Apunts del Tema 7
Memòria Virtual

Rubèn Tous
Joan Manuel Parcerisa
Jordi Tubella

Departament d'Arquitectura de Computadors
Facultat d'Informàtica de Barcelona
Juny 2018

Licencia Creative Commons

Tema 7. Memòria Virtual

5.4

1. Introducció

En el Tema 3, vam veure que durant els processos de compilació, assemblatge i enllaçat s'assignen adreces de memòria absolutes a les instruccions i les dades d'un programa. Però, si en un computador modern executem múltiples programes simultàniament, tots ells compartint la memòria, aquests programes s'haurien de reubicar en el moment d'executar-se per evitar conflictes. I, què ocorre si el conjunt d'instruccions i dades dels programes en execució supera la grandària de la memòria física del computador? En un principi ens impossibilitaria l'execució dels programes.

La solució la trobem en una tècnica que s'anomena *memòria virtual*. Per un costat, la memòria virtual permet que la memòria del computador sigui compartida per múltiples programes, implementant la **reubicació** dels programes a través d'un mecanisme de traducció d'adreces, i proporcionant mecanismes eficients i segurs de **protecció** i **compartició** entre ells. Per un altre costat, permet a un o més programes excedir la **capacitat de la memòria** principal, gràcies a l'ús de l'emmagatzematge secundari, ja sigui disc o SSD (en endavant ens referirem simplement al disc, per brevetat) com un nivell més en la jerarquia de memòria del computador.

1.1 Espai d'adreçament físic i espai d'adreçament lògic

La memòria virtual consisteix essencialment en fer servir dos espais d'adreçament diferents. Per un costat, les adreces que hi haurà als programes (quan els compilem i també quan els executem) seran adreces d'un *espai d'adreçament lògic* o virtual. Aquest espai d'adreçament és exclusiu de cada programa, i la seva grandària és la màxima permesa pel número de bits d'adreça que faci servir el computador. Aquest espai d'adreçament aïlla de la complexitat real (múltiples programes, memòria física limitada) i ofereix la impressió de que només hi ha un programa i de que es disposa de tanta memòria com sigui necessari. Les adreces amb què treballa un compilador són adreces d'aquest espai d'adreçament, *adreces lògiques*. Les adreces amb que treballarà la CPU també seran adreces lògiques.

Per un altre costat, al conjunt d'adreces de la memòria física (les adreces "reals") l'anomenarem *espai d'adreçament físic*. Durant l'execució d'un programa, caldrà carregar-lo (si més no la part que calgui executar) en alguna posició lliure de l'espai d'adreçament físic. La memòria virtual defineix un mecanisme que permet "recordar" a quines *adreces físiques* s'han carregat determinades adreces lògiques d'un programa. Gràcies a aquesta informació, quan la CPU sol·licita al subsistema de memòria una dada (o una instrucció) corresponent a una adreça lògica determinada, la Unitat de Gestió de Memòria

(MMU per les seves sigles en anglès) "tradueix" prèviament l'adreça lògica per l'adreça física d'on realment es troba la dada, i és aquesta adreça la que s'envia a la memòria.

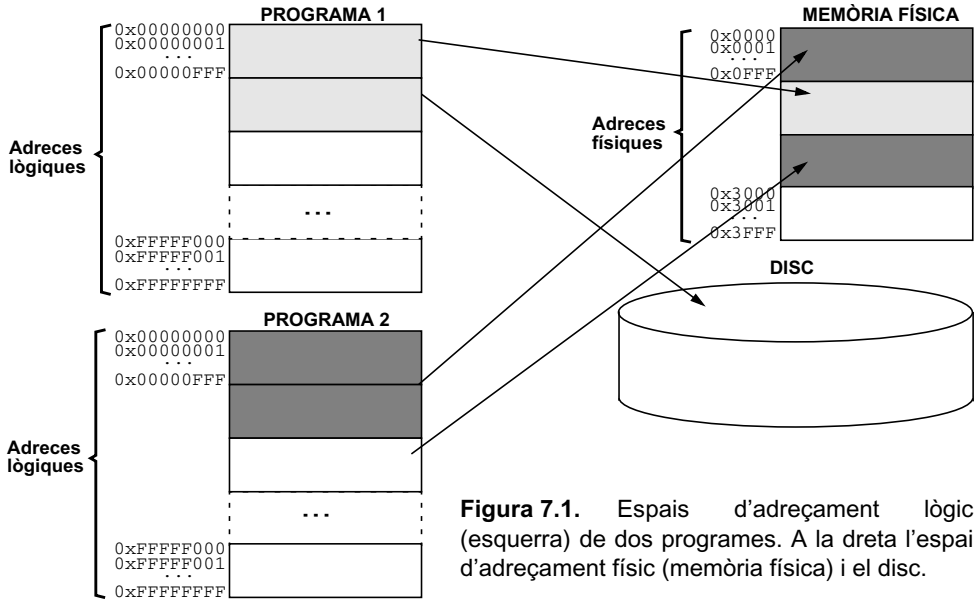

Figura 7.1. Espais d'adreçament lògic (esquerra) de dos programes. A la dreta l'espai d'adreçament físic (memòria física) i el disc.

1.2 El disc com un nivell més en la jerarquia de memòria

Una de les motivacions originals de la memòria virtual era donar solució al problema de no poder encabir tot un programa a la memòria física, la grandària de la qual era antigament molt limitada. Tradicionalment, això s'havia solucionat mitjançant la tècnica dels *overlays*. Aquesta tècnica consisteix en que el programador divideix el programa en blocs, que va movent selectivament entre la memòria física i el disc, mantenint en memòria únicament els blocs indispensables en cada moment. La memòria virtual funciona d'una manera semblant, ja que també fa servir el disc per emmagatzemar temporalment les parts d'un programa que no s'estan fent servir, però això ho gestiona el sistema operatiu, amb l'ajuda del hardware, alliberant als programadors d'aquesta tasca. En un computador amb memòria virtual, el disc passaria a ser una capa més de la jerarquia de memòria que vam veure al Tema 6. De la mateixa manera que la memòria cache només conté un subconjunt de les dades del nivell inferior (de la memòria principal), la memòria física d'un computador amb memòria virtual només conté un subconjunt de les dades necessàries per a l'execució d'un programa. La totalitat de les dades estaria sempre emmagatzemada al nivell inferior (el disc), que es troba a la base de la jerarquia.

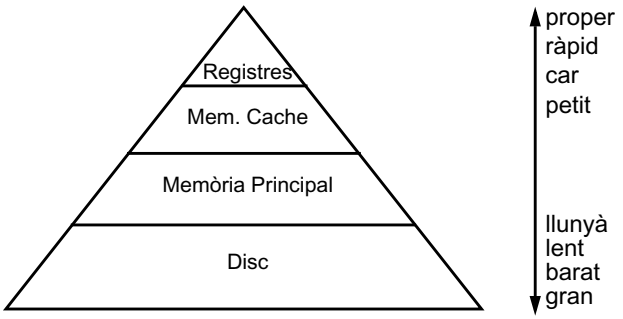

Figura 7.2. **Figura 7.2** Jerarquia de memòria en un computador amb memòria

5.4

2. Funcionament de la memòria virtual

2.1 Unitat bàsica de gestió de memòria: la pàgina

Tot i que els fonaments del funcionament de la memòria virtual i les memòries caches són els mateixos, els seus orígens històrics diferents fan que fem servir una terminologia diferent per descriure'n el funcionament. L'equivalent als blocs de memòria cache en memòria virtual s'anomenen *pàgines,* i les fallades de memòria virtual s'anomenen *fallades de pàgina*[1].

