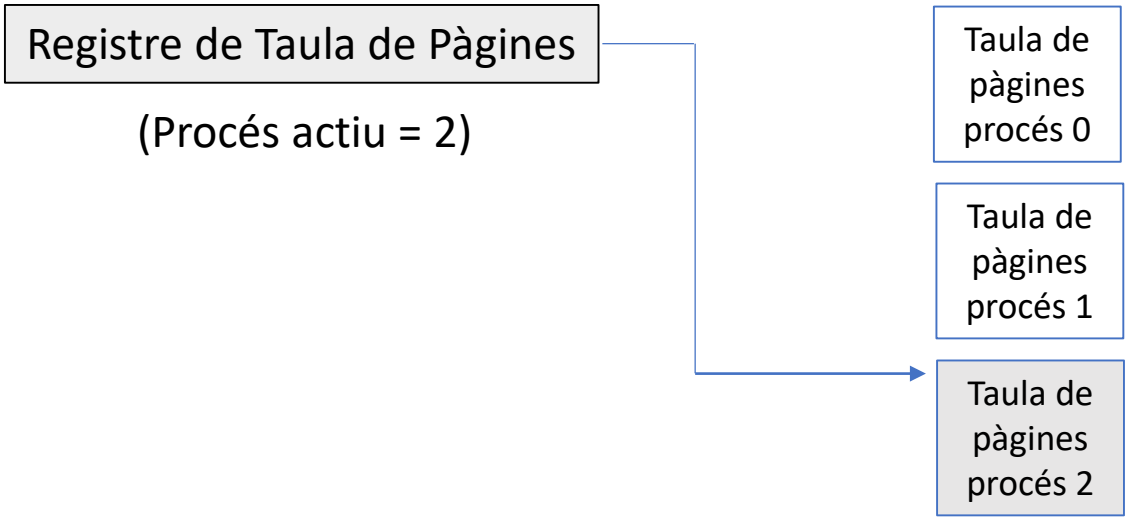
- Múltiples processos poden executar-se *concurrentment*
  - Cada un té el seu propi espai lògic → la seva Taula de Pàgines
  - Alternen l'ús de CPU al llarg del temps, fent *canvis de context*



El registre de Taula de Pàgines

- Múltiples processos poden executar-se *concurrentment*
  - Cada un té el seu propi espai lògic → la seva Taula de Pàgines
  - Alternen l'ús de CPU al llarg del temps, fent *canvis de context*

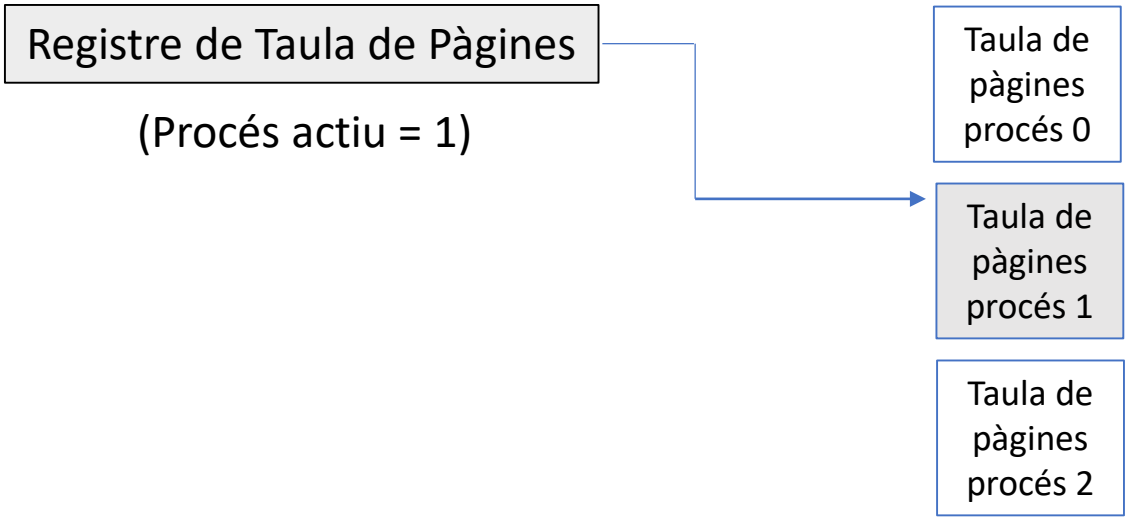
- Però sols el *procés actiu* s'executa en un instant donat
  - La CPU manté un **Registre de Taula de Pàgines**
  - Apunta a la Taula de Pàgines del *procés actiu*



El registre de Taula de Pàgines

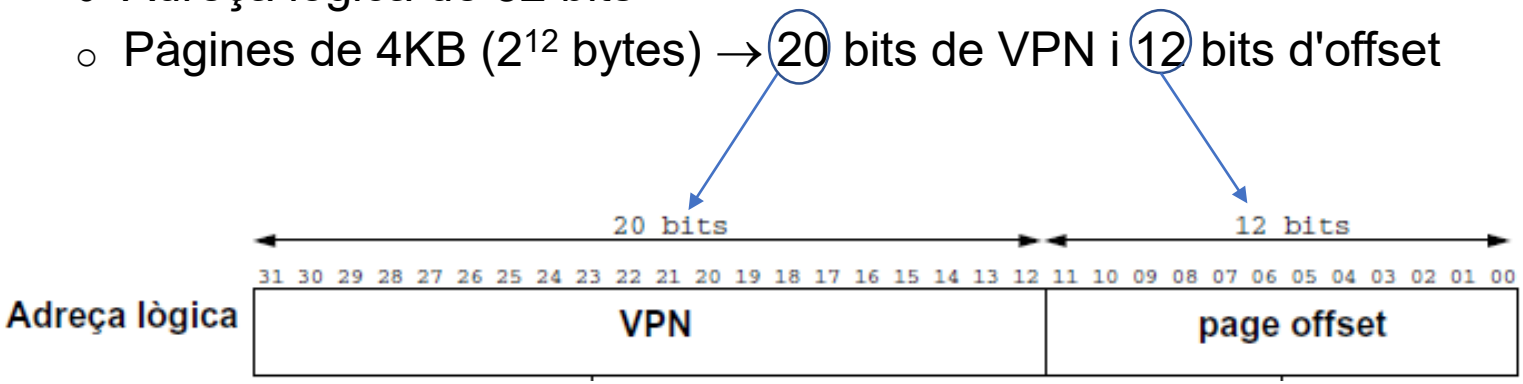
- Múltiples processos poden executar-se *concurrentment*
  - Cada un té el seu propi espai lògic → la seva Taula de Pàgines
  - Alternen l'ús de CPU al llarg del temps, fent *canvis de context*

- 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*



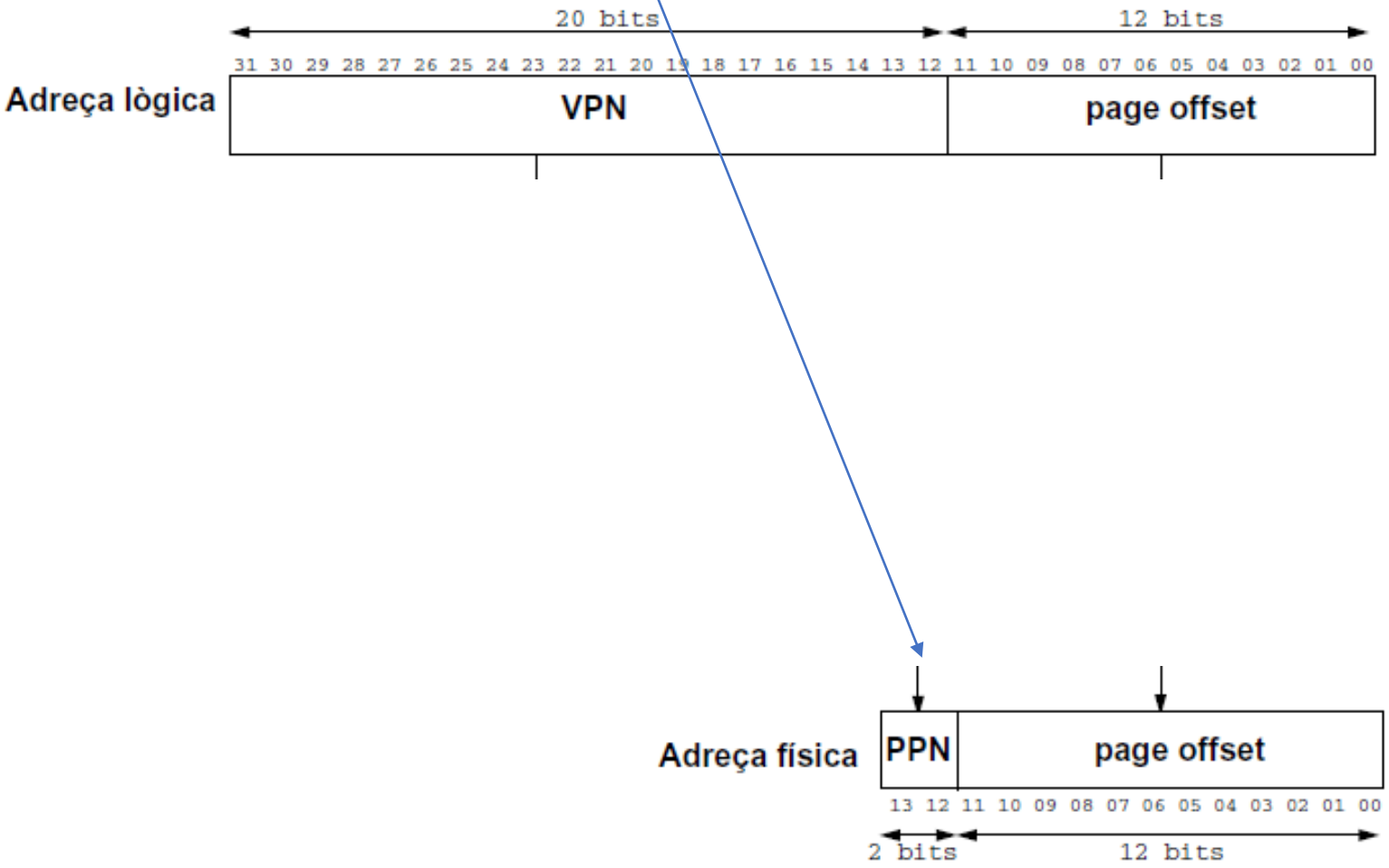
Traducció amb Taula de Pàgines (diagrama de flux)

- Adreça lògica de 32 bits
- Pàgines de 4KB (2^{12} bytes) → 20 bits de VPN i 12 bits d'offset



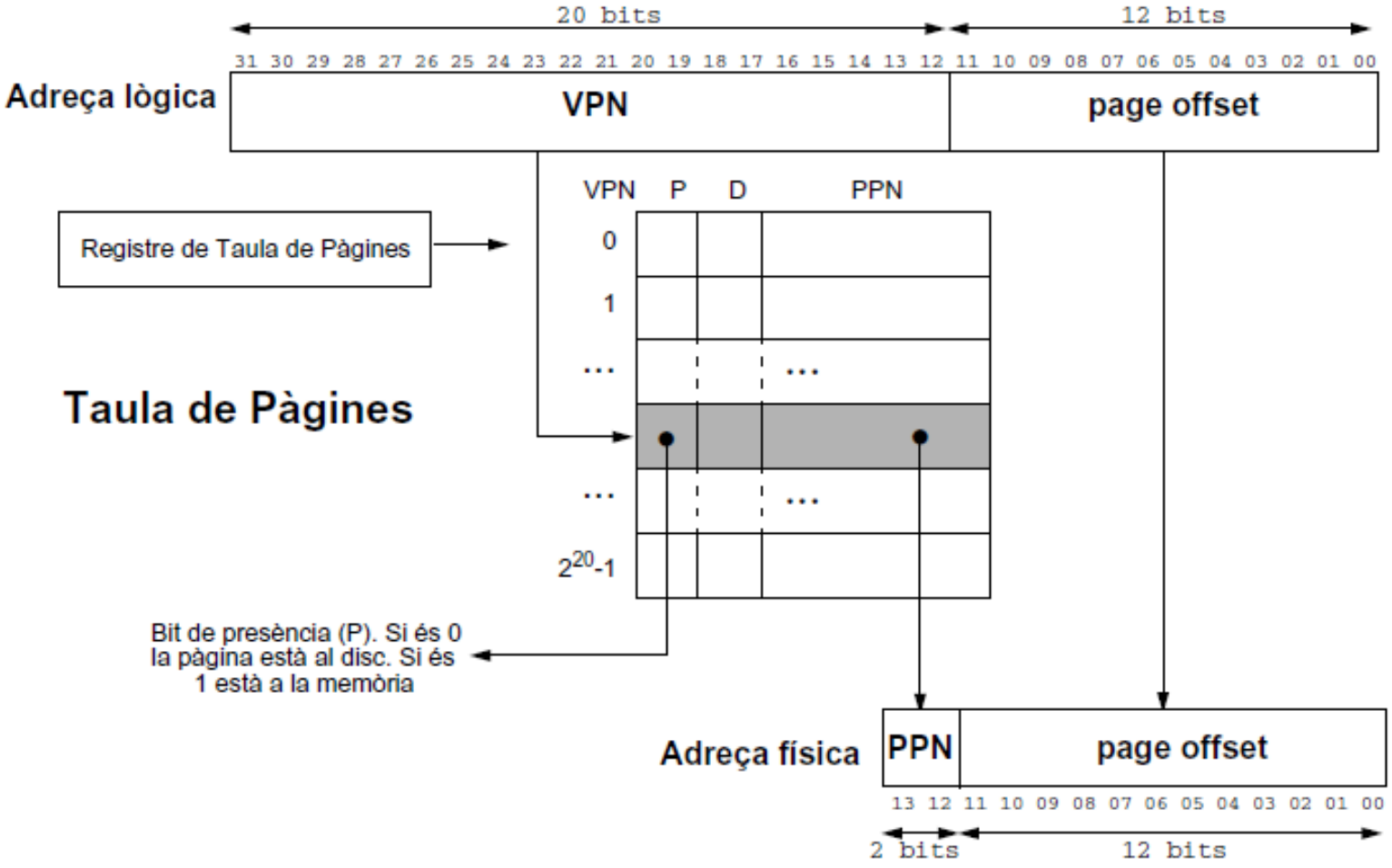
Traducció amb Taula de Pàgines (diagrama de flux)

- Adreça lògica de 32 bits
- Pàgines de 4KB (2^{12} bytes) → 20 bits de VPN i 12 bits d'offset
- MP de 16KB (4 marcs) → 2 bits de PPN



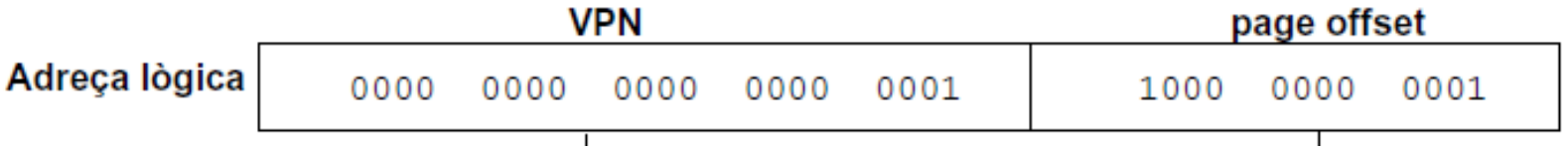
Traducció amb Taula de Pàgines (diagrama de flux)

