



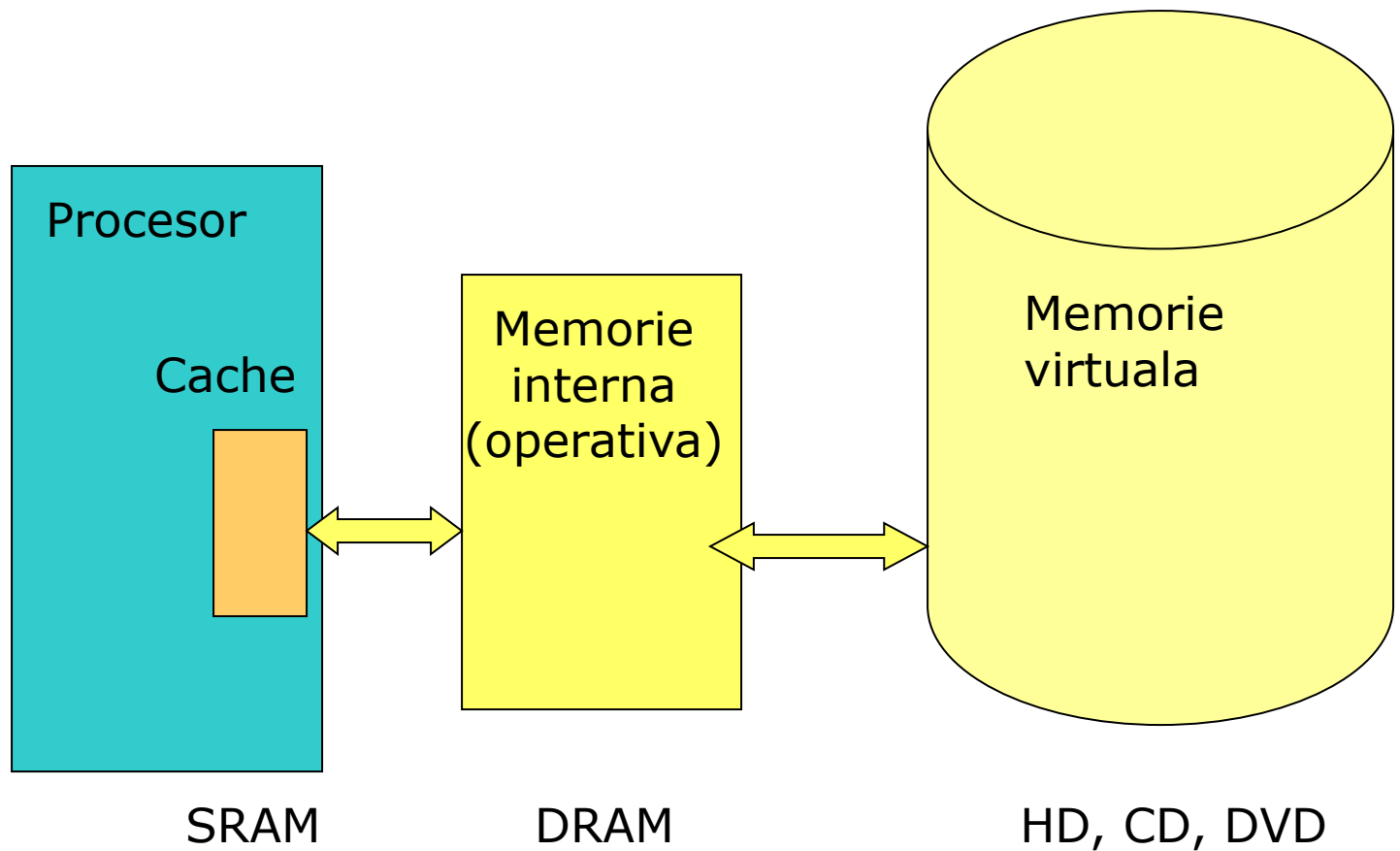
Sisteme cu microprocesoare

Cursul 7 Ierarhii de memorii

Parametrii de performanta ai unei memorii

	SRAM	DRAM	HD, CD
Capacitate	Mica 1-64ko	Medie 256-2Go	Mare 20-160Go
Timp de acce	Mic 1-10ns	Mediu 15-70ns	Mare 1-10ms
Cost	mare	mediu	mic

Ierarhie de memorii



Principii in favoarea ierarhizarii memoriei

- **Localitate temporală** – probabilitatea este mare ca o locație accesată să fie solicitată și în viitorul apropiat
- **Localitate spațială** – probabilitatea este mare ca și vecinii locației accesate să fie solicitate în viitorul apropiat
- **90/10** – 90% din timp se execută 10% din program
- **ideea de bază:** de a aduce mai aproape de procesor acele zone de memorie care au o probabilitate mai mare de a fi utilizate în viitorul apropiat



Memoria cache

- Memorie foarte rapida, de capacitate mica
- Memoria cea mai apropiata de procesor
- Organizare: linii de memorie cache
- Pastreaza copii ale unor zone (linii) din memoria interna
- Memoria cache nu este vizibila programatorului
- Transferul intre memoria cache si memoria interna se face automat sub controlul Unitatii de management a memoriei (MMU)