Una pàgina lògica, o pàgina virtual, és cada un dels blocs de memòria contigus i de grandària fixa T bytes (per exemple T = 4KB = 2^{12} bytes) en què es subdivideix l'espai d'adreçament lògic de la CPU. Cada pàgina tindrà associat un *número de pàgina virtual* (virtual page number o VPN), que indicarà la posició que ocupa dins d'aquest espai d'adreçament lògic, començant pel 0. Donada una adreça lògica d'un programa A, per exemple la d'una dada, podem determinar a quin número de pàgina virtual VPN correspon fent la divisió entera entre l'adreça i la grandària de pàgina (VPN = A/T). Si, per exemple, tenim que A = 0x10010004 i T = 4KB = 2^{12}, podem determinar que A està a la pàgina virtual amb VPN = 0x10010. Donat que treballarem amb valors de T potències de dos (T = 2^t), determinar el VPN consistirà simplement en agafar l'adreça i descartar-ne els t bits de menor pes. Aquests t bits de menor pes indiquen quina és la ubicació de la dada dins la pàgina, i els anomenarem *desplaçament* (page offset en anglès).

	VPN	page offset
@ = 0x10010004 =	0001 0000 0000 0001 0000	0000 0000 0100

Cada computador disposa d'una determinada capacitat de memòria física, ocupant un rang d'adreces que anomenem espai d'adreçament físic, i que és totalment independent del rang d'adreces lògiques (pot ser menor, igual o major). L'espai d'adreçament físic es subdivideix en blocs de grandària T (la mateixa que les pàgines lògiques). Cada subdivisió l'anomenem *marc de pàgina* ("page frame" en anglès) i podem imaginar-la com un "contenidor" que pot allotjar una pàgina lògica. Cada marc de pàgina té associat un *número de pàgina física* ("physical page number" o PPN) que indicarà la posició que ocupa dins d'aquest espai d'adreçament físic, començant pel 0.

Cada programa consta d'un cert nombre de pàgines virtuals, no totes necessàriament contígües, ocupant unes adreces que queden determinades en temps de compilació. Les adreces d'aquestes pàgines virtuals són totalment independents de les adreces físiques que ocuparan posteriorment en temps d'execució. Inicialment, un programa resideix en l'emmagatzematge secundari (disc), i durant la seva execució caldrà anar carregant del disc a la memòria física les pàgines que es vagin necessitant. Si no hi queden marcs de pàgina lliures, llavors el sistema haurà de reemplaçar-ne un d'ocupat, i possiblement l'hagi d'escriure al disc abans de carregar la pàgina nova. En un moment donat, les pàgines d'un programa que no estiguin a la memòria física estaran al disc. Passem per alt, de moment, els detalls sobre la forma en què les pàgines es llegeixen o s'escriuen al disc. Assumirem, per simplificar-ho, que la memòria física conté un subconjunt de les pàgines d'un programa, la totalitat de les quals es troba al disc. Per tant, el disc jugarà, respecte a la

1. Aquí sempre ens referirem a memòria virtual paginada però també existeix la memòria virtual segmentada, que divideix l'espai d'adreçament lògic en segments de grandària variable.

memòria física, el mateix paper que la memòria física jugava respecte a la memòria cache al Tema 6.

La Figura 7.3 mostra una possible disposició de les pàgines de dos programes en un moment determinat. Assumint adreces de 32 bits i una mida de pàgina T = 4KB = 2^{12} bytes, l'espai d'adreçament lògic de cada programa seria de 2^{32} bytes = 4GB = 2^{20} pàgines virtuals (2^{32} / 2^{12}). No obstant, a l'exemple els dos programes només fan servir 2 pàgines virtuals cadascun (VPN 0 i VPN 1). A l'exemple s'hi mostra una memòria física de 2^{14} bytes = 16KB = 4 marcs de pàgina (2^{14} / 2^{12}), PPN 0, PPN 1, PPN 2 i PPN 3. A l'exemple es mostra una possible disposició de les pàgines en què algunes estan carregades a la memòria física (VPN 0 del programa 1, VPN 0 i VPN 1 del programa 2) i d'altres (VPN 1 del programa 1) només estan al disc. L'assignació de pàgines a marcs de pàgina la realitza el sistema operatiu en funció de l'espai disponible en cada moment, de manera similar a com funciona una memòria cache completament associativa.

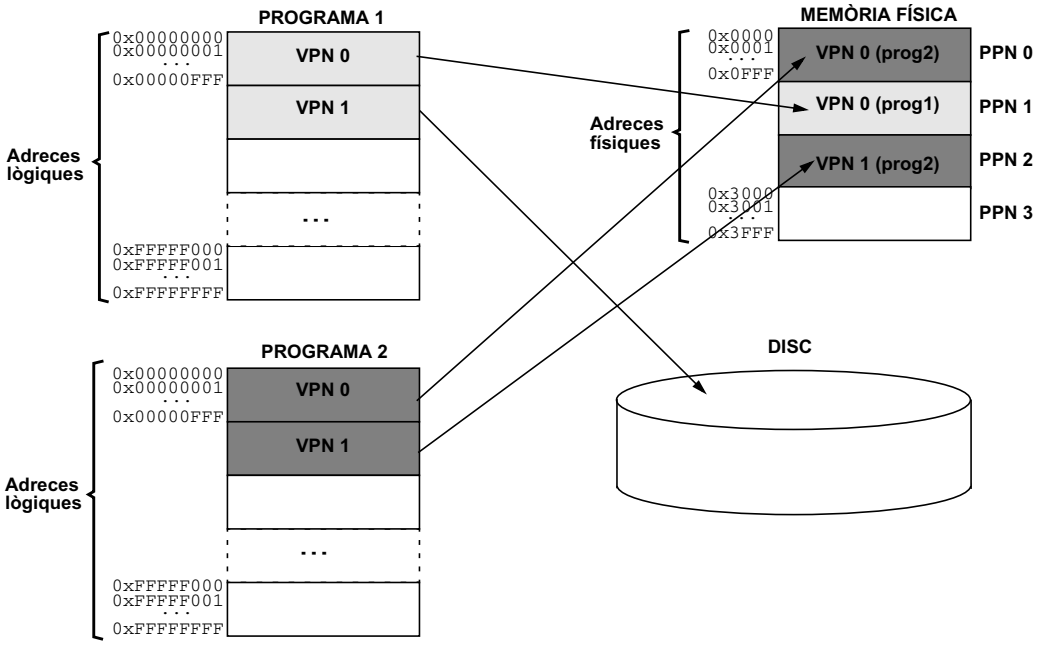

Figura 7.3. Números de pàgina lògica (VPN) a dos programes (esquerra). A la dreta, números de pàgina física (PPN).