- Adreça lògica de 32 bits
- Pàgines de 4KB (2^{12} bytes) → 20 bits de VPN i 12 bits d'offset
- MP de 16KB (4 marcs) → 2 bits de PPN



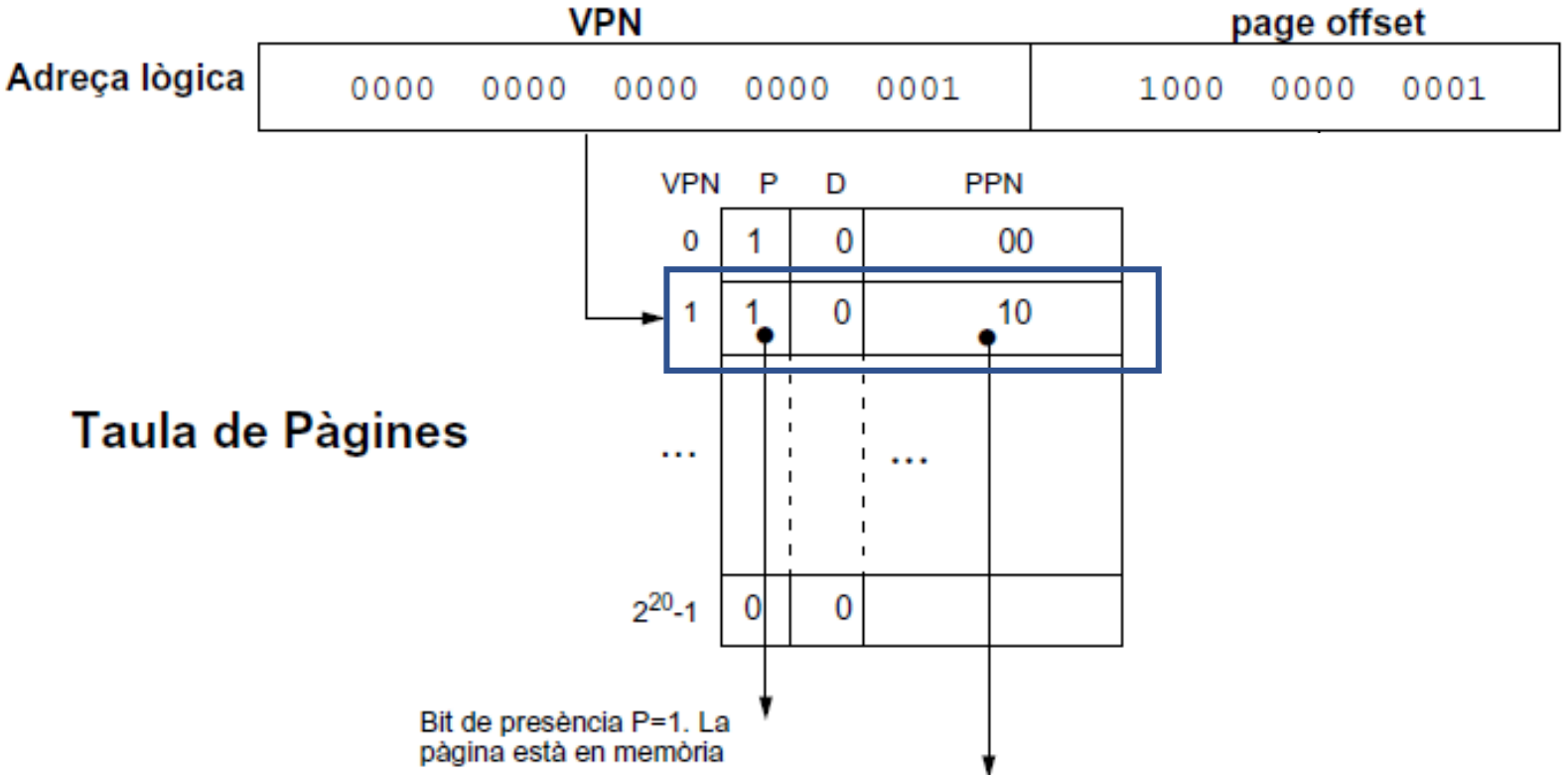
Exemple

- La MMU tradueix l'adreça lògica A = 0x00001801



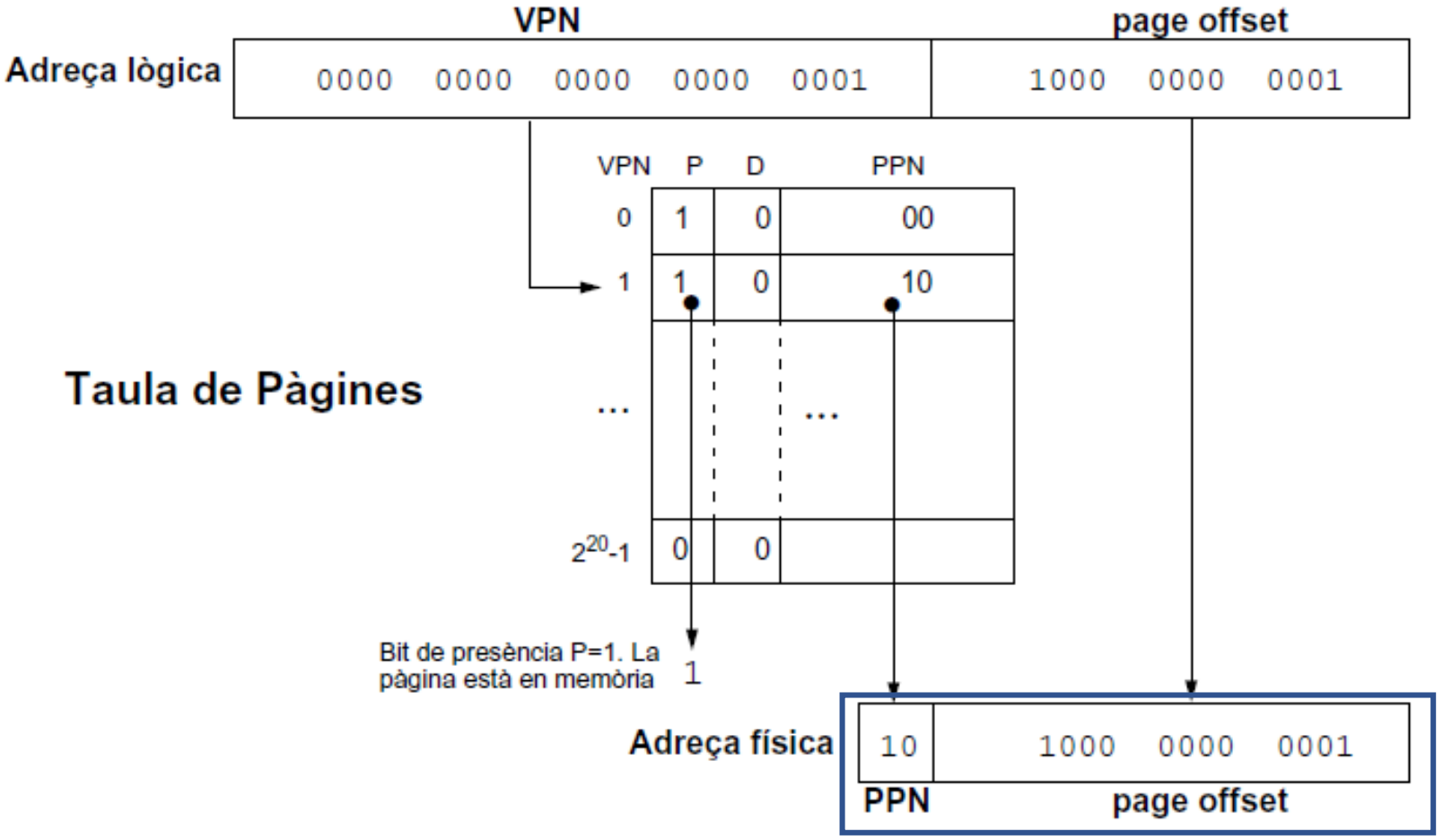
Exemple

- La MMU tradueix l'adreça lògica A = 0x00001801
- La pàgina està Present en memòria (P=1) i resideix al marc PPN = 2



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



Memòria Virtual

- Introducció
- Memòria virtual paginada
- Traducció ràpida amb TLB

Memòria Virtual

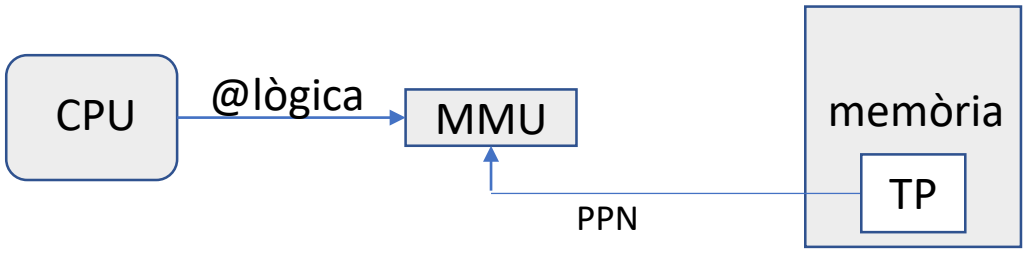
- Introducció
- Memòria virtual paginada
- Traducció ràpida amb TLB
  - Aspectes de disseny
  - Gestió de les fallades de pàgina i de TLB
  - Protecció i Compartició
  - Integració de TLB i Cache

Traducció ràpida amb TLB

- Problema: les TP resideixen en MP

Traducció ràpida amb TLB

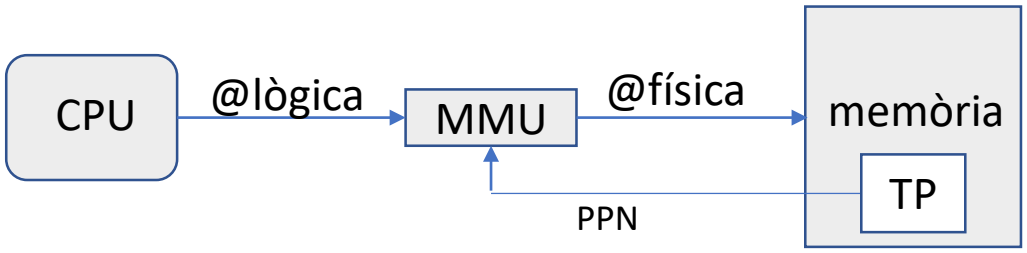
- Problema: les TP resideixen en MP
- Cada accés a memòria de la CPU en requereix 2
  - 1 accés a la TP per traduir l'adreça



Traducció ràpida amb TLB

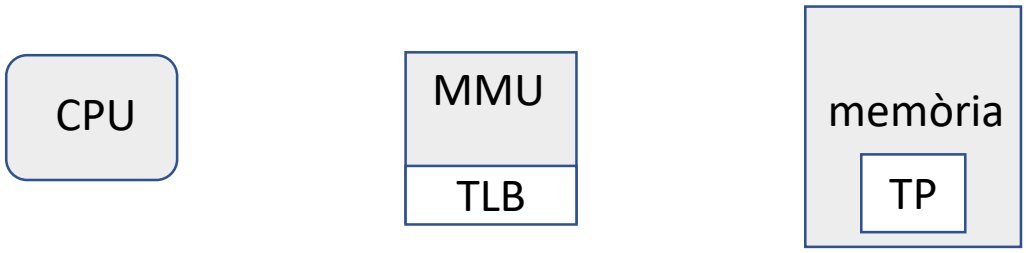
- Problema: les TP resideixen en MP
- Cada accés a memòria de la CPU en requereix 2
  - 1 accés a la TP per traduir l'adreça
  - 1 accés a MP per llegir/escriure la dada

  $\Rightarrow$ L'accés a la TP augmenta molt el temps d'accés a memòria !



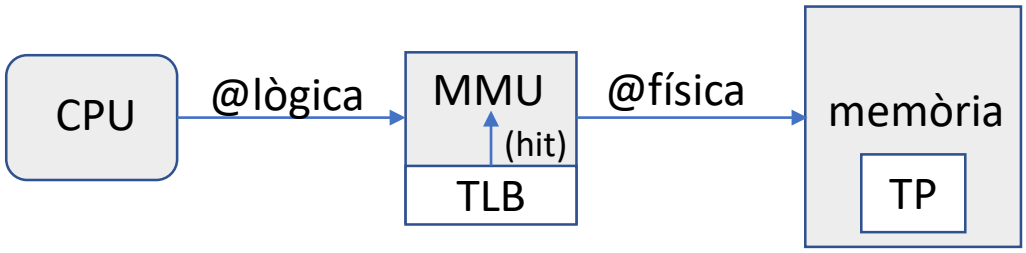
Traducció ràpida amb TLB

- Solució: el TLB (Translation-Lookaside Buffer)
  - És una *cache de traduccions*
  - Guarda les entrades (PTE) de la TP accedides més recentment



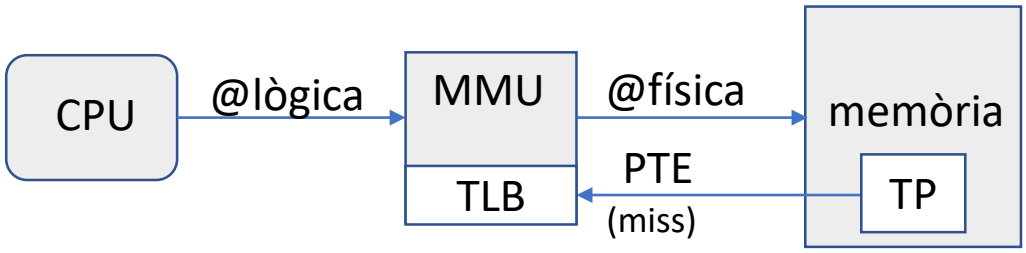
Traducció ràpida amb TLB

- Solució: el TLB (Translation-Lookaside Buffer)
  - És una *cache de traduccions*
  - Guarda les entrades (PTE) de la TP accedides més recentment
    - En cas d'encert, no cal accedir a la TP



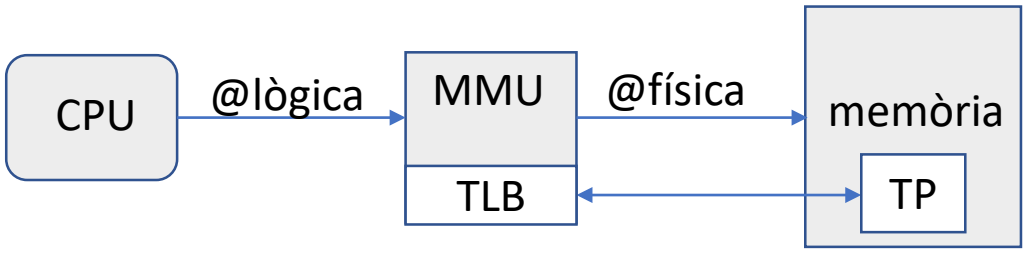
Traducció ràpida amb TLB

- Solució: el TLB (Translation-Lookaside Buffer)
  - És una *cache de traduccions*
  - Guarda les entrades (PTE) de la TP accedides més recentment
    - En cas d'encert, no cal accedir a la TP
    - En cas de fallada, ha de copiar la PTE, de la TP al TLB ☹