Parametrii tipici ai unei memorii cache

Parametru	Valoare
Dimensiune memorie	32kocteți-2Mocet
Dimensiune linie cache	16-256 ocetți
Timp de acces	0.5-10 ns
Viteza de transfer (lățime de bandă)	800-5000Mocetți/sec.
Tip circuit	RAM-ul intern al procesorului sau RAM static extern

Proiectarea memoriilor cache

- Probleme de proiectare:
 1. Care este dimensiunea optimă a unei linii cache ?
 2. Unde se amplasează o nouă linie cache ?
 3. Cum se regăsește informația conținută în memoria cache ?
 4. Care linie se înlocuiește în cazul în care memoria este plină și se solicită transferul unei noi linii ?
 5. Cum se rezolvă operațiile de scriere ?
- Arhitecturi de memorii cache:
 - memorii cache cu mapare directă
 - memorii cache asociative
 - memorii cache set asociative
 - memorii cache organizate pe sectoare

Memorii cache cu mapare directă

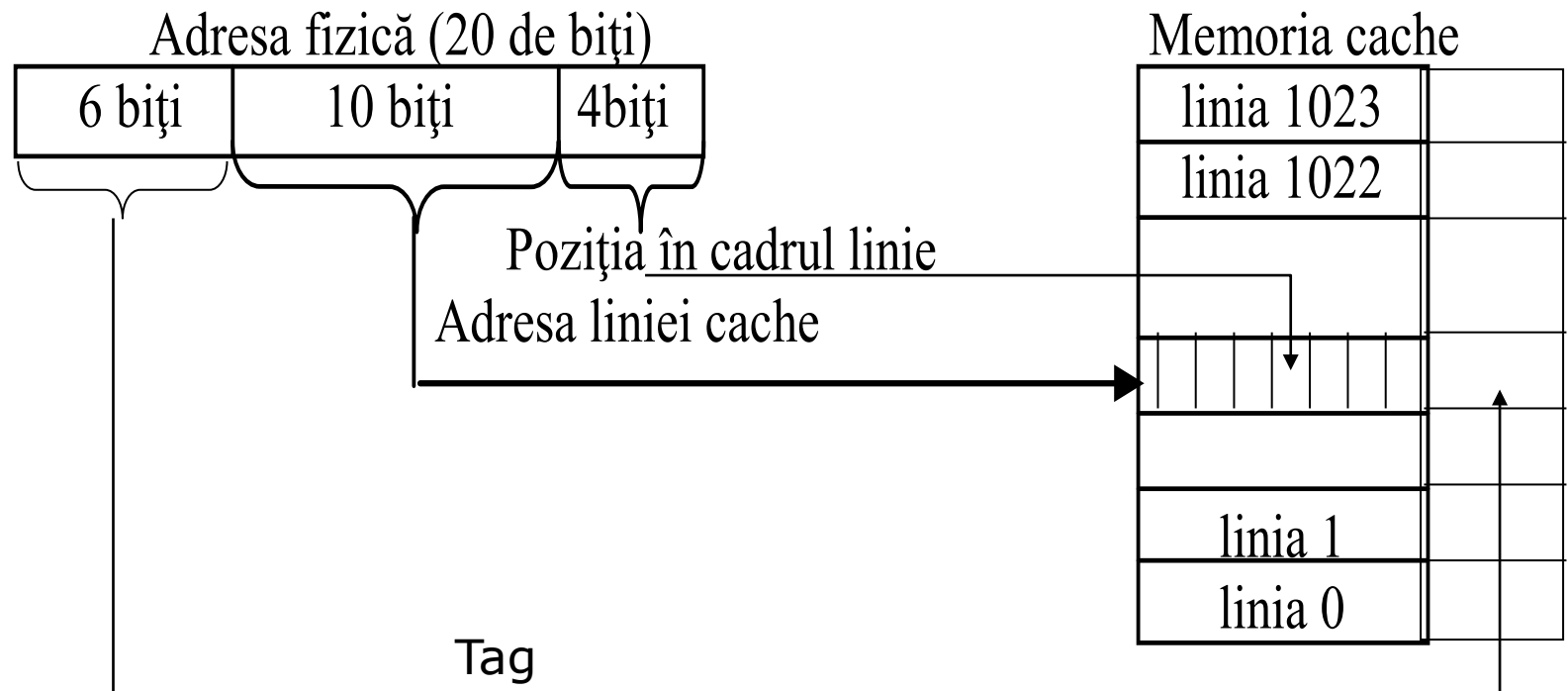


Figura 9-9 Memoria cache - Maparea directă

Memorii cache cu mapare directă

- Principiu: adresa unei linii in memoria cache este determinata direct (un camp) din adresa fizica a liniei – “direct mapping”
 - tag-ul este folosit pentru a face diferenta intre linii care in principiu ocupa aceeasi pozitie in memoria cache
- Avantaje:
 - usor de implementat
 - usor de a plasa, de a regasi si de a inlocui o linie
- Dezavantaje:
 - in anumite cazuri doua linii se inlocuiesc repetat, chiar daca memoria cache nu este plina
 - utilizare ineficienta a spatiului de memorie cache