2.2 Traducció d'adreces

En un computador amb memòria virtual, el processador treballa amb adreces lògiques. Cada vegada que necessita llegir o escriure una dada de memòria, per exemple per fer fetch d'una instrucció, pregunta per l'adreça lògica d'aquesta dada a la *unitat de gestió de memòria* (en anglés memory managment unit o MMU). Mitjançant un procés anomenat *traducció d'adreces*, la MMU tradueix l'adreça lògica sol·licitada per l'adreça física on realment es troba la dada. Donat que les dades es troben agrupades en pàgines, l'únic que ha de fer el sistema de traducció és, donat el número de pàgina (el VPN) a la qual pertany l'adreça, determinar en quin marc de pàgina (PPN) de la memòria física es troba. L'adreça física on es troba la dada s'obtindrà reemplaçant a l'adreça lògica els bits corresponents al VPN pel PPN. La Figura 7.4 il·lustra el mecanisme de traducció en base a les dades de l'exemple anterior, amb adreces de 32 bits, una mida de pàgina T = 4KB = 2^{12} bytes i una memòria física de 16KB = 2^{14} bytes.

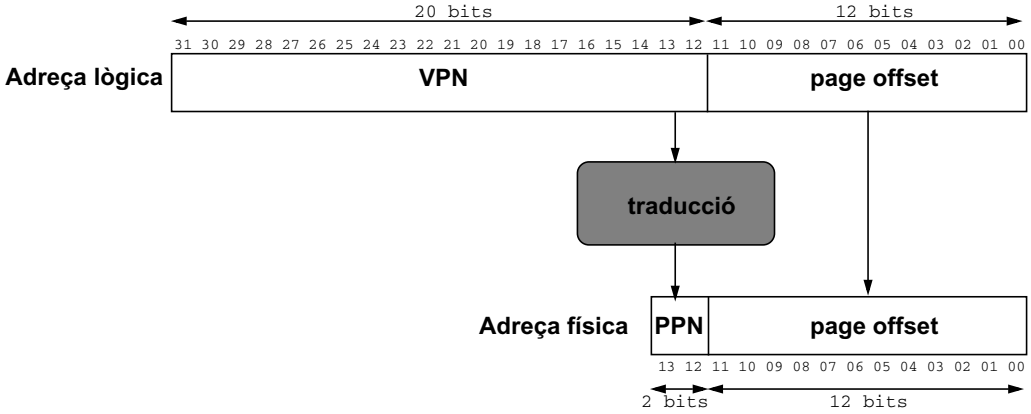

Figura 7.4. Traducció d'una adreça lògica a una adreça física.

2.3 Localització fàcil amb taula de pàgines

Com s'ha vist a la secció anterior, per poder traduir una adreça lògica en una de física cal determinar en quin marc de pàgina (PPN) es troba la pàgina lògica (VPN) a la que correspon l'adreça. La selecció de a quin marc de pàgina es carrega una pàgina donada la realitza el sistema operatiu en funció de l'espai disponible i d'altres variables. Per tant, es tracta d'un emplaçament completament associatiu, seguint la terminologia explicada al Tema 6. Per poder recordar després, durant el procés de traducció d'adreces, a quin marc de pàgina s'ha carregat la pàgina amb un VPN determinat, es fa servir una taula anomenada *taula de pàgines*. Hi ha una taula de pàgines per cada programa. Cada fila de la taula de pàgines s'anomena *entrada de la taula de pàgines* (en anglès page table entry o PTE). Hi ha tantes entrades com pàgines virtuals hi hagi[2] (2^{20} entrades a l'exemple anterior).

Cada entrada de la taula conté també un "bit de presència" (P). Aquest bit val 1 tan sols si la pàgina forma part de l'espai lògic del programa (el codi, les dades i la pila) i a més a més la pàgina en qüestió ocupa un marc de pàgina en memòria física. Si val 0,

2. En la pràctica, emmagatzemar totes les entrades possibles serà inconvenient (especialment amb adreces de 64 bits). Existeixen diferents alternatives per solucionar aquest problema, com les *taules de pàgines multi-nivell* o *jeràrquiques* (utilitzades per Linux).

indica que la pàgina lògica és invàlida o bé que resideix tan sols en l'emmagatzematge secundari (disc), com s'ha comentat a la secció 1.2.

La Figura 7.5 il·lustra com la taula de pàgines implementa el mecanisme de traducció d'adreces explicat a la secció anterior.

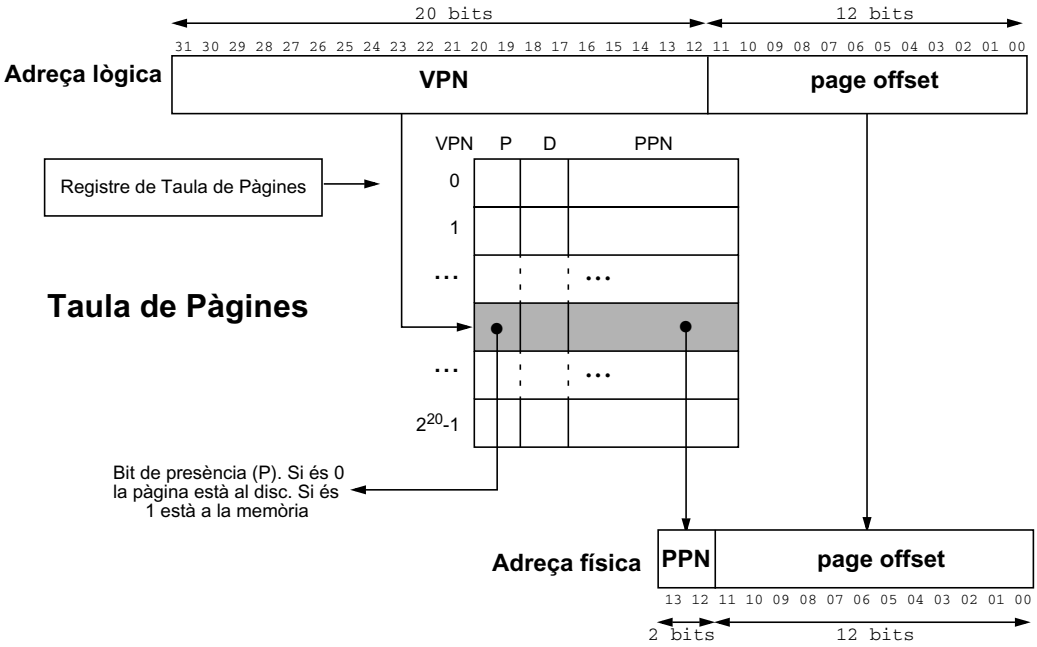

Figura 7.5. Donat un VPN, la taula de pàgines ens diu si la pàgina és a la memòria i a quin marc de pàgina (PPN) es troba.