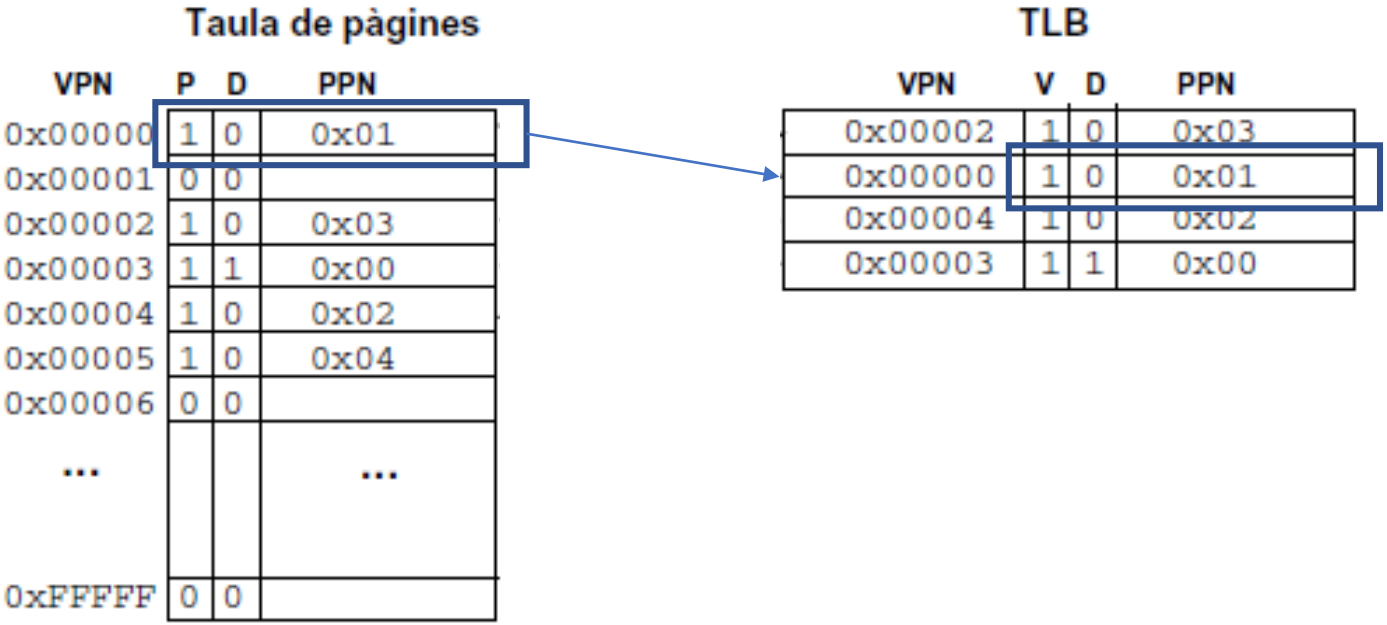
Traducció ràpida amb TLB

- Solució: el TLB (Translation-Lookaside Buffer)
  - És una *cache de traduccions*
  - Guarda les entrades (PTE) de la TP accedides més recentment
    - En cas d'encert, no cal accedir a la TP
    - En cas de fallada, ha de copiar la PTE, de la TP al TLB
  - Paràmetres típics
    - Capacitat: 16 - 512 PTEs
    - Temps de hit: 0,5 - 1 cicles
    - Taxa de fallades: 0,01% - 1% $\rightarrow$ L'accés a TP té molta localitat!



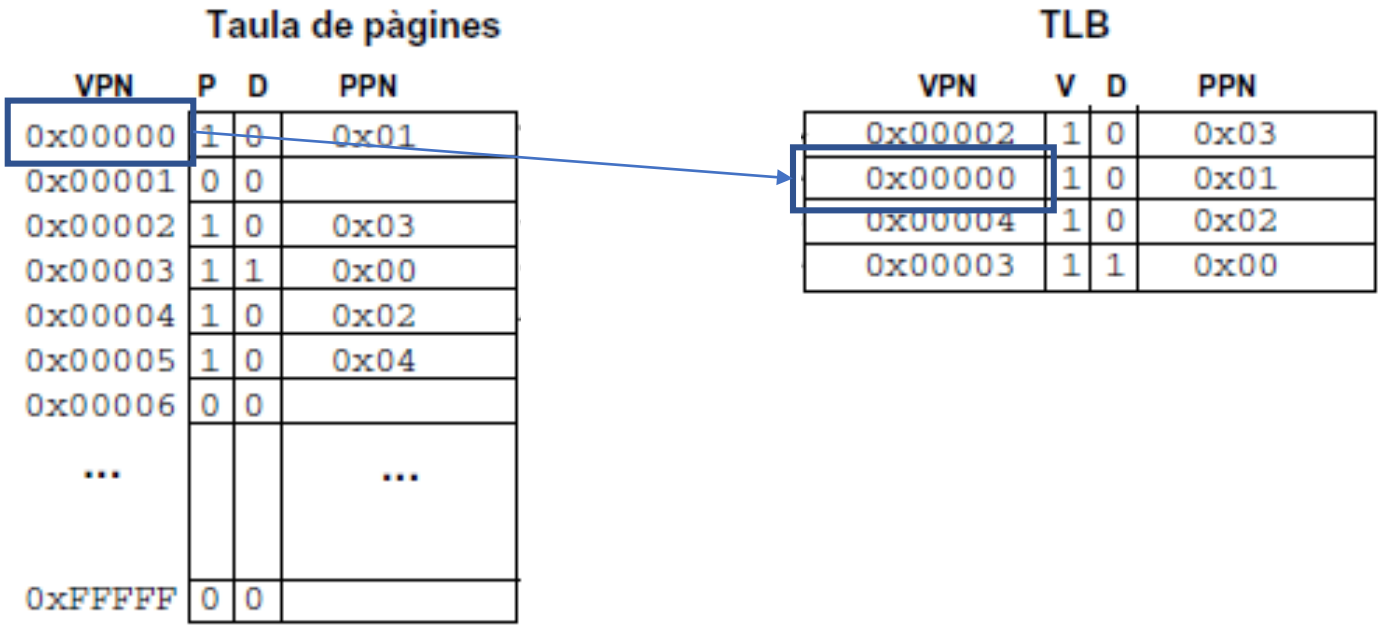
Organització del TLB

- Cada entrada conté (exemple de MIPS)
  - Còpia de la PTE: PPN, bit V (còpia del bit P), bit D



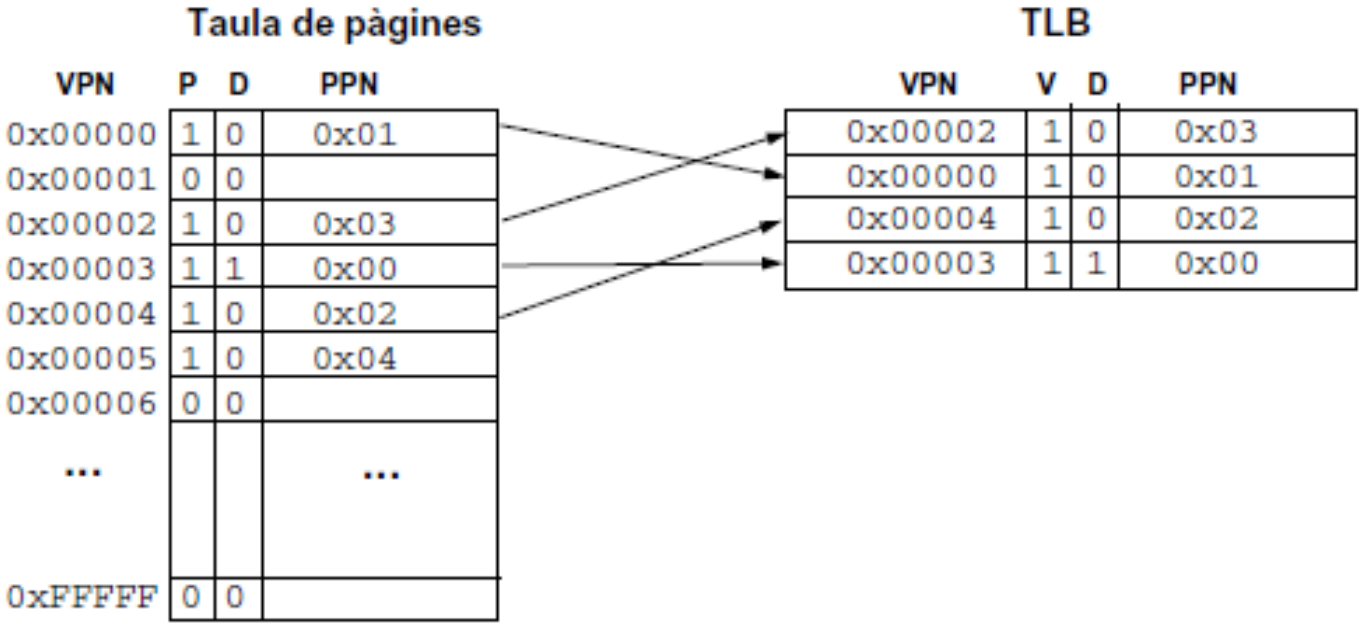
Organització del TLB

- 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)



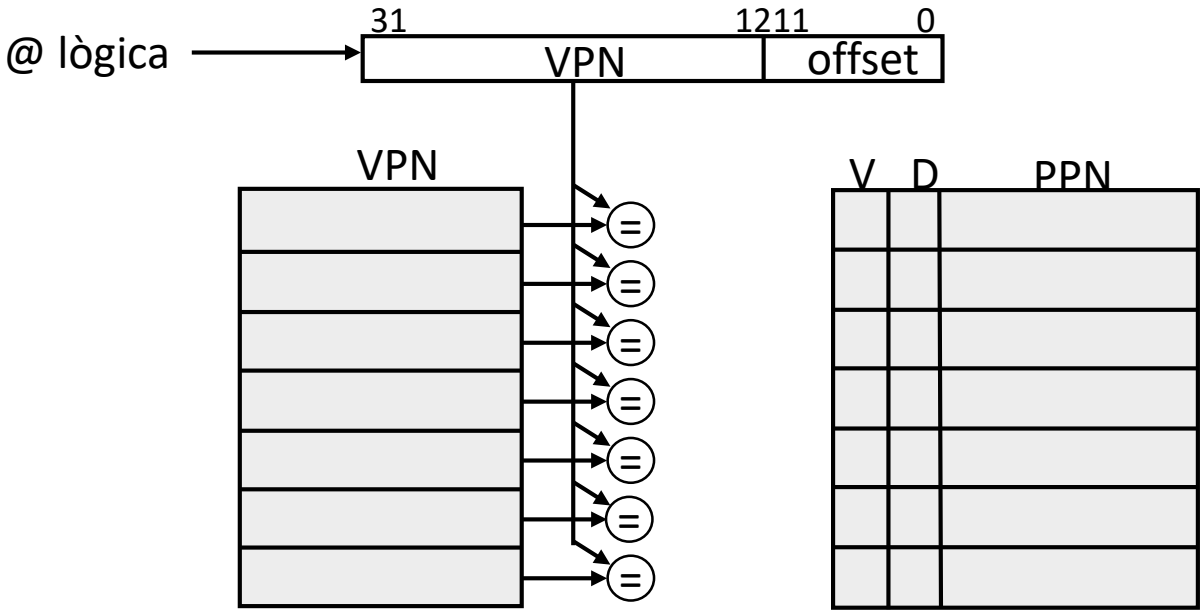
Organització del TLB

- 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)
- Mapeig completament associatiu
  - Una PTE es pot guardar en qualsevol entrada del TLB



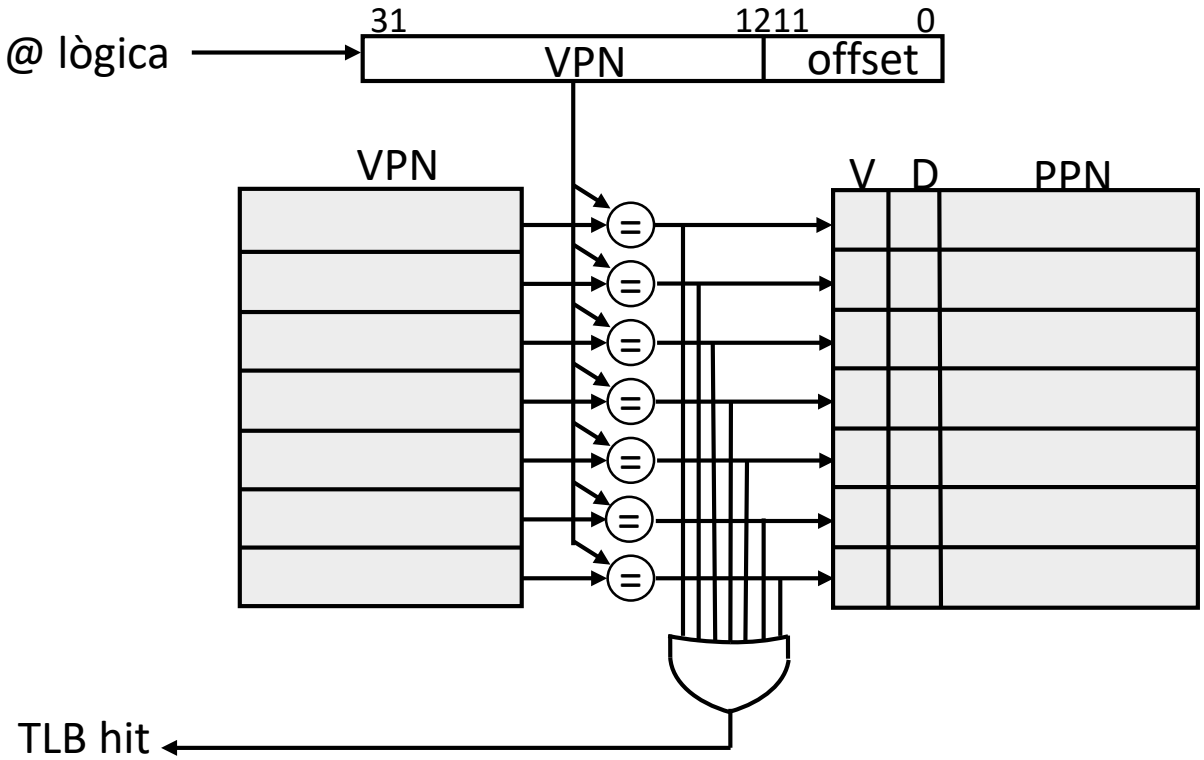
Traducció ràpida amb TLB: encert

- Quan la MMU consulta el TLB, compara TOTS els VPN (etiquetes)