Memoria cache asociativă

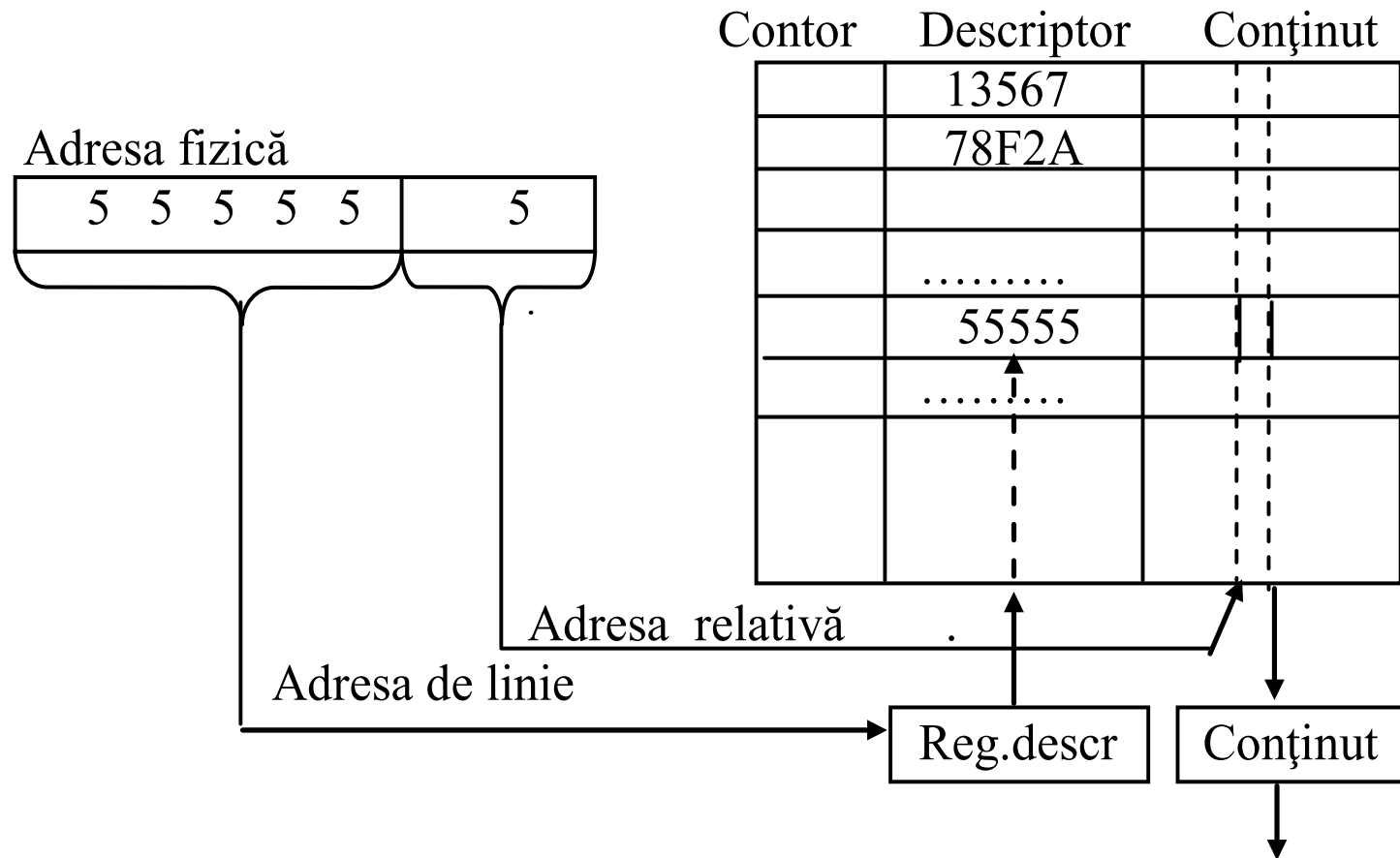


Figura 9-10 Memoria cache asociativa

Memoria cache asociativă

- Principiu:
 - o linie este plasata in orice zona libera (cadru de linie gol) din memoriei cache
 - o locatie se regaseste in cache prin compararea campului propriu de descriptor cu descriptorii liniilor pastrate in cache
 - comparatie hardware – necesita foarte multe circuite de comparatie (un circuit/linie)
 - comparatie secventiala – necesita mult timp pentru comparatii repetate
- avantaje:
 - utilizare eficienta a spatiului cache
- Dezavantaje:
 - numar limitat de linii cache si implicit capacitate redusa a memoriei cache – datorita operatiei de comparare