La Figura 7.6 mostra com es realitzaria la traducció de l'adreça lògica 0x00001801 mitjançant la taula de pàgines corresponent al Programa 2 de la Figura 7.3.

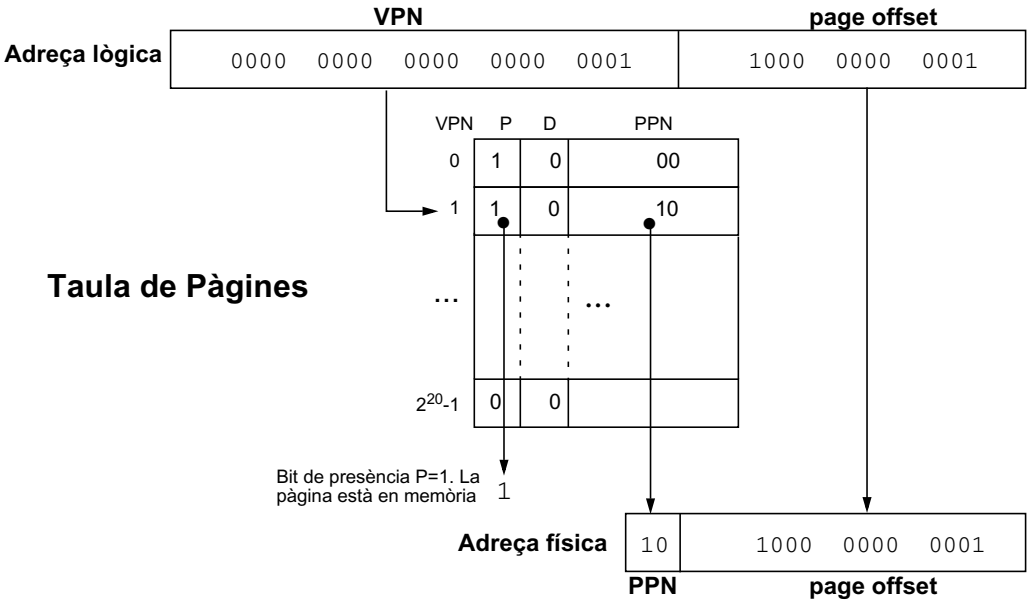

Figura 7.6. Traducció de l'adreça lògica 0x00001801 mitjançant la taula de pàgines corresponent al Programa 2 de la Figura 7.3.

En un sistema multiprogramat, diversos programes poden executar-se de manera concurrent en una única CPU seguint una organització anomenada de "temps compartit" (*time sharing)*. Encara que donin la impressió d'estar executant-se simultàniament, en realitat el control de la CPU es reparteix en petits períodes de temps (fraccions de segon) entre els diferents programes, els quals s'alternen segons una determinada política de planificació governada pel sistema operatiu. En un instant de temps, només hi ha un programa en execució, al qual anomenem *procés actiu*. Cada cop que es canvia el procés actiu, el sistema operatiu ha de salvar en memòria l'estat del procés que deixa d'estar actiu i restaurar l'estat del procés que passa a ser actiu. El conjunt de dades que formen l'estat de cada procés rep el nom de *context*, i inclou el valor de tots els registres de propòsit general, el PC i altres estructures de dades.

En un sistema amb memòria virtual, encara que seria possible mantenir una única taula de pàgines en una posició fixa de memòria, i anar-la reescrivint en cada canvi de context amb les traduccions pròpies dels successius processos actius, aquest procediment faria el canvi de context massa costós. Una solució més efectiva i generalment adoptada consisteix que cada cada procés tingui la seva pròpia taula de pàgines en memòria. Llavors, per saber on està la taula de pàgines del procés actiu, el hardware inclou un registre que apunta a la posició inicial de la seva taula de pàgines. S'anomena *registre de taula de pàgines*, i forma part del context del procés actiu.

5.4

3. Fallada de pàgina

En un sistema de memòria virtual, direm que es produeix una *fallada de pàgina* (en anglès "page fault") quan la CPU sol·licita a la MMU una operació de lectura o escriptura sobre una adreça lògica pertanyent a una pàgina que no es troba a la memòria física. Això succeeix quan el bit P de l'entrada de la taula de pàgines corresponent al VPN sol·licitat val 0. En aquest cas, el sistema operatiu verifica en primer lloc que la pàgina en qüestió pertanyi al rang d'adreces vàlides del procés[3], i si és així inicia una operació de lectura del disc[4] per tal de llegir la pàgina[5], la carrega en un marc de pàgina de la memòria física, actualitza la informació de la taula de pàgines i finalment reintenta l'operació reexecutant la instrucció que ha causat la fallada. Passem per alt, de moment, la manera que té el sistema operatiu de localitzar la pàgina al disc. Donat que l'accés a un disc magnètic és desenes de milers de vegades més lent que a la memòria física, mai llegirem o escriurem dades individuals directament al disc, sinó que sempre carregarem primer la pàgina corresponent a la memòria física. Per tant, pel que fa a les escriptures, podem dir que seguirem una política d'escriptura diferida amb assignació, tal com s'explica al Tema 6.

3.1 Reemplaçament de pàgina

Quan el sistema operatiu necessita carregar una pàgina a memòria i no queda cap marc de pàgina lliure, reemplaça una altra pàgina de memòria[6]. Per fer-ho, empra algun *algorisme de reemplaçament de pàgines*, com per exemple LRU (de l'anglès "least recently used", menys utilitzada recentment)[7]. La possibilitat que una pàgina sigui reemplaçada, i el fet que s'usi una política d'escriptura diferida, obliga a incloure a la taula de pàgines un bit D de "pàgina modificada" ("Dirty bit" en anglés). Quan s'ha de reemplaçar una pàgina amb bit D = 1, abans de llegir la nova pàgina cal escriure al disc la pàgina modificada que hi ha a la memòria[8], de manera anàloga al funcionament de les memòries cache amb escriptura diferida. En els sistemes Unix o derivats, a la zona del disc on s'emmagatzemen les pàgines d'un procés que s'han reemplaçat se l'anomena *espai d'intercanvi* (swap space en anglès).

---

3. El bit P val 0 també per a pàgines invàlides, és a dir les que estan en rangs d'adreces no reservats al programa. Els rangs vàlids estan registrats pel sistema operatiu dins el context del procés i inclouen el codi, les dades i un cert espai per a la pila. Si el sistema comprova que l'adreça és invàlida, avorta l'execució del programa, generalment amb un missatge "segmentation fault".
4. Aquí fem referència a fallades de pàgines *majors* (major page faults), però als sistemes de memòria virtual reals també existeixen les fallades de pàgina *menors* (minor page faults), que no requereixen accedir al disc.
5. Alguns sistemes reals no es limiten a llegir del disc la pàgina requerida sinó que aprofiten per llegir-ne algunes més. Aquesta tècnica s'anomena *anticipatory paging*.
6. La selecció de la pàgina a reemplaçar pot fer-se entre totes les pàgines físiques (*global page replacement*, utilitzat per Linux) o només entre un conjunt de pàgines assignades al programa (*local plage replacement)*.
7. Existeixen moltes variants de l'LRU, requerint moltes d'elles mantenir un *reference bit* associat a cada pàgina. Nosaltres no tindrem en compte aquest bit en els nostres exemples.
8. En algunes implementacions reals, s'utilitza la tècnica de *precleaning*, consistent en anticipar l'escriptura al disc de pàgines marcades com a modificades.