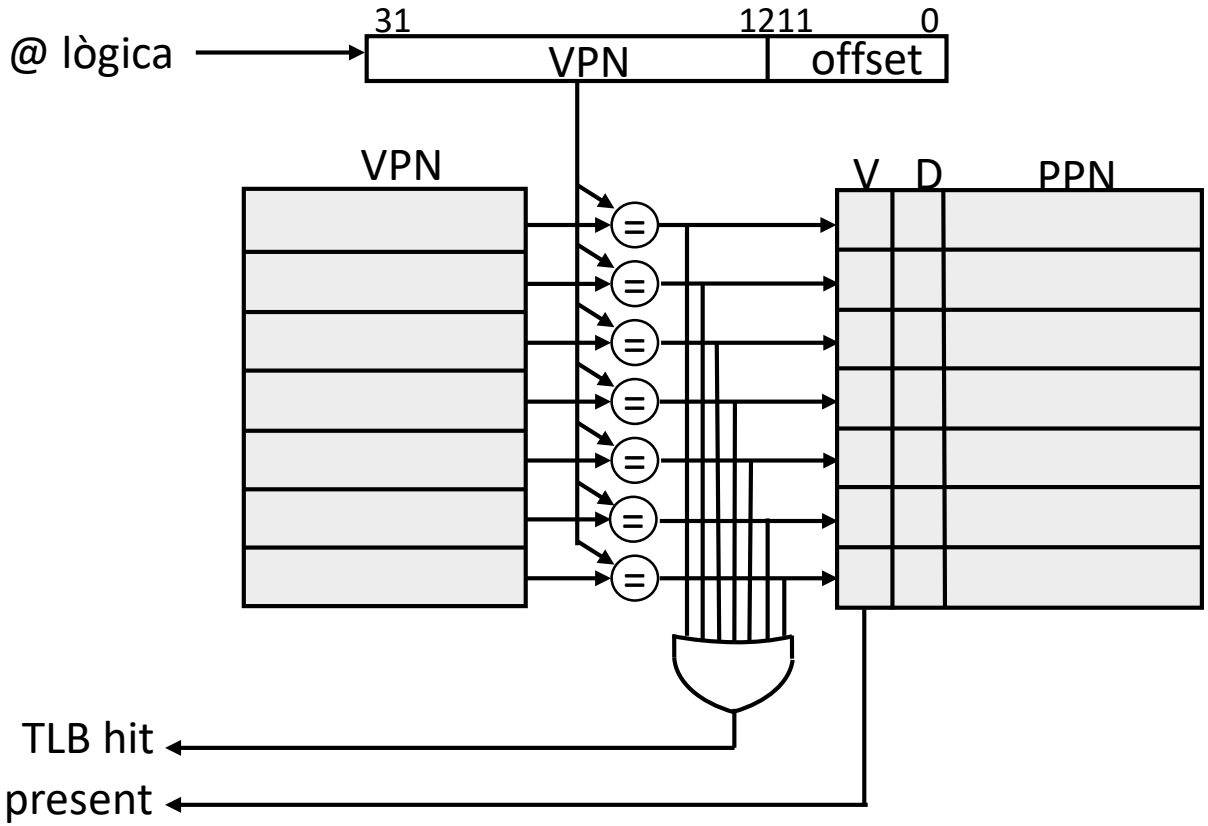
Traducció ràpida amb TLB: encert

- Quan la MMU consulta el TLB, compara TOTS els VPN (etiquetes)
- Si algun VPN coincideix, tenim un encert de TLB



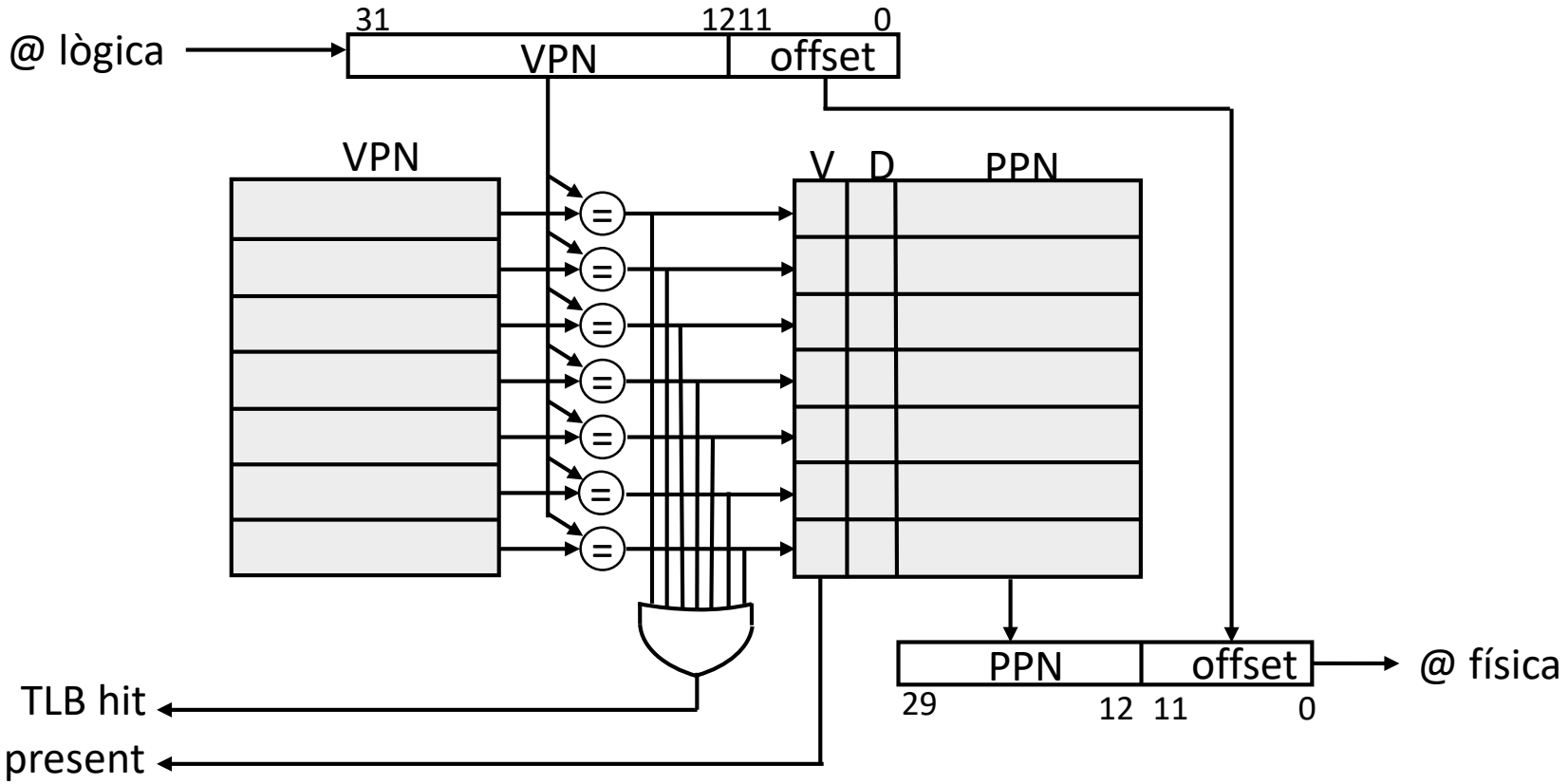
Traducció ràpida amb TLB: encert

- Quan la MMU consulta el TLB, compara TOTS els VPN (etiquetes)
- Si algun VPN coincideix, tenim un encert de TLB
  - És un encert de TLB inclús si el bit V=0 (pàgina no-present)



Traducció ràpida amb TLB: encert

- Quan la MMU consulta el TLB, compara TOTS els VPN (etiquetes)
- Si algun VPN coincideix, tenim un encert de TLB
  - És un encert de TLB inclús si el bit V=0 (pàgina no-present)
- El camp PPN és la traducció buscada



Fallada de TLB

- Si cap VPN coincideix, tenim una fallada de TLB
  - Cal accedir a la TP i copiar la PTE al TLB

Fallada de TLB

- Si cap VPN coincideix, tenim una fallada de TLB
  - Cal accedir a la TP i copiar la PTE al TLB

- Algorisme d'emplaçament: on es pot allotjar una PTE?
  - Mapeig totalment associatiu: en qualsevol entrada del TLB

Fallada de TLB

- Si cap VPN coincideix, tenim una fallada de TLB
  - Cal accedir a la TP i copiar la PTE al TLB
- Algorisme d'emplaçament: on es pot allotjar una PTE?
  - Mapeig totalment associatiu: en qualsevol entrada del TLB
- Algorisme de reemplaçament: quina entrada seleccionar?
  - Si en queden, en una entrada lliure del TLB (amb el bit V=0)

Fallada de TLB

- Si cap VPN coincideix, tenim una fallada de TLB
  - Cal accedir a la TP i copiar la PTE al TLB
- Algorisme d'emplaçament: on es pot allotjar una PTE?
  - Mapeig totalment associatiu: en qualsevol entrada del TLB
- Algorisme de reemplaçament: quina entrada seleccionar?
  - Si en queden, en una entrada lliure del TLB (amb el bit V=0)
  - Si no en queden, reemplaçar-ne una d'ocupada
  - Reemplaçament LRU (o Random)

Fallada de TLB

- Si cap VPN coincideix, tenim una fallada de TLB
  - Cal accedir a la TP i copiar la PTE al TLB
- Algorisme d'emplaçament: on es pot allotjar una PTE?
  - Mapeig totalment associatiu: en qualsevol entrada del TLB
- Algorisme de reemplaçament: quina entrada seleccionar?
  - Si en queden, en una entrada lliure del TLB (amb el bit V=0)
  - Si no en queden, reemplaçar-ne una d'ocupada
  - Reemplaçament LRU (o Random)
- Alternatives per a gestionar la fallada
  - MIPS: gestió per software
    - La MMU genera una excepció i invoca el S.O.

Fallada de TLB

- Si cap VPN coincideix, tenim una fallada de TLB
  - Cal accedir a la TP i copiar la PTE al TLB
- Algorisme d'emplaçament: on es pot allotjar una PTE?
  - Mapeig totalment associatiu: en qualsevol entrada del TLB
- Algorisme de reemplaçament: quina entrada seleccionar?
  - Si en queden, en una entrada lliure del TLB (amb el bit V=0)
  - Si no en queden, reemplaçar-ne una d'ocupada
  - Reemplaçament LRU (o Random)
- Alternatives per a gestionar la fallada
  - 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 sobre els bits Valid i Dirty (MIPS)

1. El bit V del TLB pot valdre 0 per 2 raons
   - Entrada lliure, sense inicialitzar

Nota sobre els bits Valid i Dirty (MIPS)

1. El bit V del TLB pot valdre 0 per 2 raons
   - Entrada lliure, sense inicialitzar
   - Entrada que conté una PTE acabada de copiar de la TP després d'una fallada de TLB, i on el bit de presència era 0
     → Encara queda per resoldre la fallada de pàgina

Nota sobre els bits Valid i Dirty (MIPS)

1. El bit V del TLB pot valdre 0 per 2 raons
   - Entrada lliure, sense inicialitzar
   - Entrada que conté una PTE acabada de copiar de la TP després d'una fallada de TLB, i on el bit de presència era 0
     → Encara queda per resoldre la fallada de pàgina

2. El bit D del TLB
   - Es posa a 1 quan s'executa un Store
   - És l'únic camp del TLB que pot ser modificat pel programa

Nota sobre els bits Valid i Dirty (MIPS)

1. El bit V del TLB pot valdre 0 per 2 raons
   - Entrada lliure, sense inicialitzar
   - Entrada que conté una PTE acabada de copiar de la TP després d'una fallada de TLB, i on el bit de presència era 0
     → Encara queda per resoldre la fallada de pàgina

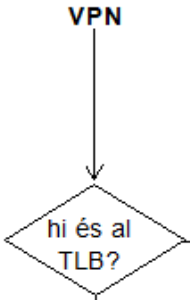
2. El bit D del TLB
   - Es posa a 1 quan s'executa un Store
   - És l'únic camp del TLB que pot ser modificat pel programa
   - Escriptura immediata
     - S'escriu D=1 al TLB i també a la TP (sempre sincronitzats)

Nota sobre els bits Valid i Dirty (MIPS)

1. El bit V del TLB pot valdre 0 per 2 raons
   - Entrada lliure, sense inicialitzar
   - Entrada que conté una PTE acabada de copiar de la TP després d'una fallada de TLB, i on el bit de presència era 0
     → Encara queda per resoldre la fallada de pàgina

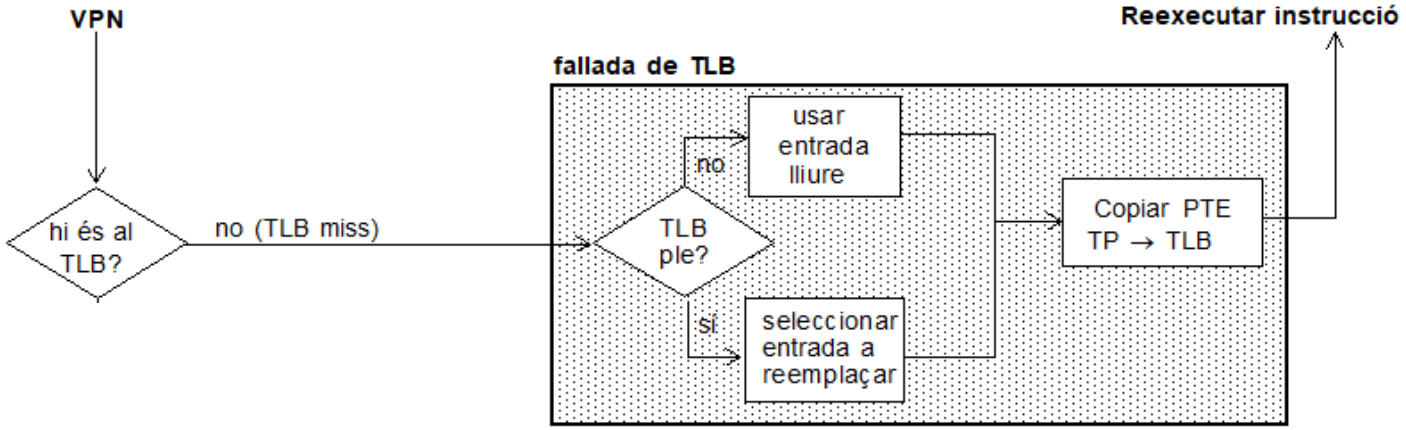
2. El bit D del TLB
   - Es posa a 1 quan s'executa un Store
   - És l'únic camp del TLB que pot ser modificat pel programa
   - Escriptura immediata
     - S'escriu D=1 al TLB i també a la TP (sempre sincronitzats)
   - Tots els stores han d'accedir a la TP en MP?
     - No! solament el primer cop que escrivim a la pàgina, quan D encara val 0 al TLB