Memoria cache set asociativa

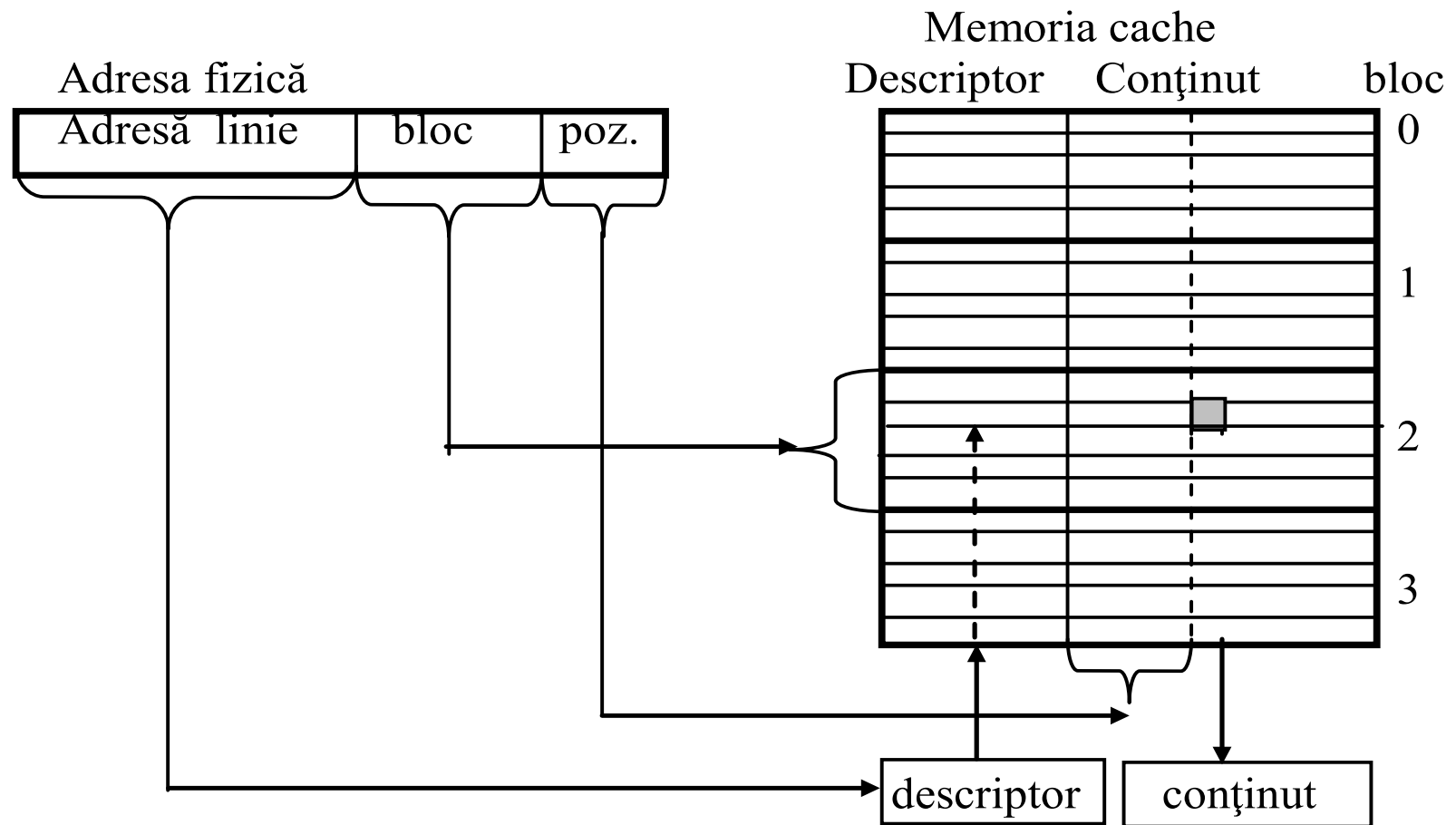


Figura 9-11 Memoria cache set asociativa



Memoria cache set asociativa

- Principiu: combinarea celor 2 metode anterioare, asociativa si cu mapare directa
 - mai multe linii organizate in blocuri
 - blocurile identificate prin mapare directa
 - liniile (in cadrul blocului) identificate prin metoda asociativa
- Avantaje:
 - combina avantajele metodelor anterioare:
 - numar mare de linii, fara limitari de capacitate
 - utilizarea eficienta a capacitatii memoriei cache
- Dezavantaj:
 - implementare mai complexa