5.4

4. Traducció ràpida amb TLB

4.1 El TLB, una cache de traduccions

Tal i com hem vist, en un sistema amb memòria virtual cada vegada que necessitem llegir o escriure una dada a la memòria caldrà accedir abans a la taula de pàgines per traduir-ne l'adreça. Donat que la taula de pàgines està en memòria, serà necessari fer dos accessos a memòria per cada dada que vulguem accedir, un per traduir l'adreça i un per llegir o escriure la dada. Per mitigar aquest inconvenient, els processadors moderns incorporen una cache de traduccions, anomenada TLB (de l'anglès translation-lookaside buffer) que emmagatzema les darreres entrades utilitzades de la taula de pàgines. El TLB forma part de la MMU, i per tant és un component hardware. Existeixen diferents tipus de TLB i diferents maneres d'utilitzar-los. D'ara en endavant, ens referirem sempre a les característiques d'un TLB simplificat, inspirat en el de l'arquitectura MIPS. La Figura 7.7 mostra l'estat d'un TLB de quatre entrades en un moment donat.

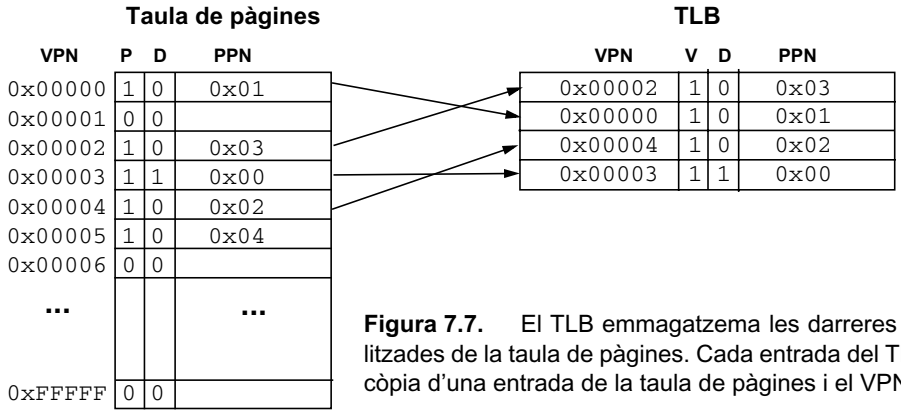

Figura 7.7. El TLB emmagatzema les darreres entrades utilitzades de la taula de pàgines. Cada entrada del TLB conté una còpia d'una entrada de la taula de pàgines i el VPN d'aquesta.

Cada entrada del TLB conté una còpia d'una entrada de la taula de pàgines (bits P, D i PPN). A més a més, cal emmagatzemar a cada entrada del TLB el VPN de l'entrada de la taula de pàgines a la que correspon. Per motius que s'explicaran més endavant, al bit del TLB que emmagatzema el bit P (Presència) de la taula de pàgines l'anomenarem bit V (Validesa).

4.2 Encert de TLB

Quan calgui traduir una adreça, es buscarà la seva traducció en el TLB. Com en el cas de les caches, aquesta búsqueda requereix consultar una sola entrada, totes les entrades o un subconjunt solament, depenent de l'*algorisme d'emplaçament*. L'algorisme més comunament usat pel TLB, i que assumirem nosaltres per defecte en endavant, és el de correspondència totalment associativa. Per tant, es buscarà el VPN de l'adreça entre tots els VPNs de les entrades del TLB. Si es troba una entrada amb aquest VPN, i independentment del valor dels altres camps (fins i tot si V val 0), direm que s'ha produït un *encert de TLB* (TLB hit). Quan això succeeix, passarem a utilitzar la informació continguda a l'entrada trobada. Si el bit V de la traducció obtinguda del TLB val 0, després de l'encert de TLB es produirà una fallada de pàgina. La gestió de la fallada de pàgina es realitzarà tal i com s'ha explicat anteriorment i, un cop resolta i actualitzada l'entrada corresponent de la taula de pàgines, es reescriurà l'entrada al TLB amb els nous valors.

4.3 Fallada de TLB

Quan calgui traduir una adreça i el VPN d'aquesta no es trobi entre els VPNs de les entrades del TLB, direm que s'ha produït una *fallada de TLB* (TLB miss). Les accions necessàries per resoldre-la es poden executar per software, a càrrec de la CPU (com en el cas de MIPS), o bé per hardware (a càrrec de la MMU). Per resoldre la fallada de TLB caldrà llegir l'entrada corresponent de la taula de pàgines i copiar-la a una entrada del TLB. Si hi ha alguna entrada del TLB lliure (i.e. amb el bit V a 0, més endavant s'explicarà amb més detall l'ús d'aquest bit per indicar que una entrada no està inicialitzada), es reescriurà aquesta entrada. En cas que no hi hagi cap entrada lliure, caldrà reemplaçar-ne una seguint un *algorisme de reemplaçament* de forma anàloga a les caches (LRU, aleatori, FIFO, etc.). Un cop resolta la fallada, cal reintentar l'accés a memòria que l'ha causat. El contingut de l'entrada de la taula de pàgines que s'ha copiat no s'analitza durant la fallada de TLB, i si el bit P val 0, el bit V de l'entrada del TLB quedarà a 0, provocant una fallada de pàgina quan es reintenti l'accés. Per tant, el bit V d'una entrada del TLB pot valer 0 per dos motius: perquè l'entrada no ha estat inicialitzada o perquè l'entrada acaba de ser copiada de la taula de pàgines i s'està resolent la fallada de pàgina corresponent.

4.4 Consistència del bit D del TLB amb el de la taula de pàgines

De tots els camps de bits que conté una entrada del TLB tan sols un és susceptible de ser modificat per l'execució del programa: el bit D, que es posa a 1 quan el programa fa una escriptura (instrucció de store). Com que el TLB és de fet una cache, cal preguntar-se quina política d'escriptura ha de seguir respecte d'aquest bit, a fi que sigui consistent amb el corresponent bit D de la taula de pàgines. El mètode adoptat per MIPS, i que nosaltres assumirem per defecte, és que el bit D del TLB s'escriu a la taula de pàgines seguint una política d'*escriptura immediata*.[9] És a dir, que en cas d'un accés a memòria per escriptura, el bit D es posa a 1 simultàniament en el TLB i a la taula de pàgines (i per tant ens caldrà accedir a la memòria). No obstant, això no suposa accedir a la taula de pàgines per a totes les escriptures sinó tan sols quan el bit D valgui 0, és a dir la primera vegada que s'escrigui en una pàgina.