Fallades de TLB i de pàgina: diagrama

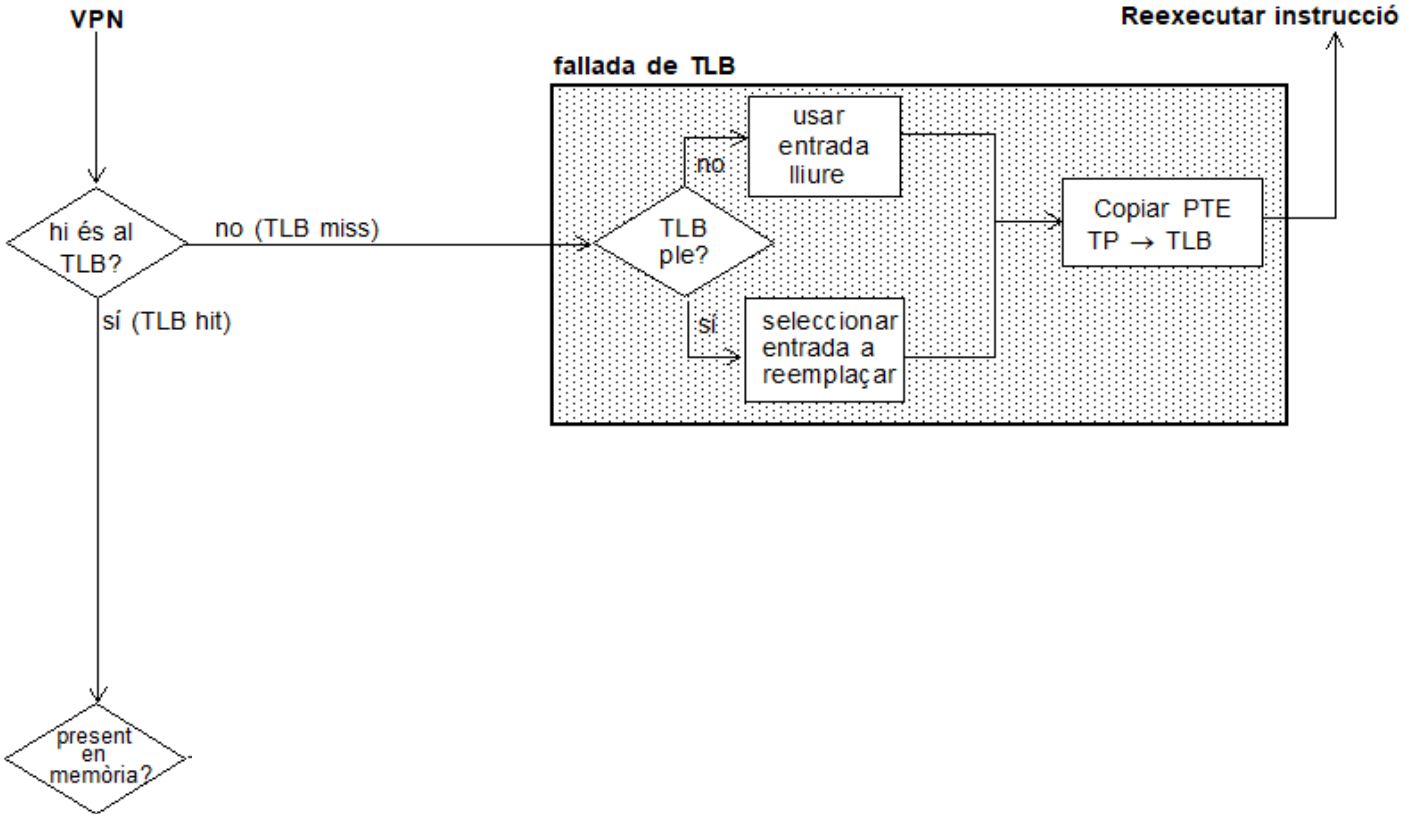


La CPU inicia un accés a memòria

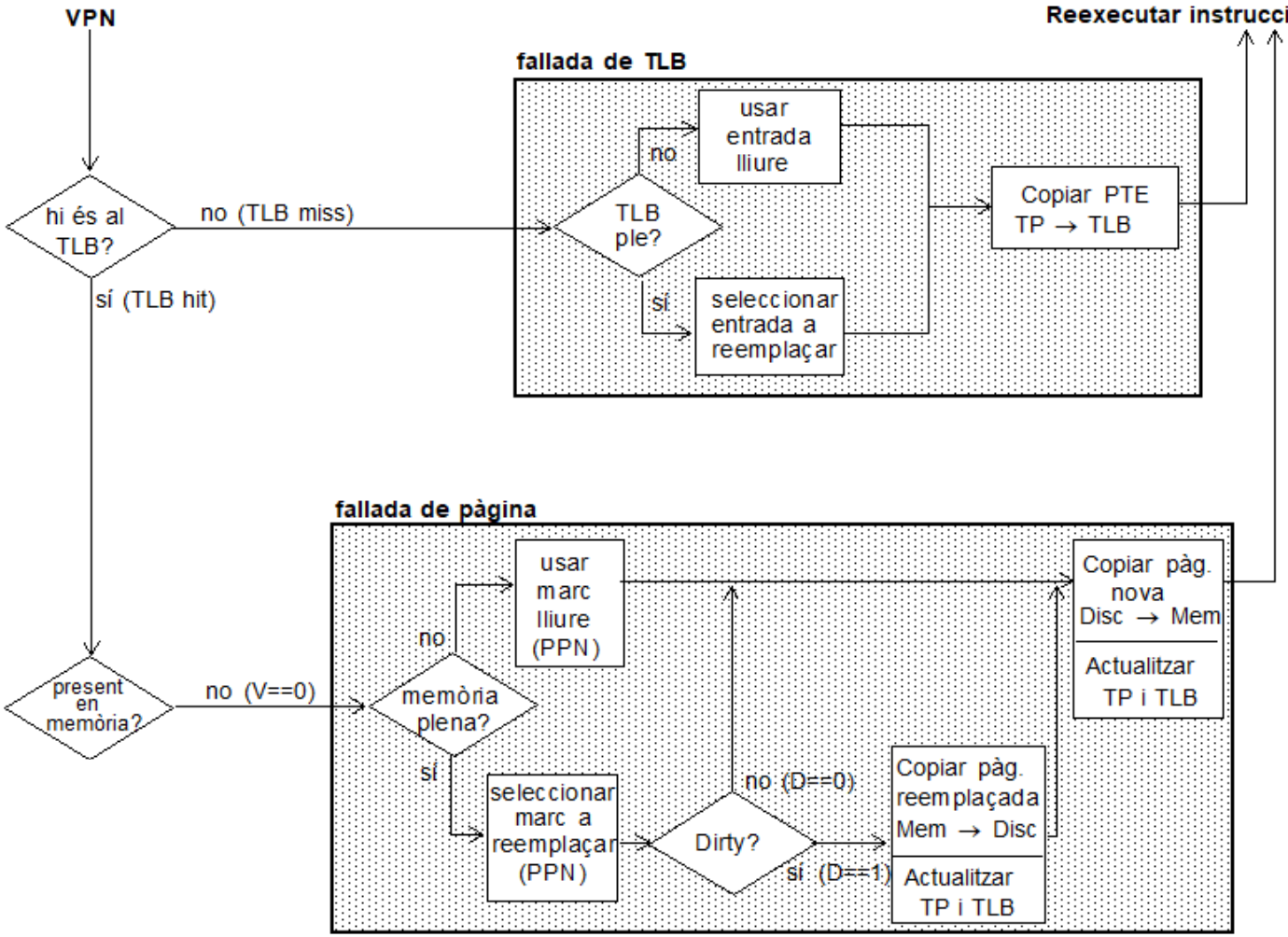
Fallades de TLB i de pàgina: diagrama



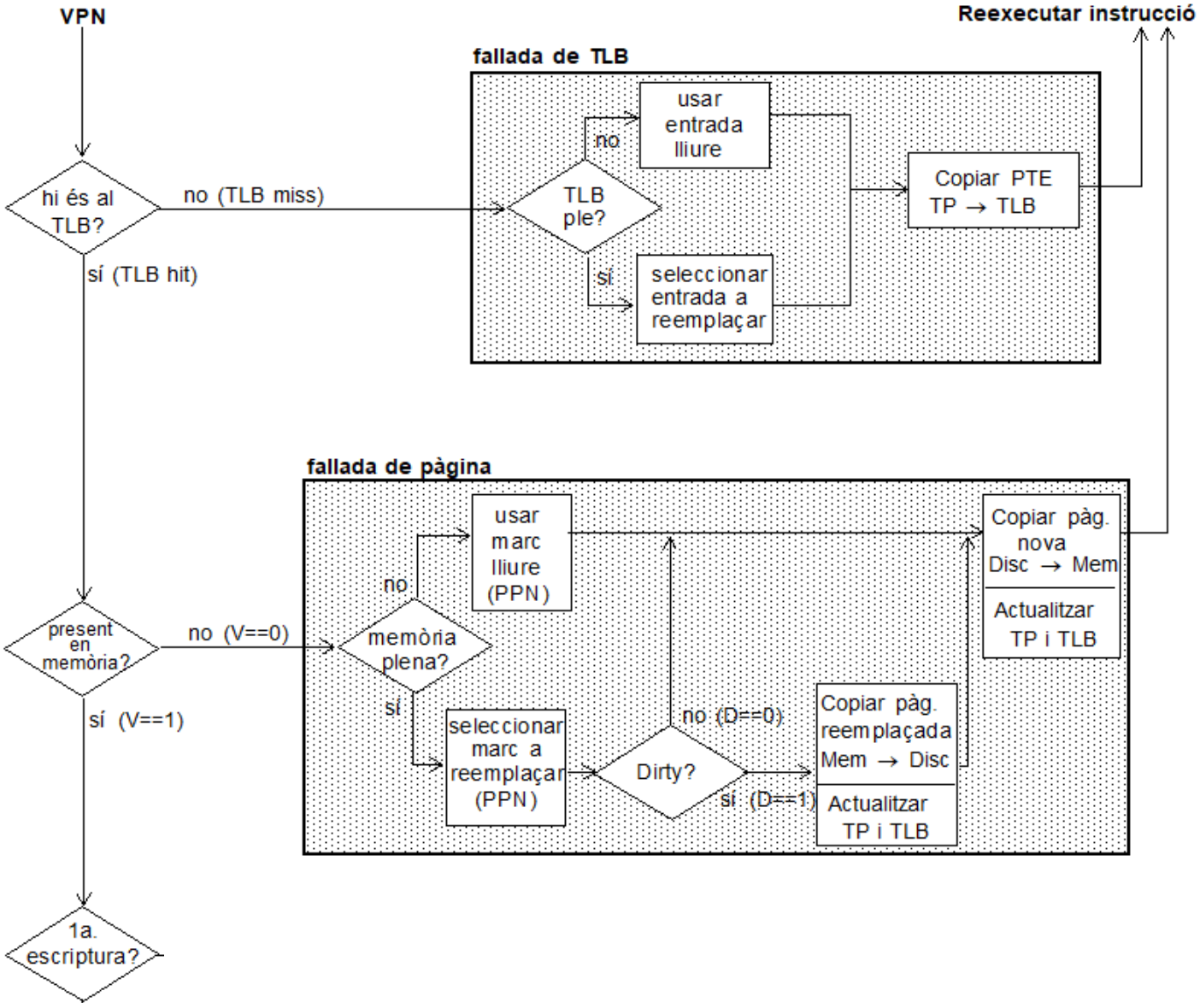
Fallades de TLB i de pàgina: diagrama



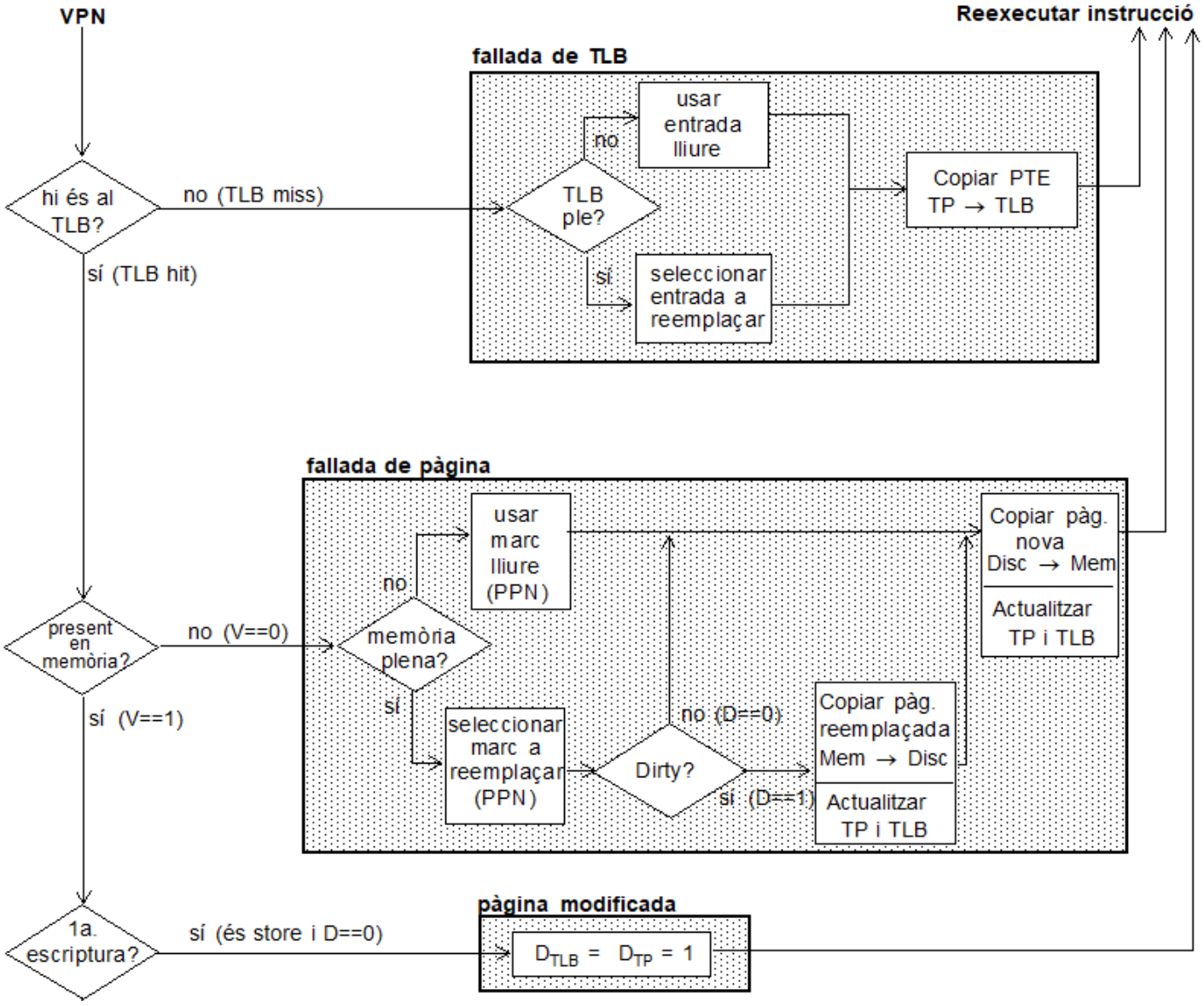
Fallades de TLB i de pàgina: diagrama



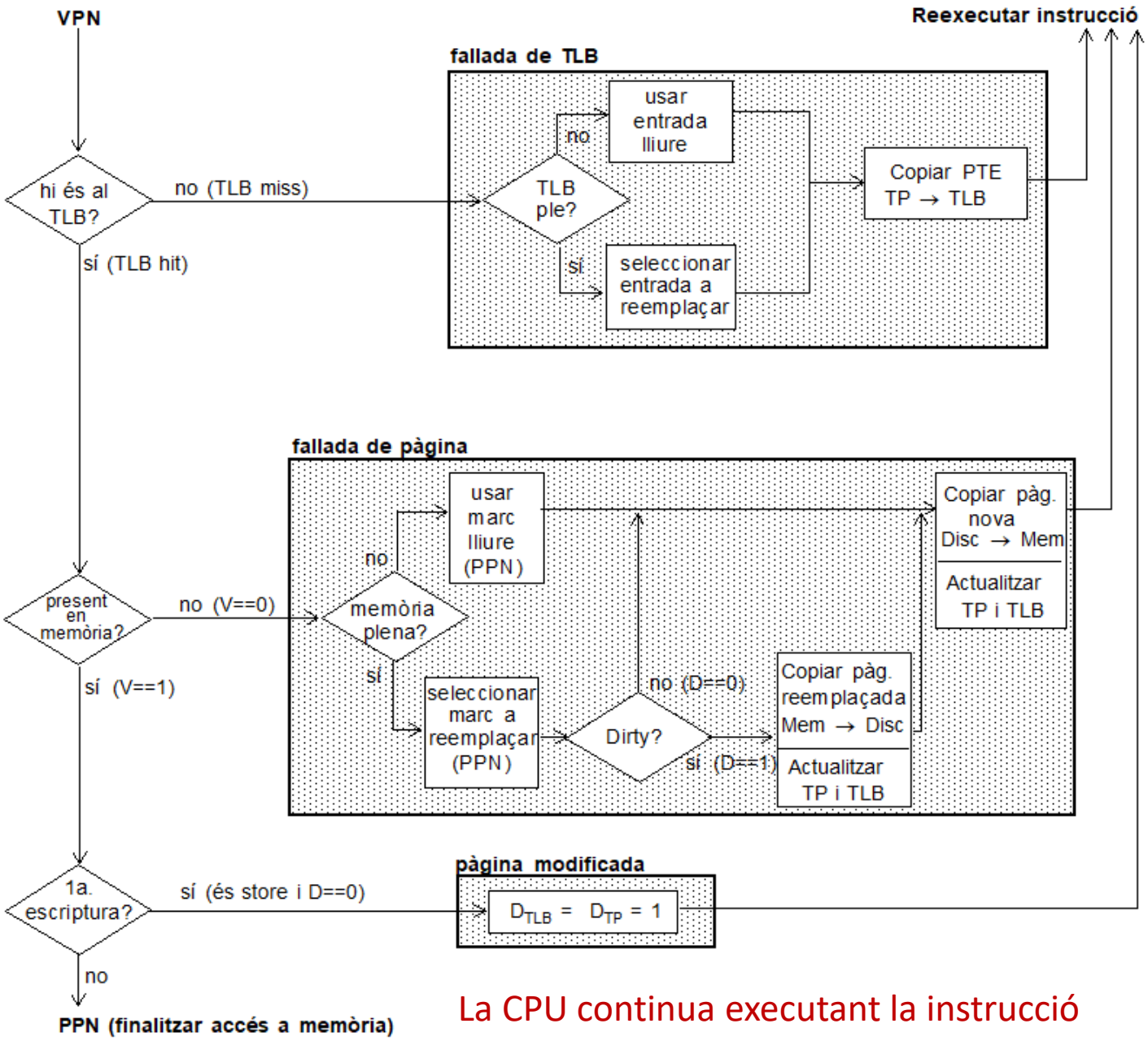
Fallades de TLB i de página: diagrama



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 TLB s'accedeix la TP en MP (*page table walk*)
  - En MIPS hi ha una excepció, i la resol el S.O. (per software)