Memoria cache organizată pe sectoare

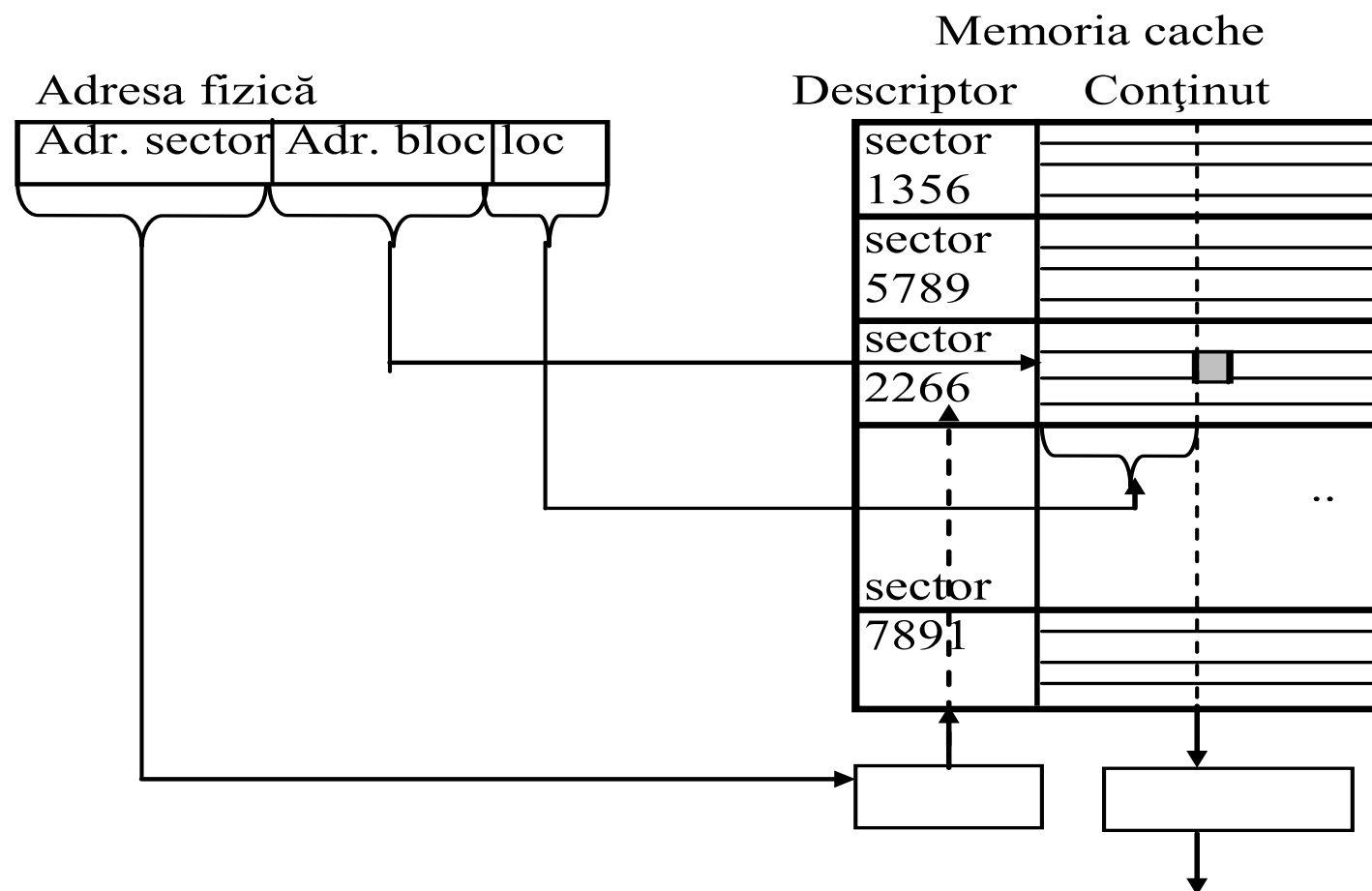


Figura 9-12 Memoria cache organizată pe sectoare

Memoria cache organizată pe sectoare

- Principiu: similar cu metoda set asociativa, dar ordinea este schimbata:
 - sectorul (blocul) este identificat prin metoda asociativa, iar linia in cadrul sectorului prin metoda maparii directe
- Avantaje si dezavantaje: similare cu metoda anterioara

Scrierea in memoria cache

- Problema cu operatia de scriere: apare o inconsistenta intre memoria interna si copia din memoria cache daca scrierea se face numai in cache
- 2 metode posibile:
 - **Write back** – scriere numai in memoria cache si actualizare in memoria interna numai la descarcarea liniei cache
 - Avantaj: eficienta maxima
 - Dezavantaj: inconsistenta temporara intre cache si memoria interna, critica in cazul sistemelor multi-master (multi-procesor)
 - **Write through** – scriere atat in memoria cache cat si in memoria interna
 - Avantaj: nu exista inconsistenta
 - Dezavantaj: operatiile de scriere se fac la viteza (mai mica) a memoriei interne
 - dar operatiile de scriere sunt mai rare decat cele de citire (in raport de 1 la 10)

Eficiența memoriei cache

- $t_a = t_c + (1-R_r) \cdot t_p$
 - unde:
 - t_a – timpul de acces mediu
 - t_p – timpul de acces al memoriei principale
 - t_c – timpul de acces al memoriei cache
 - R_r – rata de succes
 - $(1-R_r)$ – rata de eșec (miss rate)