4.5 Inicialització del TLB i bit V

Inicialment, abans que s'hagi realitzat cap traducció d'adreces, donarem al bit V de totes les entrades del TLB el valor 0 (d'aquí el seu nom, bit de validesa). Això ens permetrà saber si hi ha entrades lliures durant la resolució d'una fallada del TLB. Per tant, aquest bit jugarà dos papers diferents en funció de la situació. En la resolució d'una fallada de TLB el bit V ens indicarà si una entrada està lliure. Després d'un encert de TLB el bit V ens dirà quin és el valor del bit P de l'entrada corresponent de la taula de pàgines. Donat que el valor del camp VPN d'una entrada no inicialitzada és desconegut, podria succeir que la cerca d'un VPN provoqués, per casualitat, un encert de TLB en una entrada del TLB no inicialitzada. Donat que el valor del seu bit V serà 0, això no serà un problema ja que la conseqüent fallada de pàgina provocarà la inicialització correcta de l'entrada.

9. Una altra possibilitat seria seguir una política d'escriptura diferida, fent que el bit D de la taula de pàgines no s'actualitzi fins que l'entrada del TLB sigui reemplaçada.

4.6 Flux del procés de traducció d'adreces amb TLB

La Figura 7.8 resumeix el flux d'accions a dur a terme en el procés de traducció d'una adreça en un sistema amb memòria virtual com l'explicat en aquest tema. Es mostren 3 possibles incidències durant la traducció: fallada de TLB, fallada de pàgina i actualització del bit D (pàgina modificada). La fallada de pàgina sempre causa una excepció, la qual cancel·la l'execució de la instrucció i invoca el sistema operatiu perquè la gestioni (per software). Un cop resolta, es retorna al programa i es reexecuta la instrucció que l'ha causat. Les fallades de TLB i l'actualització del bit D de la TP (pàgina modificada) els gestiona la MMU en alguns processadors, de forma anàloga a les fallades de cache, és a dir que no es cancel·la l'execució de la instrucció sinó que la traducció continua un cop resolta la incidència (fletxa [1] del diagrama). En el MIPS en canvi, aquests events causen una excepció, es cancel·la l'execució de la instrucció, s'invoca el sistema operatiu per gestionar la incidència (per software), i un cop resolta es reexecuta la instrucció causant, al igual que per a les fallades de pàgina (fletxa [2] del diagrama).

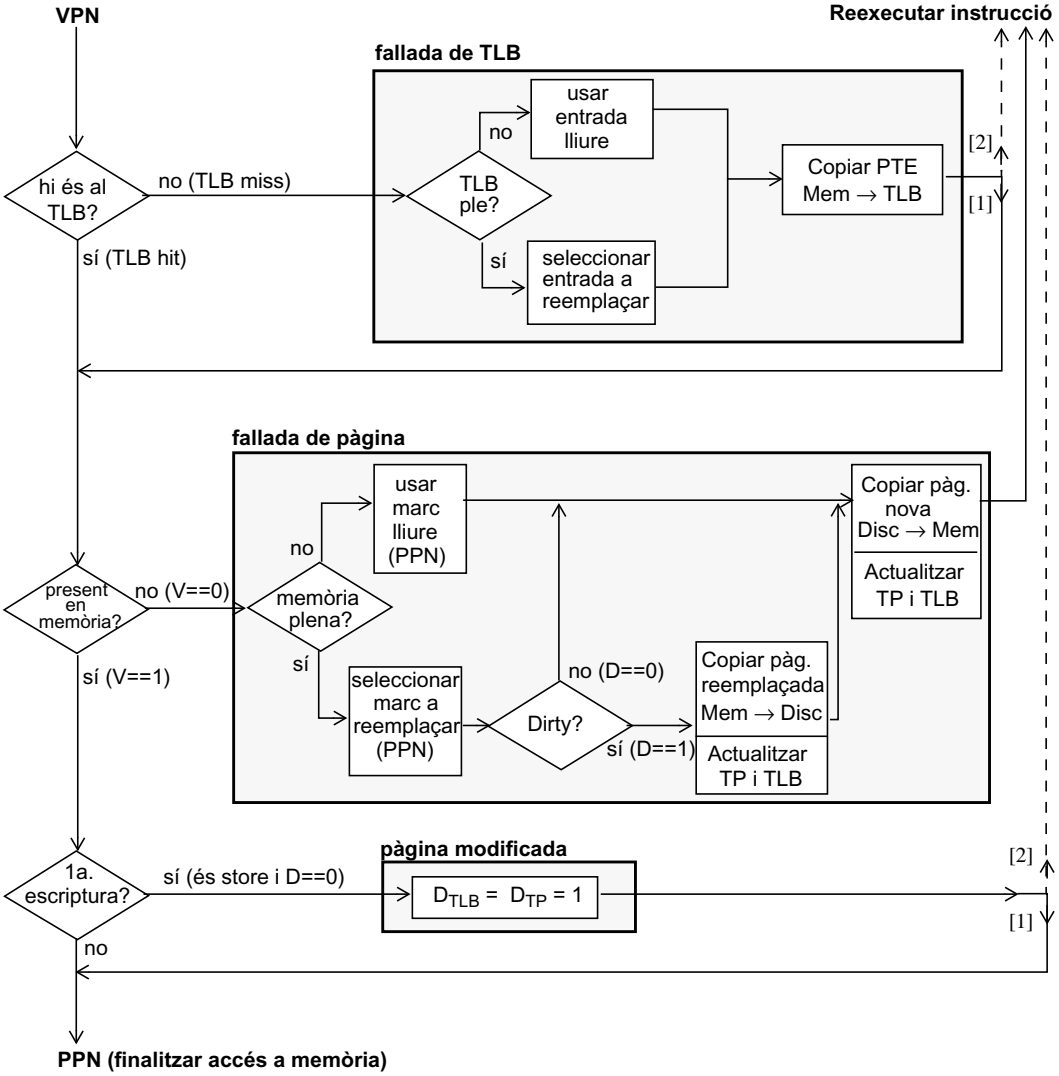

Figura 7.8. Flux del processament de la traducció d'una adreça. [1] Gestió per hardware (MMU). [2] Gestió per software (excepció)

5. Protecció i Compartició

5.1 Protecció amb memòria virtual

Probablement, la funció més important de la memòria virtual sigui la de permetre compartir la memòria del computador de manera segura entre múltiples processos. Per satisfer aquest requeriment, cal un mecanisme de protecció que garanteixi que cap procés, de manera intencionada o accidental, pugui llegir o escriure dades de l'espai d'adreçament d'un altre procés o del sistema operatiu. Fins ara, hem assumit que una pàgina física només pot estar assignada a un procés en un moment donat, i.e. no pot aparèixer a la taula de pàgines de més d'un procés. Si es satisfà aquesta premissa, i.e. si els processos no comparteixen cap pàgina física, el mecanisme de traducció d'adreces fa impossible que puguin accedir a un espai d'adreces que no sigui el seu. No obstant, un procés podria intentar eludir aquesta restricció modificant la seva pròpia taula de pàgines. Per evitar-ho, ubicarem les taules de pàgines a l'espai d'adreçament reservat al sistema operatiu[10].