    → Un cop resolta, la instrucció causant s'ha de reexecutar

Fallades de TLB i de página

- En una fallada de TLB s'accedeix la TP en MP (*page table walk*)
  - En MIPS hi ha una excepció, i la resol el S.O. (per software)
    → Un cop resolta, la instrucció causant s'ha de reexecutar
  - En alguns sistemes (x86, PowerPC) la resol el hardware
    → Un cop resolta, l'accés a memòria prossegueix (com en les caches)

Fallades de TLB i de página

- En una fallada de TLB s'accedeix la TP en MP (*page table walk*)
  - En MIPS hi ha una excepció, i la resol el S.O. (per software)
    → Un cop resolta, la instrucció causant s'ha de reexecutar
  - En alguns sistemes (x86, PowerPC) la resol el hardware
    → Un cop resolta, l'accés a memòria prossegueix (com en les caches)

- En una fallada de pàgina s'accedeix al disc (milions de cicles)
  - Hi ha una excepció, i la resol el S.O. (per software)
    → Mentre es resol, es pot executar un altre programa (canvi de context)
    → Un cop resolta, la instrucció causant s'ha de reexecutar

Fallades de TLB i de página

- En una fallada de TLB s'accedeix la TP en MP (*page table walk*)
  - En MIPS hi ha una excepció, i la resol el S.O. (per software)
    → Un cop resolta, la instrucció causant s'ha de reexecutar
  - En alguns sistemes (x86, PowerPC) la resol el hardware
    → Un cop resolta, l'accés a memòria prossegueix (com en les caches)

- En una fallada de pàgina s'accedeix al disc (milions de cicles)
  - Hi ha una excepció, i la resol el S.O. (per software)
    → Mentre es resol, es pot executar un altre programa (canvi de context)
    → Un cop resolta, la instrucció causant s'ha de reexecutar

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

Protecció amb Memòria Virtual

- Com impedeix la MV que un procés accedeixi a pàgines d'un altre procés?
  - Cada marc de pàgina está assignat a un únic procés, i no apareix a les TP de cap altre procés

Protecció amb Memòria Virtual

- Com impedeix la MV que un procés accedeixi a pàgines d'un altre procés?
  - Cada marc de pàgina está assignat a un únic procés, i no apareix a les TP de cap altre procés
- Podria un procés modificar la seva TP o el TLB?
  - No! El procesador té 2 modes de funcionament: usuari i sistema
  - Les TP es guarden en adreces del S.O. (sols accesibles en mode sistema): en MIPS, adreces lògiques amb el bit 31=1

Protecció amb Memòria Virtual

- Com impedeix la MV que un procés accedeixi a pàgines d'un altre procés?
  - Cada marc de pàgina está assignat a un únic procés, i no apareix a les TP de cap altre procés
- Podria un procés modificar la seva TP o el TLB?
  - No! El procesador té 2 modes de funcionament: usuari i sistema
  - Les TP es guarden en adreces del S.O. (sols accesibles en mode sistema): en MIPS, adreces lògiques amb el bit 31=1
  - El TLB (com tots els dispositius) sols és accesible en mode sistema

Protecció amb Memòria Virtual

- Com impedeix la MV que un procés accedeixi a pàgines d'un altre procés?
  - Cada marc de pàgina está assignat a un únic procés, i no apareix a les TP de cap altre procés
- Podria un procés modificar la seva TP o el TLB?
  - No! El procesador té 2 modes de funcionament: usuari i sistema
  - Les TP es guarden en adreces del S.O. (sols accesibles en mode sistema): en MIPS, adreces lògiques amb el bit 31=1
  - El TLB (com tots els dispositius) sols és accesible en mode sistema
- És convenient prohibir l'escriptura en algunes pàgines
  - P. ex., en pàgines de codi (no es permet el codi automodificable)
  - Bit de permís d'escriptura (E) en cada PTE (a la TP i al TLB)

Protecció amb Memòria Virtual

- Com impedeix la MV que un procés accedeixi a pàgines d'un altre procés?
  - Cada marc de pàgina está assignat a un únic procés, i no apareix a les TP de cap altre procés
- Podria un procés modificar la seva TP o el TLB?
  - No! El procesador té 2 modes de funcionament: usuari i sistema
  - Les TP es guarden en adreces del S.O. (sols accesibles en mode sistema): en MIPS, adreces lògiques amb el bit 31=1
  - El TLB (com tots els dispositius) sols és accesible en mode sistema
- És convenient prohibir l'escriptura en algunes pàgines
  - P. ex., en pàgines de codi (no es permet el codi automodificable)
  - Bit de permís d'escriptura (E) en cada PTE (a la TP i al TLB)
- I si un procés escriu 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

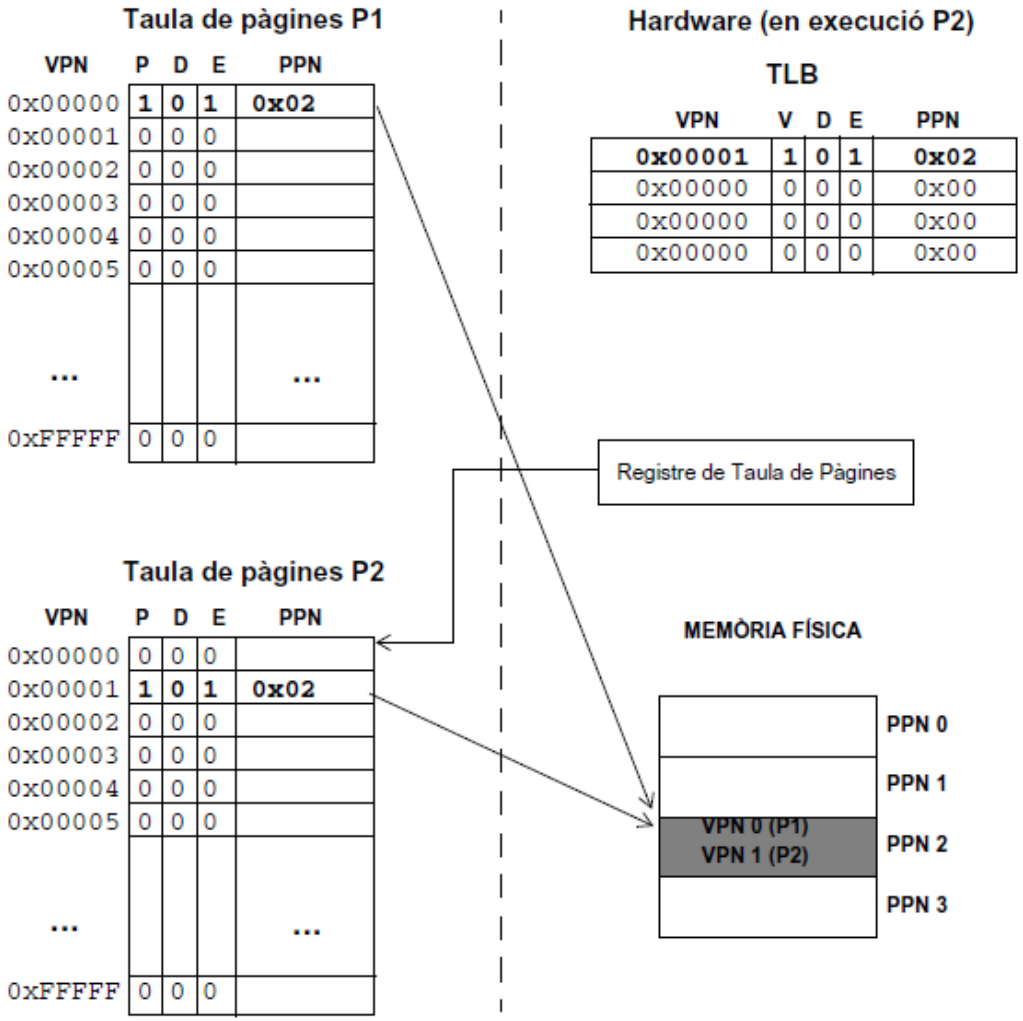
  $\rightarrow$ El S.O. escriu el PPN del marc de página compartit en les dues TPs

Compartició de memòria

- Un procés P1 pot demanar al S.O. compartir una página amb P2

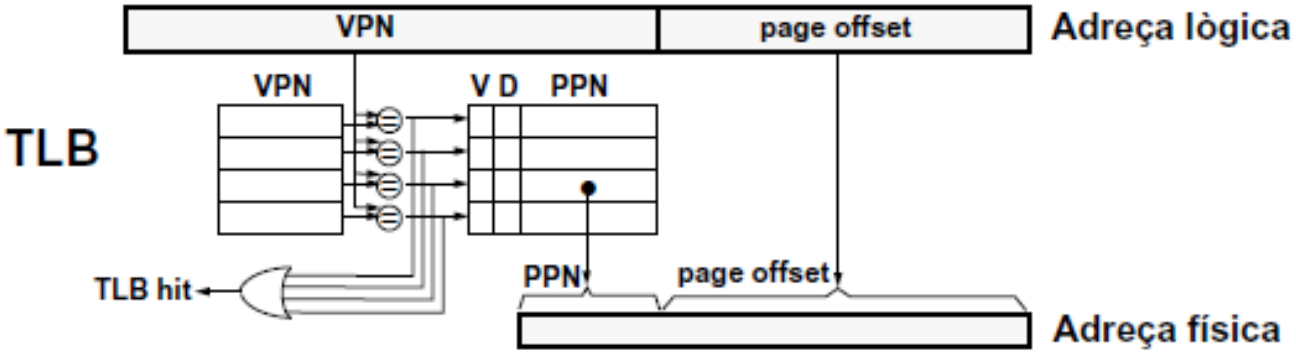
  → El S.O. escriu el PPN del marc de página compartit en les dues TPs