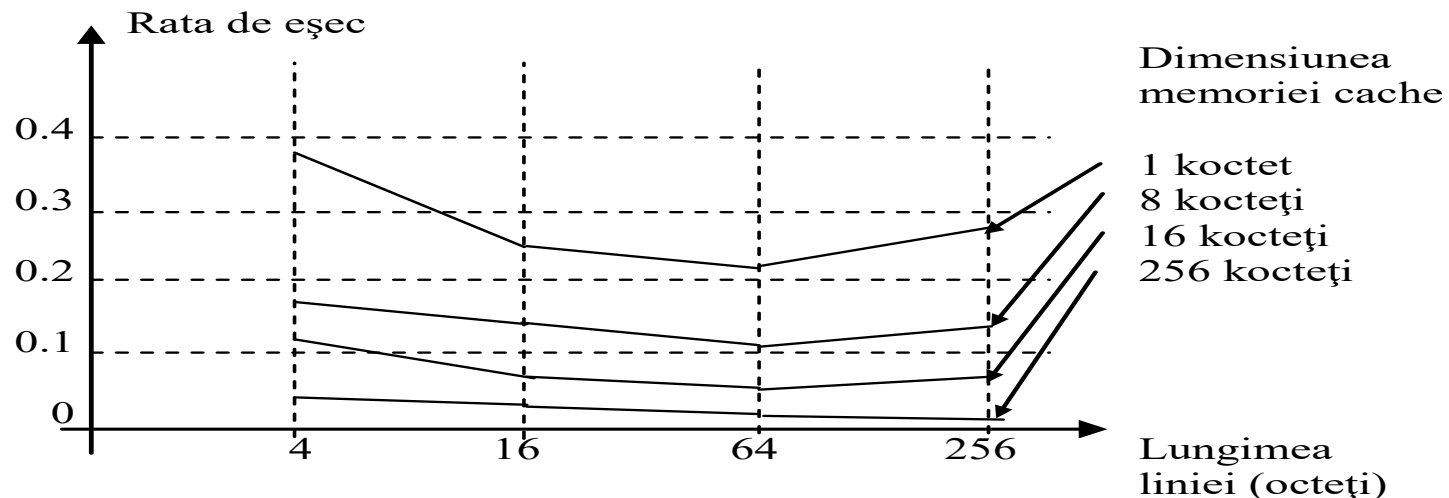


Figura 9-13 Rata de eșec funcție de lungimea liniei cache



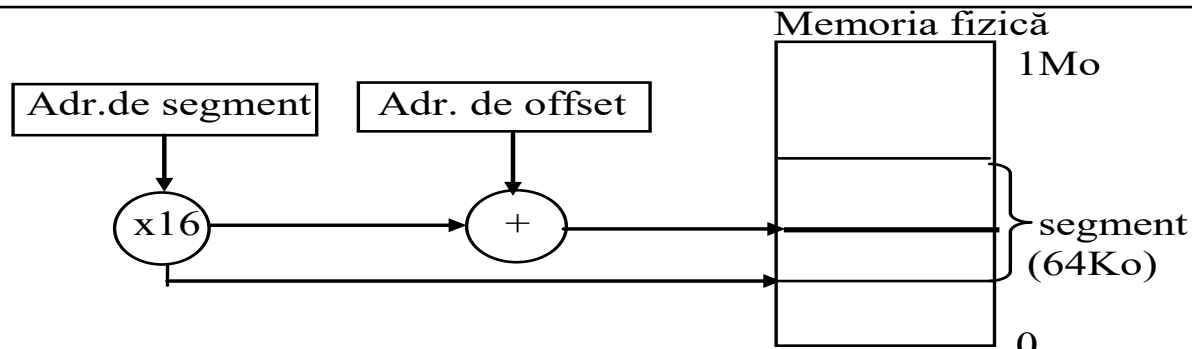
Memoria virtuala

- Obiective:
 - Extinderea memorie interne peste memoria externa
 - Protejarea zonelor de memorie impotriva unor accese neautorizate
- Tehnici de implementare:
 - Paginarea
 - Segmentarea

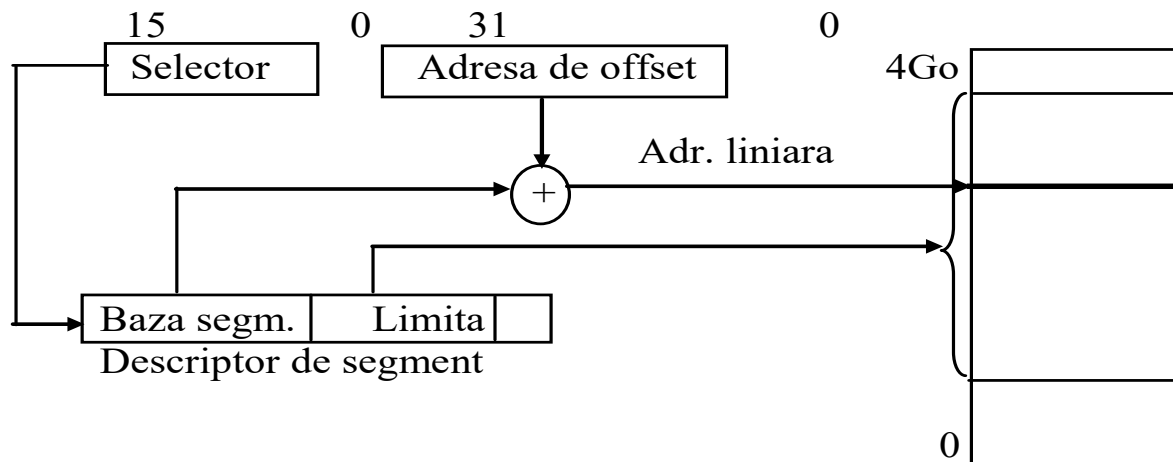
Segmentarea

- Impartirea memoriei in blocuri (segmente)
- Adresarea unei locatii prin:
 - $\text{Adresa de segment} + \text{Adresa de offset} = \text{Adresa fizica}$
- Atasarea de attribute de acces la fiecare segment
- Avantaje:
 - - accesul unui program sau task este limitat numai la locațiile conținute în segmentele care i-au fost alocate
 - -se pot separa zonele de memorie funcție de destinația acestora: cod, date, stivă
 - - specificarea adresei relative în cadrul segmentului necesită un număr mai mic de biți, ceea ce duce implicit la reducerea lungimii instrucțiunilor
 - - segmentele unui program pot fi amplasate în diferite locuri ale spațiului de adresare sau pot fi relocatate, cu modificări minore în codul programului