El sistema operatiu no és un procés, sinó un programari comú a tots els processos que s'executa en determinades circumstàncies (per exemple quan es produeix una fallada de pàgina). No disposa d'un espai d'adreçament com el que té un procés, sinó que té reservada una part de l'espai d'adreçament de tots els processos (en MIPS, les adreces lògiques amb el bit 31=1). Per permetre que el sistema operatiu tingui privilegis que no tenen els programes d'usuari (per exemple el de modificar el TLB o la taula de pàgines), el processador disposa de dos modes de funcionament, mode usuari i mode sistema. Només quan el processador està en mode sistema serà possible modificar el TLB o les taules de pàgines dels processos. D'aquesta manera es garanteix el mecanisme de protecció. Al Tema 8 s'expliquen més detalls sobre els dos modes d'execució i sobre la manera en que s'executa el sistema operatiu.

5.2 Protecció contra escriptura

Algunes vegades resulta convenient prohibir l'escriptura en determinades pàgines i permetre-ho en altres. Això és possible per mitjà d'un un bit de permís d'escriptura (E) que s'inclou en cada entrada de la taula de pàgines i del TLB. Si una instrucció intenta escriure en una pàgina de sols lectura (E=0) el sistema operatiu avortarà l'execució del programa indicant un intent de violar la protecció.

5.3 Compartició de memòria entre processos

Hi pot haver situacions en què un procés (e.g. P1) vulgui permetre a un altre (e.g. P2) accedir al seu espai d'adreçament. El sistema operatiu, a petició de P1, pot habilitar la compartició simplement escrivint una entrada a la taula de pàgines del procés P2 que assigni una pàgina lògica de P2 a la mateixa pàgina física que P1 vol compartir. A petició del procés P1, la pàgina compartida pot tenir permís d'escriptura o no per al procés P2, sols cal que el sistema operatiu inicialitzi adecuadament el bit E de la pàgina lògica de P2.

La Figura 7.9 mostra un exemple amb les taules de pàgines de P1 i P2, després que els dos processos comparteixin la mateixa pàgina física (PPN 2), amb accés per a escrip-

10. Quan parlem del sistema operatiu ens referim al nucli del sistema operatiu (kernel en anglès)

tura tots dos. P1 la té assignada a VPN 0, i P2 la té assignada a VPN 1. La Figura mostra el moment en què P2 està en execució, i per tant el registre de Taula de pàgines apunta a la taula de P2, i el TLB conté sols traduccions de la taula de P2.

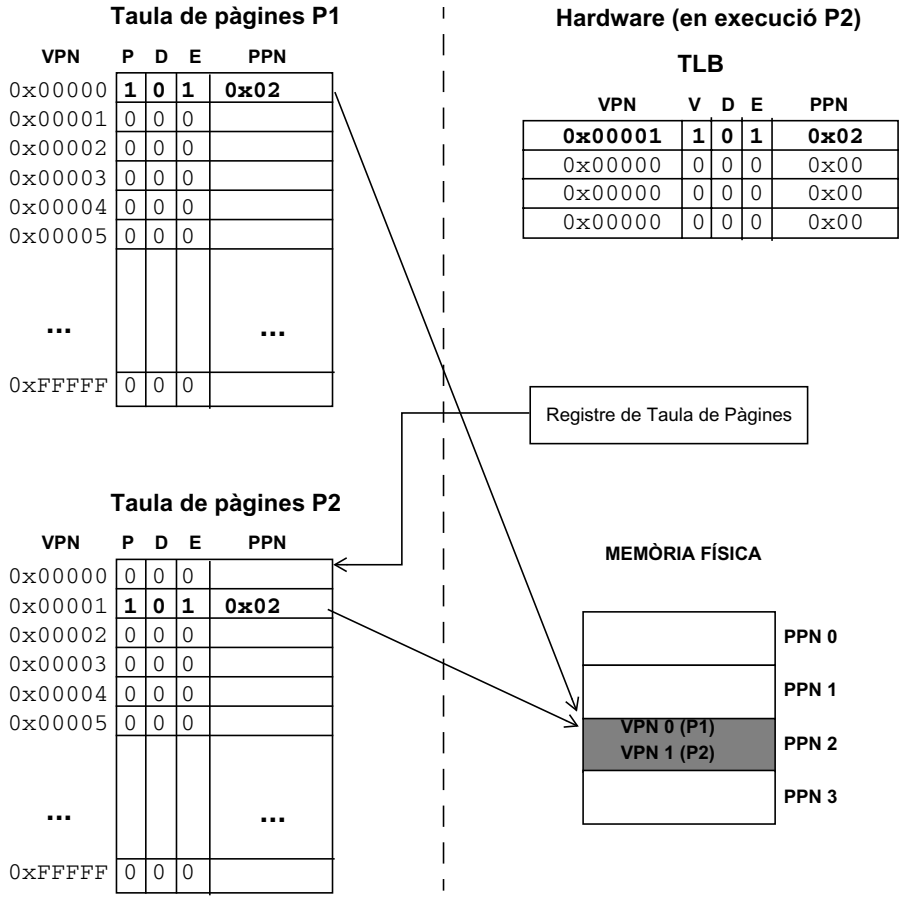

Figura 7.9. Exemple de compartició en mode escriptura. El procés P1 sol·licita al sistema operatiu compartir la página lógica VPN 0 amb el procés P2 en mode escriptura (E=1). El sistema operatiu escriu una entrada a la taula de pàgines de P2 on s'assigna la página lógica VPN 1 a la mateixa página física que P1 vol compartir (PPN 2)

La compartició de pàgines físiques té una aplicació òbvia quan es vol que diversos processos col·laborin intercanviant-se dades. Però també és molt freqüent que el sistema operatiu ho faci amb el codi de les anomenades biblioteques compartides (*shared libraries*) o d'alguns programes usats simultàniament per multiples usuaris (un editor, per exemple). Si múltiples processos concurrents usen una mateixa pàgina de codi simultàniament, en comptes d'ocupar múltiples pàgines físiques en memòria, una per a cada procés, el sistema pot estalviar molta memòria si en manté una sola còpia en memòria física, i fa que cada procés que la usarà tingui assignada una pàgina lògica a aquesta mateixa pàgina física. Naturalment, això requereix que tots els processos tinguin aquesta pàgina lògica marcada com de sols lectura (E=0).

5.4

6. Integració del TLB i la memòria cache

En un sistema amb memòria virtual, els accessos a la cache es poden organitzar de diverses maneres. L'esquema més senzill d'entendre és la *cache indexada físicament*, i consisteix a fer la traducció amb TLB en primer lloc, i a continuació (si no s'ha produït ni fallada de TLB ni fallada de pàgina) accedir a la cache amb l'adreça física (Figura 7.10).