Exemple:
PPN2 apareix a les 2 TPs



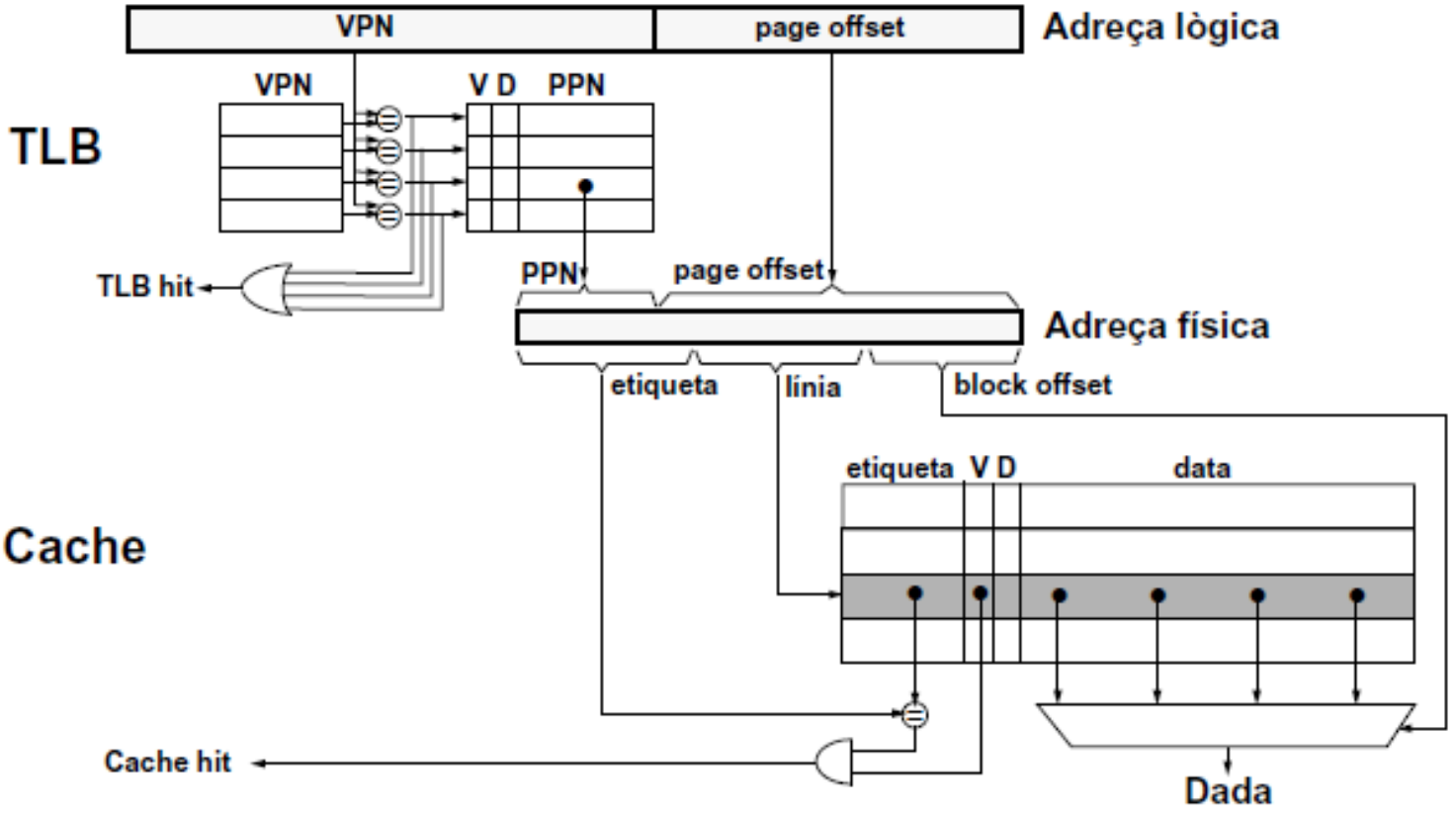
Integració del TLB i la memòria cache

- 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

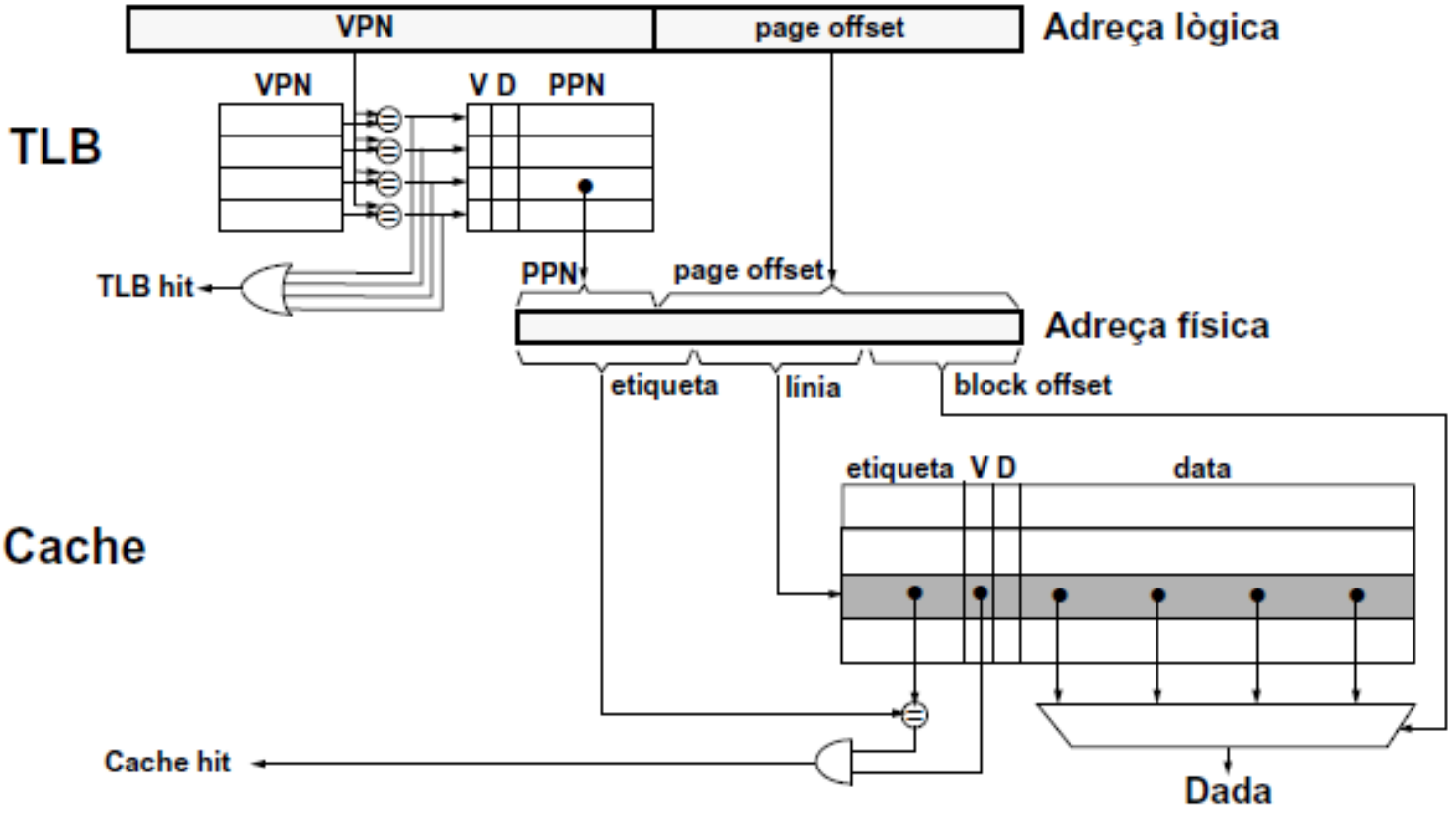
4. En memòria virtual paginada, sempre que reemplacem una pàgina de la memòria física, cal escriure-la en disc

Annex (no entra)

- Integració de TLB i caches
- Taula de Pàgines multinivell

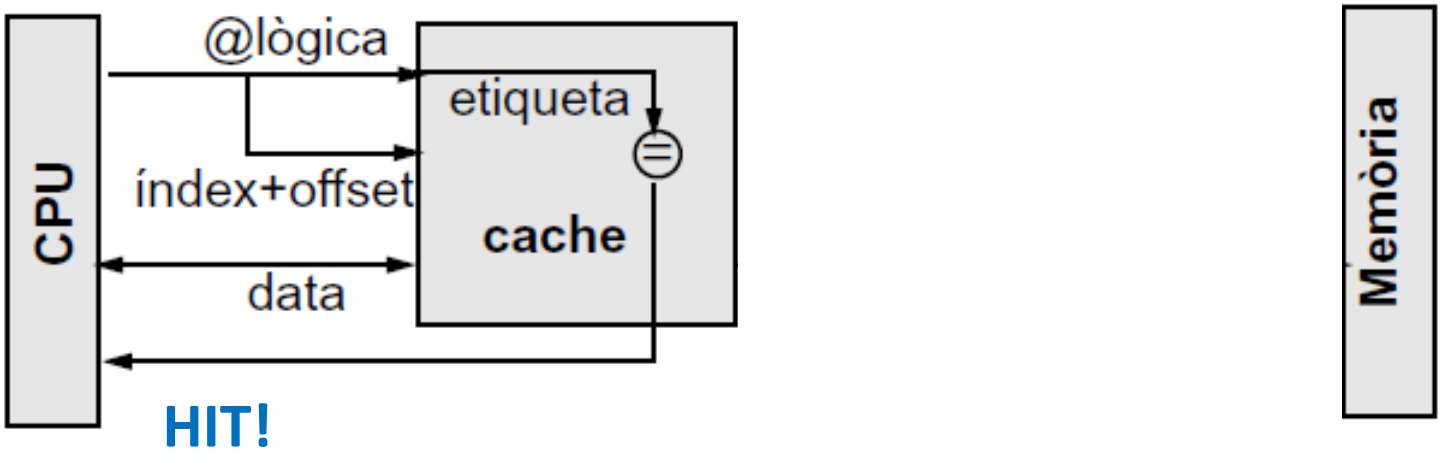
Integració del TLB i la memòria cache

- Hem vist abans: Cache **indexada físicament**
  - Simple, però té un temps d'accés elevat (s'accedeix seqüencialment a TLB i cache)



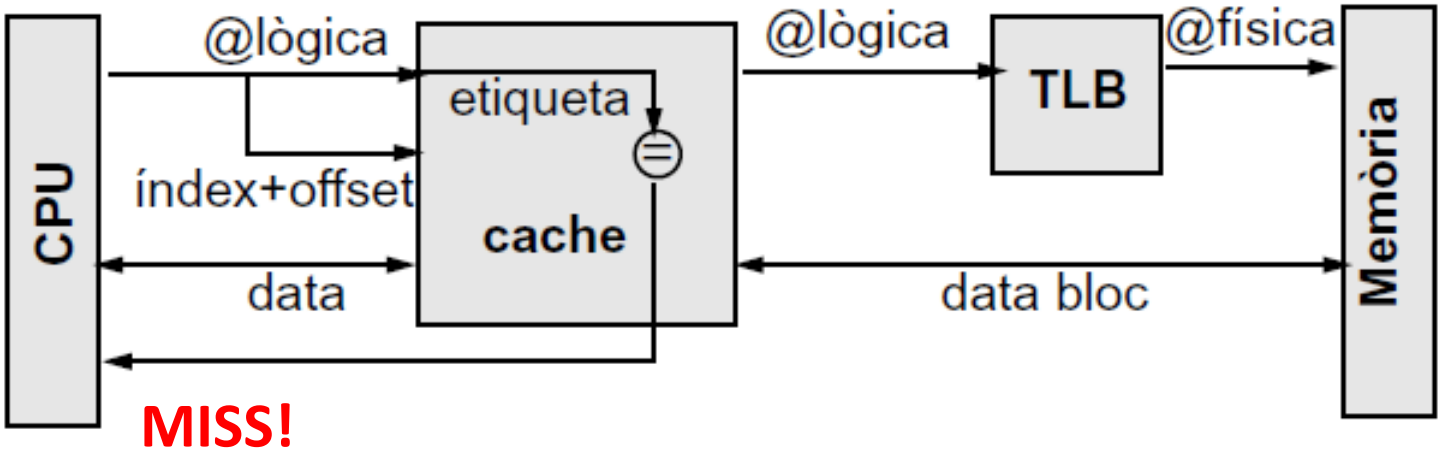
Integració del TLB i la memòria cache

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



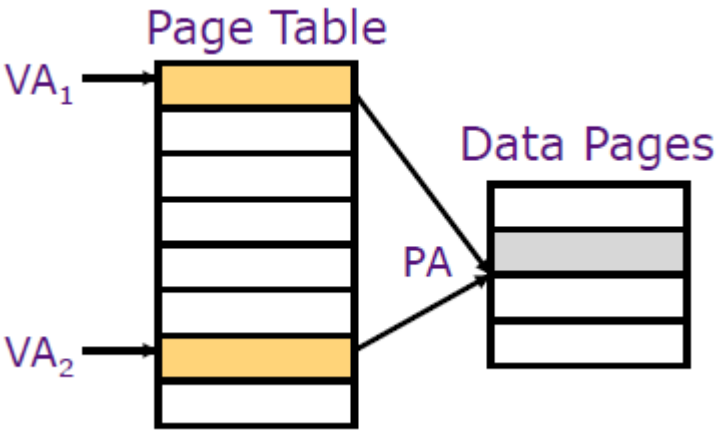
Integració del TLB i la memòria cache

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



Integració del TLB i la memòria cache

- Cache **indexada virtualment**
  - Temps d'accés menor: en cas d'encert, no cal traduir l'adreça, només en cas de fallada
  - 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
    - P.ex. quan un procés en crea un altre, sovint comparteixen codi

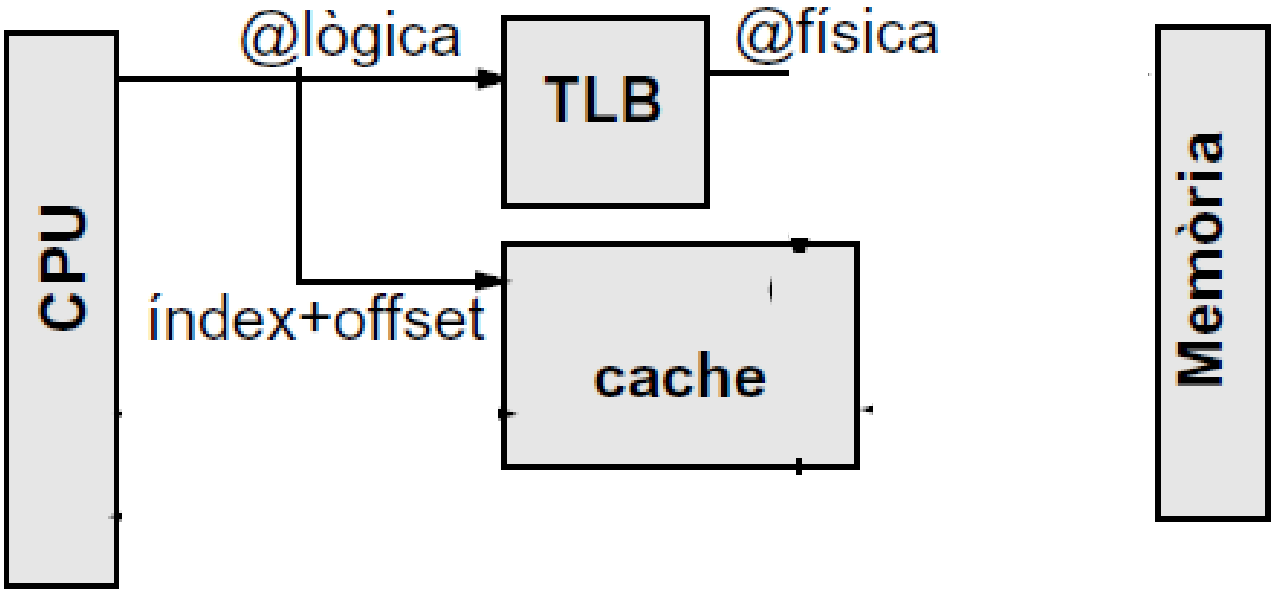


Cache

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

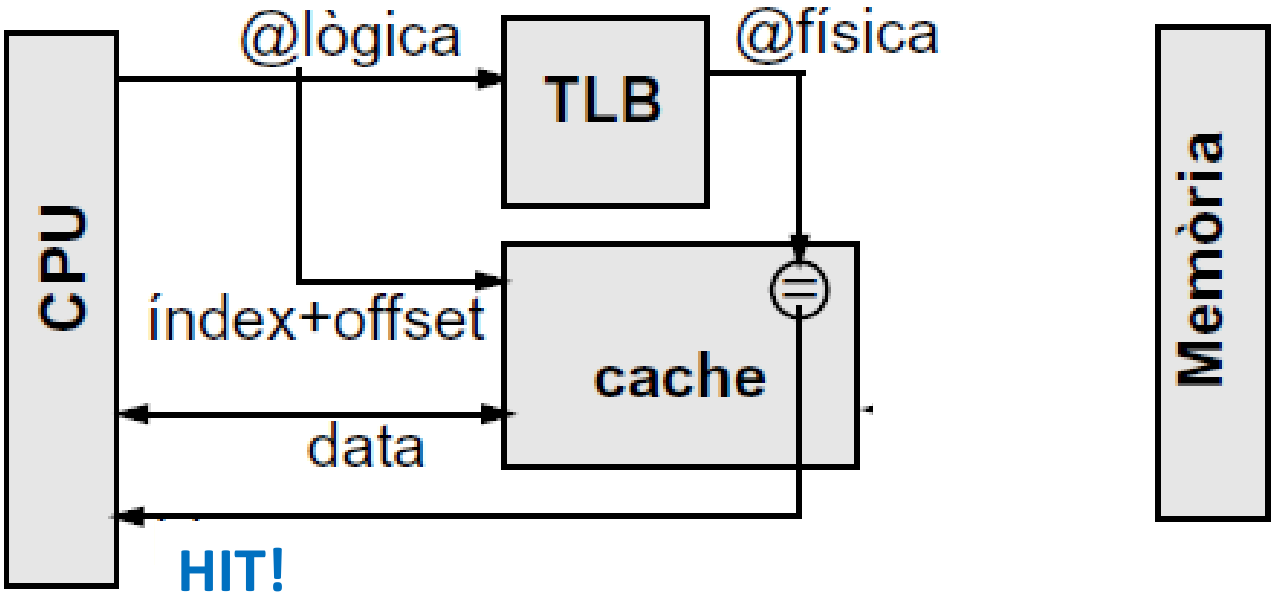
Integració del TLB i la memòria cache

- Cache **indexada virtualment i etiquetada físicament**
  - S'accedeix a TLB i la cache simultàniament
  - L'índex de línea o conjunt s'obté dels bits de page-offset, que no s'han de traduir