Segmentarea la procesoarele Intel



Calculul adresei in modul real



Calculul adresei in modul protejat

Segmentarea la procesoarele Intel

- Detalii de implementare a segmentarii in Modul protejat:
 - Selector:
 - contine:
 - Index – pozitia descriptorului de segment intr-o tabela de descriptori
 - TI – table identification bit: GDT sau LDT
 - RPL – requested privilege level – nivelul de privilegiu pe care trebuie sa-l detina un task pentru a accesa segmentul
 - Descriptor de segment:
 - controleaza accesul la segment prin elementele sale componente:
 - adresa segmentului
 - lungimea segmentului
 - drepturile de acces (privilegii)
 - indicatori
 - Tabele de descriptori:
 - General Descriptor Table (GDT) – pastreaza descriptori pentru segmente comune
 - Local Descriptor Tables (LDT) – unul pentru fiecare task; contine descriptori pentru segmente alocate unui singur task
 - Tipuri de descriptori
 - Descriptori pentru segmente de Cod si de Date
 - Descriptori de sistem
 - Descriptori pentru porti de acces – controleaza accesul la functii ale sistemului de operare

Mecanisme de protectie oferite prin segmentare (in cazul procesoarelor Intel)

- Acces la memorie numai prin descriptori pastrati in GDT si LDT
 - GDT pastreaza descriptorii segmentelor accesibile mai multor taskuri
 - LDT pastreaza descriptorii segmentelor alocate unui singur task => segmente protejate
- Operatiile de citire si scriere sunt permise in functie de tipul segmentului (Cod sau date) si starea anumitor indicatori (pastrati in descriptor)
 - pentru segmente de cod: citire instructiuni (in vederea executiei) si eventual citire date
 - pentru segmente de date: citire si eventual scriere
- Nivele de privilegiu:
 - 4 nivele, 0 cel mai privilegiat, 3 cel mai putin privilegiat
 - nivelele 0,1 si 2 alocate pentru sistemul de operare, iar nivelul 3 pentru programele utilizator
 - un task mai putin privilegiat nu poate accesa un segment cu nivel de privilegiu mai mare (ex: un program utilizator nu poate accesa segmente alocate sistemului de operare)



Paginarea

- Impartirea memoriei interne si externe in blocuri (pagini) de lungime fixa
- Incarcarea in memoria interna a paginilor care au probabilitatea cea mai mare de a fi utilizate in viitorul apropiat
- Implementare similara cu memoria cache
- Probleme de proiectare:
 - dimensionarea optimă a paginii
 - amplasarea în memorie a unei noi pagini
 - regăsirea paginilor în memorie
 - selectarea paginilor care se descarcă din memorie
 - implementarea operațiilor de scriere.