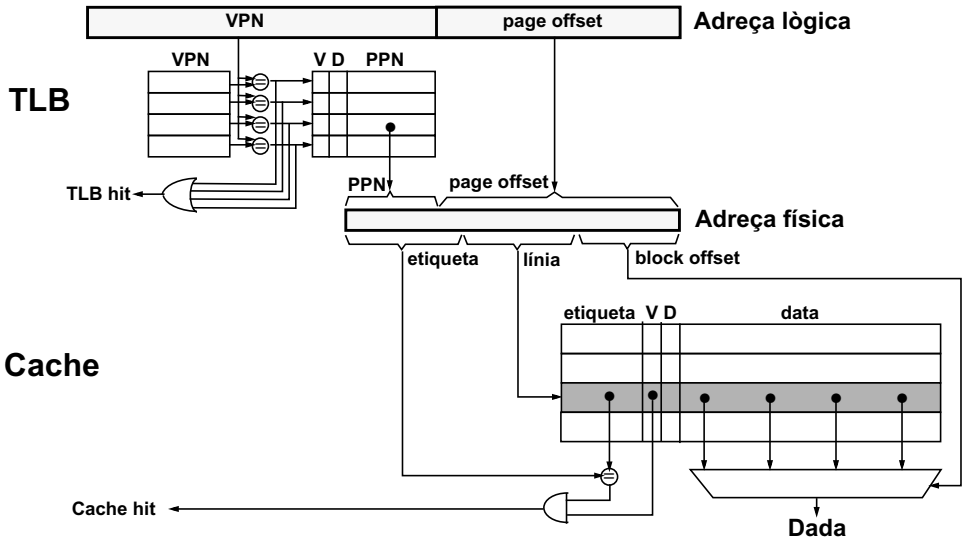

Figura 7.10. Funcionament combinat del TLB i una cache indexada físicament.

Aquest esquema és simple però té un temps d'encert massa llarg, ja que accedeix seqüencialment al TLB i a la cache. La *cache indexada virtualment* (Figura 7.11 esq.) té un temps més curt, però sofreix un problema d'aliasing[11]. La *cache indexada virtualment i etiquetada fisicament* (Figura 7.11 dreta) és una altra alternativa àmpliament adoptada i que permet fer l'accés al TLB i a la cache en paral·lel sense causar aliasing[12].

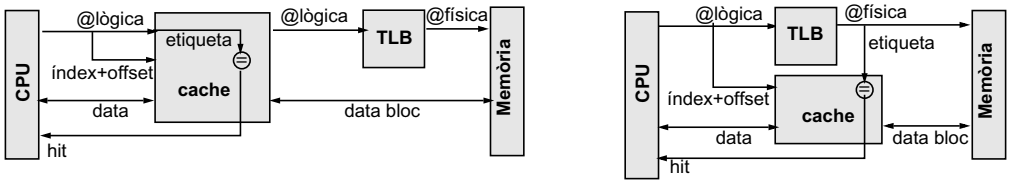

Figura 7.11. Cache indexada virtualment (esquerra). Cache indexada virtualment i etiquetada físicament (dreta).

11. La cache indexada virtualment s'accedeix usant l'adreça virtual. Si en resulta un encert, llavors no cal fer cap accés al TLB i s'escurça el temps d'accés. Al contrari, la traducció d'adreces sols és necessària en cas de fallada de cache, per tal d'accedir a l'adreça física correcta. Aquest mètode té un defecte, l'aliasing: si dues o més pàgines lògiques es refereixen a la mateixa pàgina física (veure apartat 5.3), llavors ambdues poden ocupar blocs diferents de cache i allò que escrivim en una pàgina lògica pot no ser consistent amb allò que llegim en l'altra pàgina lògica, tot i ser còpies de la mateix pàgina física.

12. En una cache indexada virtualment i etiquetada físicament, les etiquetes que s'hi guarden són els bits alts de les adreces físiques. Si la mida de la pàgina ens assegura que l'índex de la cache usi solament bits de l'offset de pàgina (i no del VPN), llavors l'indexat no necessita esperar a la traducció, sinó que pot iniciar-se simultàniament, llegint de la cache les etiquetes i blocs de dades corresponents. Un cop llegides les etiquetes, si la traducció amb el TLB ja ha acabat, disposarem dels bits alts de l'adreça física (PPN) amb els quals comparar les etiquetes llegides per saber si és un encert o no.

7. Exemple pràctic

Suposem un sistema computador amb les següents característiques:

- Mida de pàgina de 4KB = 2^{12} bytes.
- Espai d'adreçament lògic: 4GB = 2^{32} bytes = 2^{20} pàgines virtuals ($2^{32}/2^{12}$).
- Memòria física: 16KB = 2^{14} bytes = 4 marcs de pàgina ($2^{14}/2^{12}$).
- TLB de 4 entrades, associatiu, amb reemplaçament LRU.

L'exemple de la Figura 7.12 mostra l'estat inicial de la taula de pàgines i del TLB d'un únic programa en execució en un moment donat. Sabem, a més a més, que sols 3 dels 4 marcs de pàgina estan ocupats, el quart (PPN=3) està lliure. Per altra banda, a fi de gestionar el reemplaçament LRU del TLB i de les pàgines en memòria, sabem que els 3 darrers accessos han estat a les pàgines VPN=3, VPN=0 i VPN=2, per aquest ordre.

A partir d'aquest estat inicial, suposem que s'executa una seqüència de cinc accessos a memòria. Una taula descriu el que succeeix en cada accés: Lectura o Escriptura, Adreça lògica, VPN, si hi ha fallada de TLB, quina entrada (VPN) s'ha reemplaçat al TLB, si s'ha produït una fallada de pàgina i la consegüent lectura de la pàgina al disc, quina pàgina (VPN) s'ha escrit al disc (en cas que se n'hagi hagut d'escriure alguna), i quin ha estat el PPN resultant de la traducció.

Taula de pàgines (a l'inici)

VPN	P	D	PPN
0x00000	1	0	0x01
0x00001	0	0	
0x00002	1	0	0x02
0x00003	1	1	0x00
0x00004	0	0	
0x00005	0	0	
0x00006	0	0	
...	0	0	...
0xFFFFF	0	0	

TLB (a l'inici)

VPN	V	D	PPN
0x00003	1	1	0x00
0x00000	1	0	0x01
0x00002	1	0	0x02
0x0AFB3	0	0	0xFF

Seqüència d'accessos a memòria

L/E	Adreça lògica	VPN	fallo TLB?	VPN reemplaçada al TLB	fallo pàg?	lect. disc?	VPN pàg. escrita al disc	PPN
E	0x00002A0B	0x00002	no		no	no		0x02
L	0x00001F21	0x00001	sí		sí	sí		0x03
L	0x0000420C	0x00004	sí	0x00003	sí	sí	0x00003	0x00
L	0x00003001	0x00003	sí	0x00000	sí	sí		0x01
L	0x00005120	0x00005	sí	0x00002	sí	sí	0x00002	0x02

Figura 7.12. Exemple pràctic. Estat inicial de la taula de pàgines i TLB, i evolució subsegüent per a una seqüència de 5 accessos a memòria..