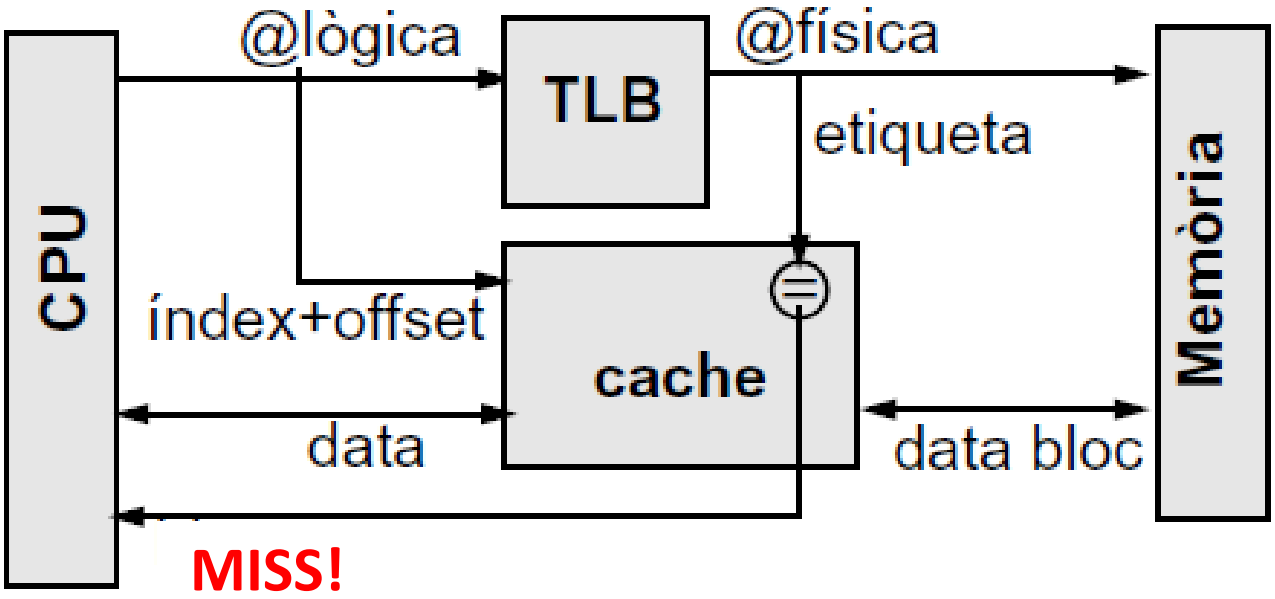
Integració del TLB i la memòria cache

- Cache **indexada virtualment i etiquetada físicament**
  - S'accedeix a TLB i la cache simultàniament
  - L'índex de línea o conjunt s'obté dels bits de page-offset, que no s'han de traduir
  - Les etiquetes es comproven al final, amb l'adreça física
    - En cas d'encert



Integració del TLB i la memòria cache

- Cache **indexada virtualment i etiquetada físicament**
  - S'accedeix a TLB i la cache simultàniament
  - L'índex de línea o conjunt s'obté dels bits de page-offset, que no s'han de traduir
  - Les etiquetes es comproven al final, amb l'adreça física
    - En cas de fallada



TP multinivell

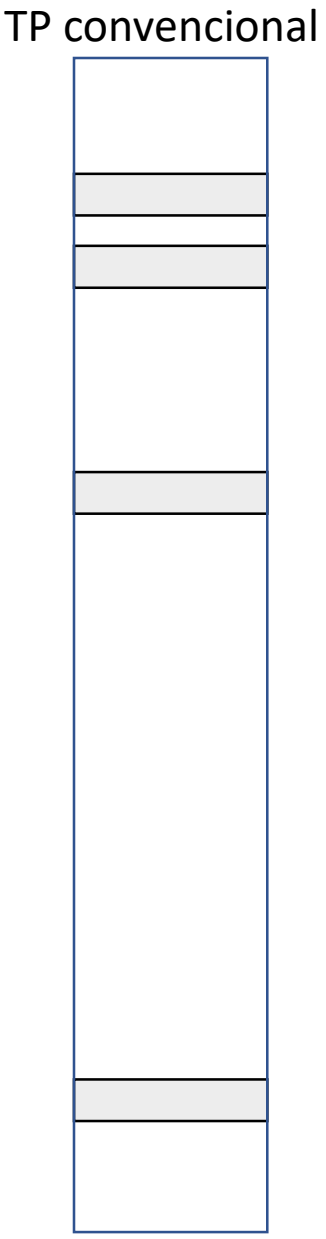
- Vegem el **problema,** amb quatre números. Suposem:
  - Espai lògic de 64 bits
  - Memòria física de 43 bits
  - Pàgines de 8KB = 2^{13} bytes

- Quanta memòria cal per implementar la TP?
  - Offset de 13 bits
  - VPN de 64 – 13 = 51 bits $\rightarrow$ TP de 2^{51} entrades
  - PPN de 43 – 13 = 30 bits
  - Cada entrada de la TP (PTE) té
    - PPN (30 bits) + P (1 bit) + D (1 bit) = 32 bits = 4 bytes

- En total
  - 2^{51} entrades x 4 bytes = 2^{53} bytes = 8 Petabytes (!!!)

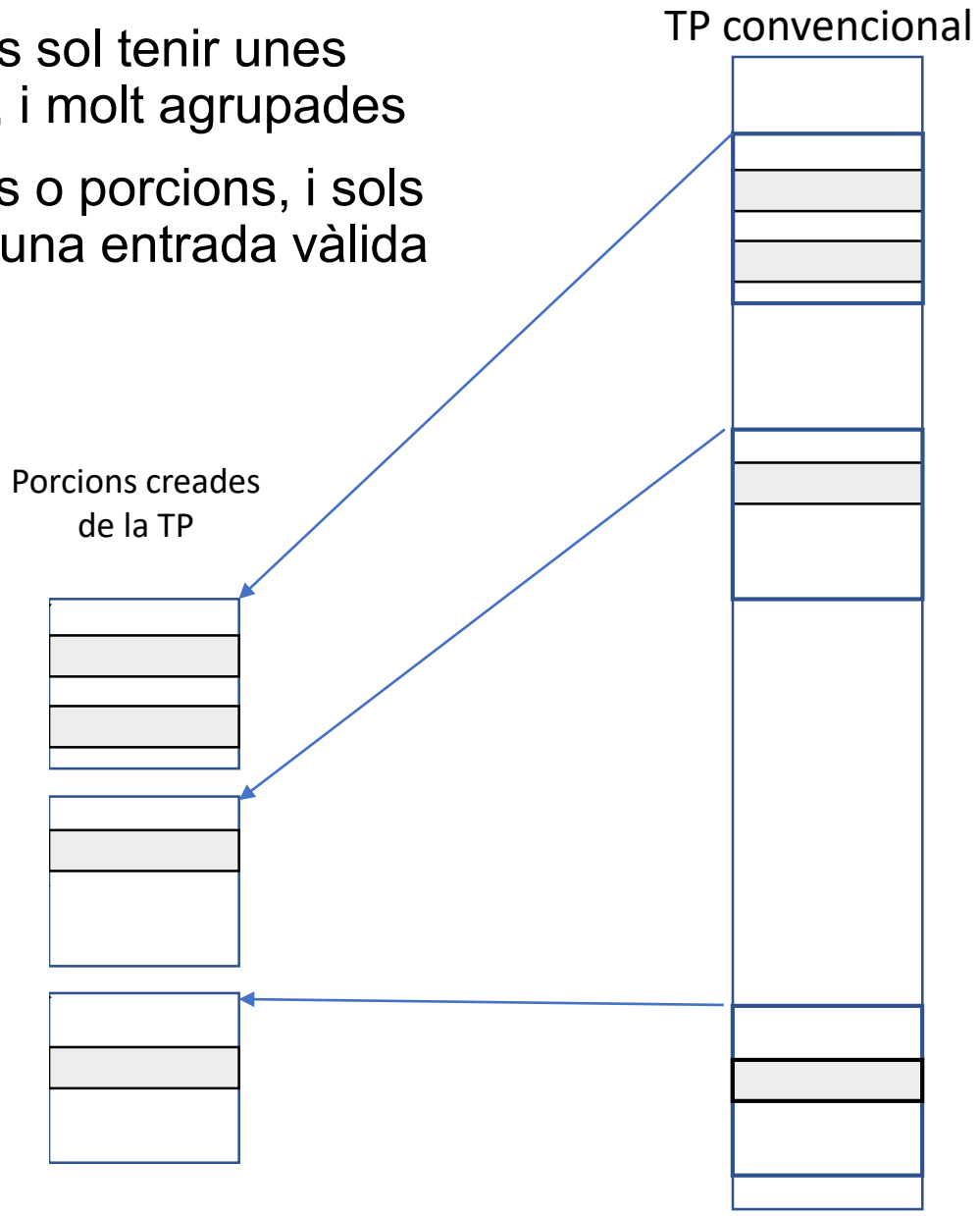
Solució: TP multinivell

- Observació: cada procés sol tenir unes poques pàgines vàlides, i molt agrupades



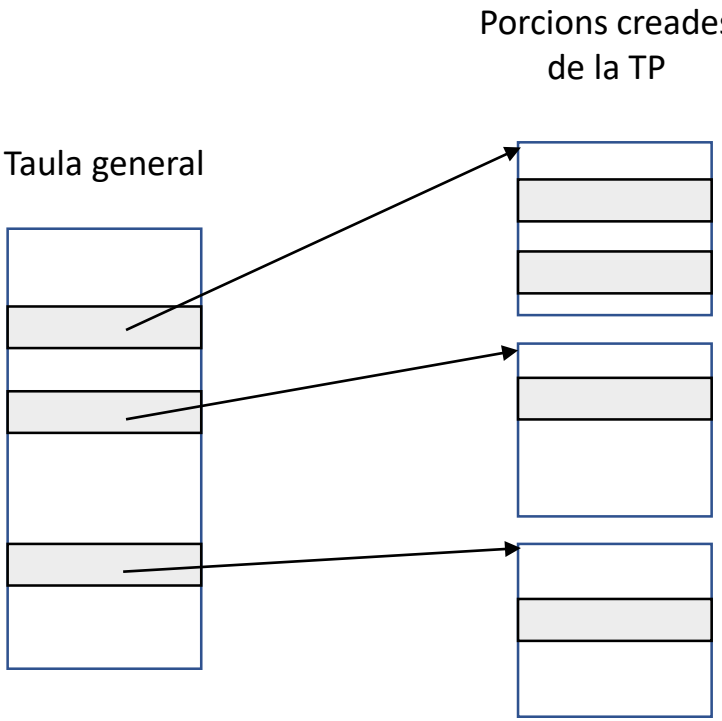
Solució: TP multinivell

- Observació: cada procés sol tenir unes poques pàgines vàlides, i molt agrupades
- Dividim la TP en pàgines o porcions, i sols creem aquelles amb alguna entrada vàlida

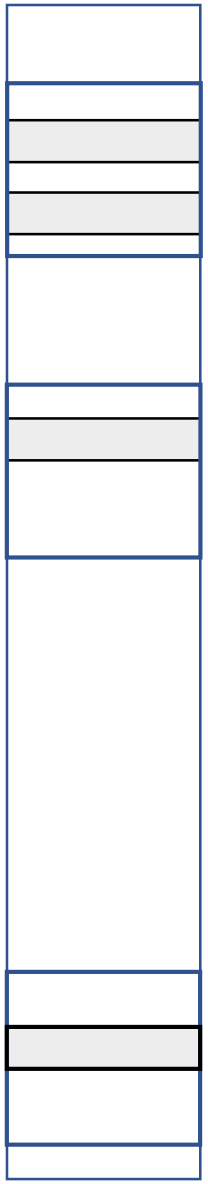


Solució: TP multinivell

- Observació: cada procés sol tenir unes poques pàgines vàlides, i molt agrupades
- Dividim la TP en pàgines o porcions, i sols creem aquelles amb alguna entrada vàlida
- Generem una taula general amb punters a les porcions de la TP

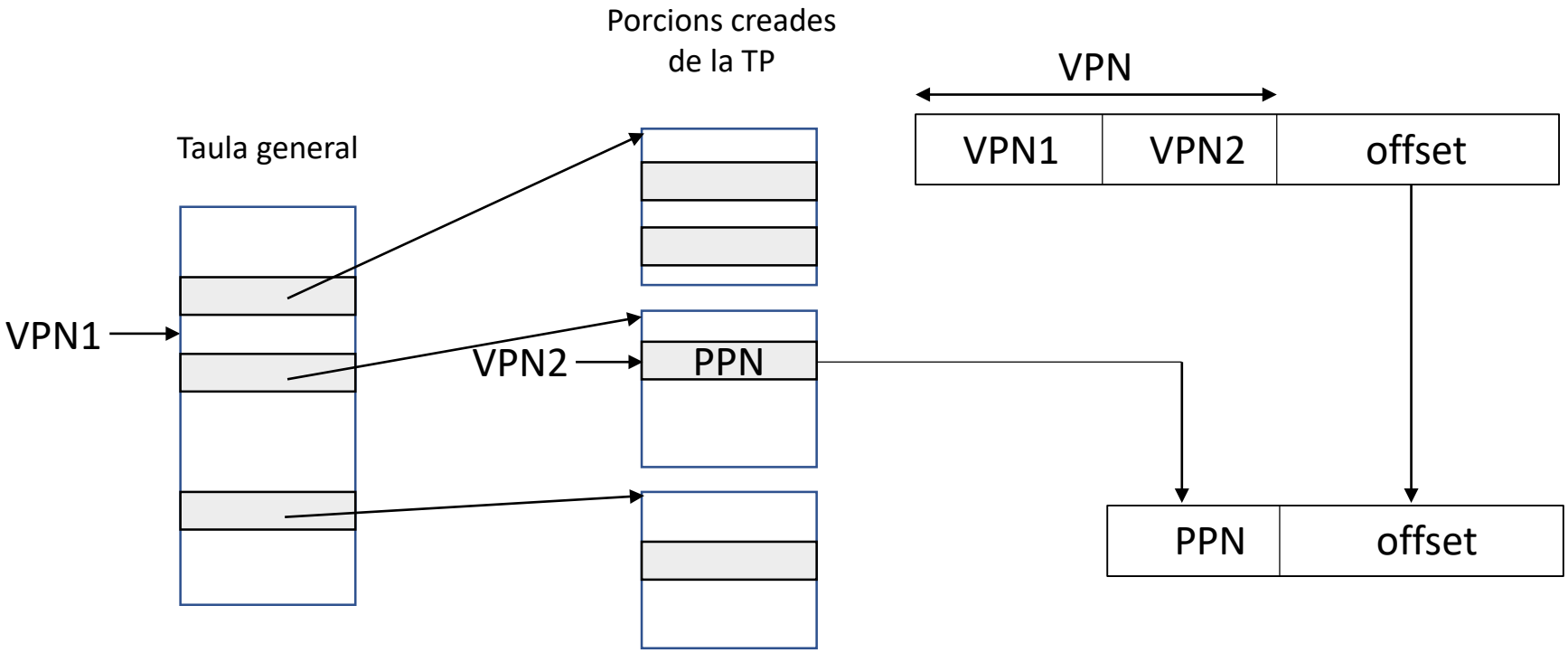


TP convencional



Solució: TP multinivell

- Observació: cada procés sol tenir unes poques pàgines vàlides, i molt agrupades
- Dividim la TP en pàgines o porcions, i sols creem aquelles amb alguna entrada vàlida
- Generem una taula general amb punters a les porcions de la TP



PT multinivell: exemple