Paginarea

– implementare prin tehnica asociativa

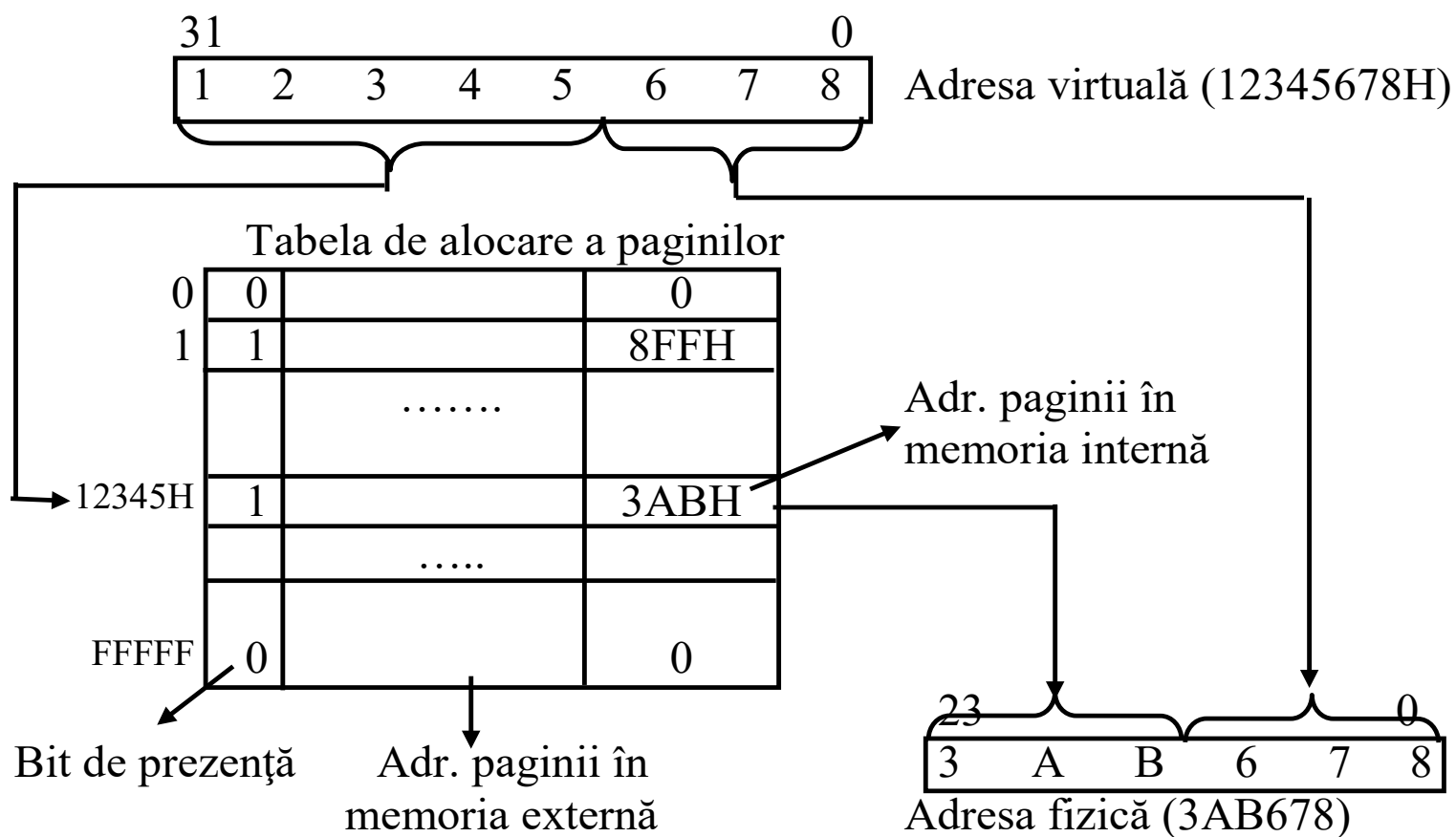


Figura 9-20 Exemplu de implementare a paginării

Paginarea la procesoarele Intel

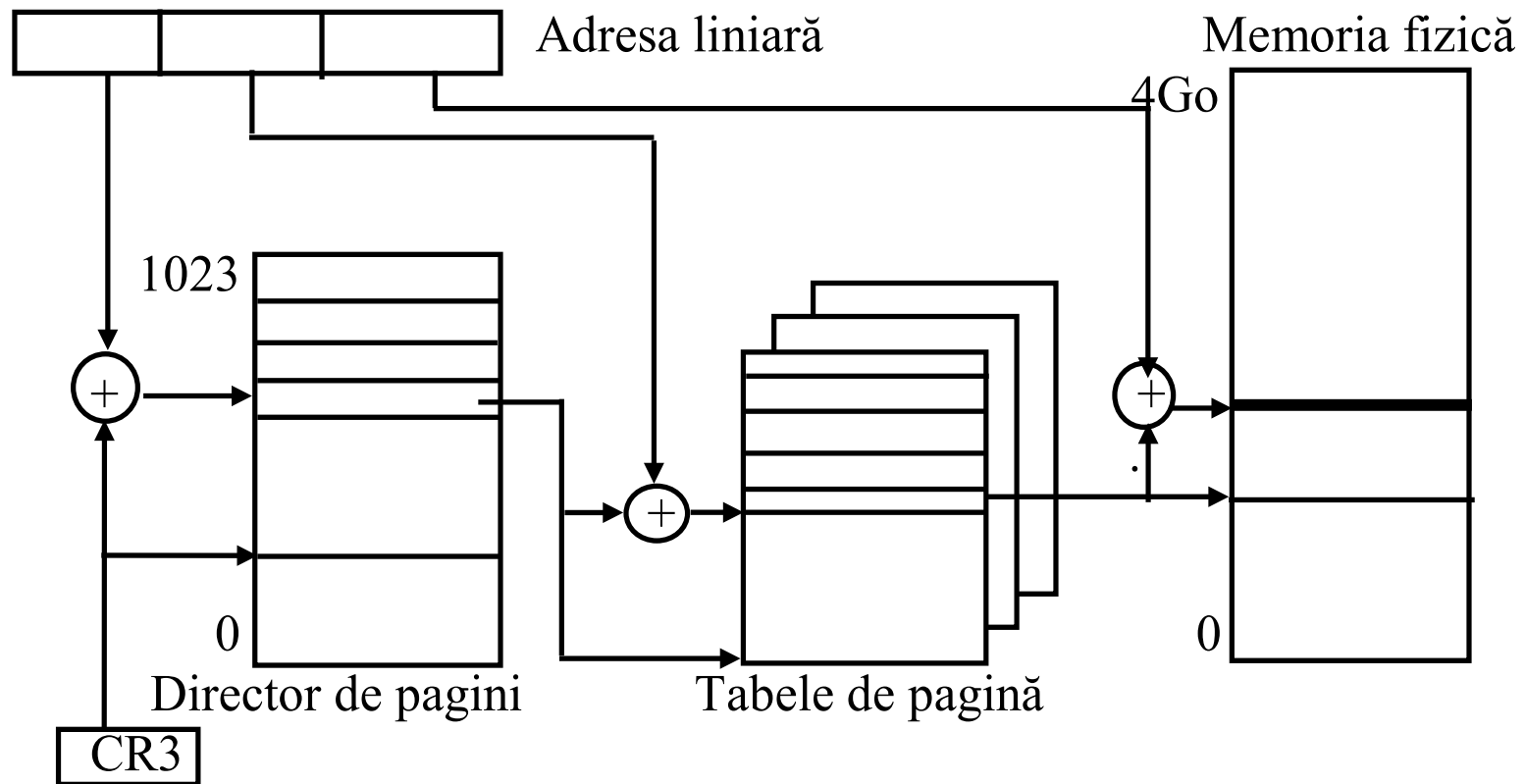


Figura 9-21 Transformarea adresei liniare în adresă fizică



Paginarea – operatia de scriere

- Problema: la scriere apare o inconsistenta intre memoria interna si cea virtuala
 - este mai critica in cazul sistemelor multi-master (multi-procesor)
- Solutia: Write back
 - tehnica “write through” nu este fezabila deoarece timpul de acces la memoria externa este foarte mare, ceea ce ar afecta timpul mediu de acces la memorie