Sisteme de memorie

- Ierarhia memoriilor
- Tipuri de memorii
- Memorii semiconductoare
- Memoria cu unități multiple
- Memoria asociativă
- Memoria *cache*
- Memoria virtuală

Memoria asociativă (1)

- Memorii RAM: datele sunt identificate cu ajutorul unor adrese unice
- Memorii asociative: datele sunt identificate prin conținutul acestora → memorii adresabile prin conținut (CAM – *Content Addressable Memory*)
- Memorie asociativă cu *n* cuvinte: timpul de căutare a unei date este independent de *n*
  - Toate cuvintele din memorie pot fi comparate în paralel cu cuvântul căutat

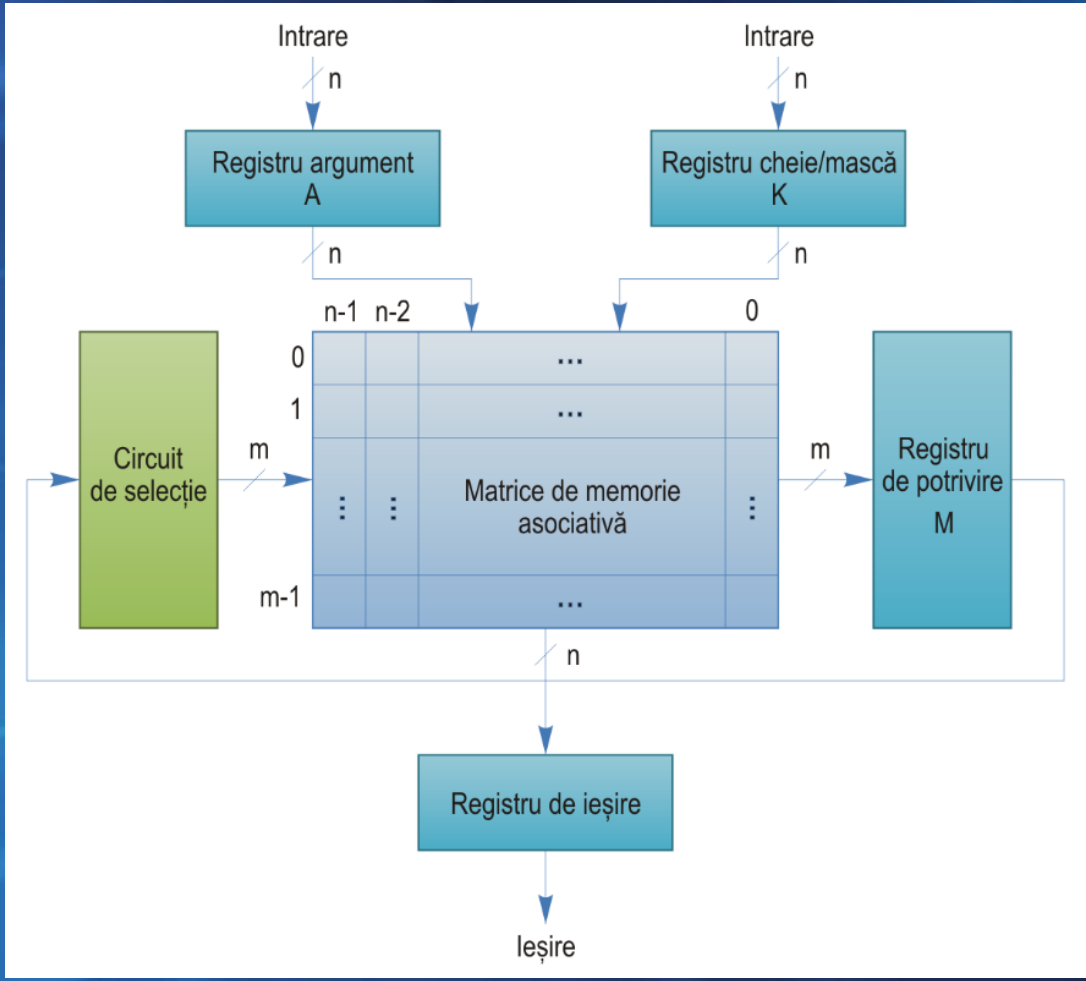
Memoria asociativă (2)

- Cost mai ridicat decât al memoriilor RAM
- Exemple de utilizare:
  - Memorii *cache* → translatarea adreselor
  - Recunoașterea modelelor
- Două tipuri de memorii asociative:
  - *Cu potrivire exactă:* datele sunt identificate pe baza egalității cu o cheie
  - *Cu comparație:* utilizează operatori relaționali

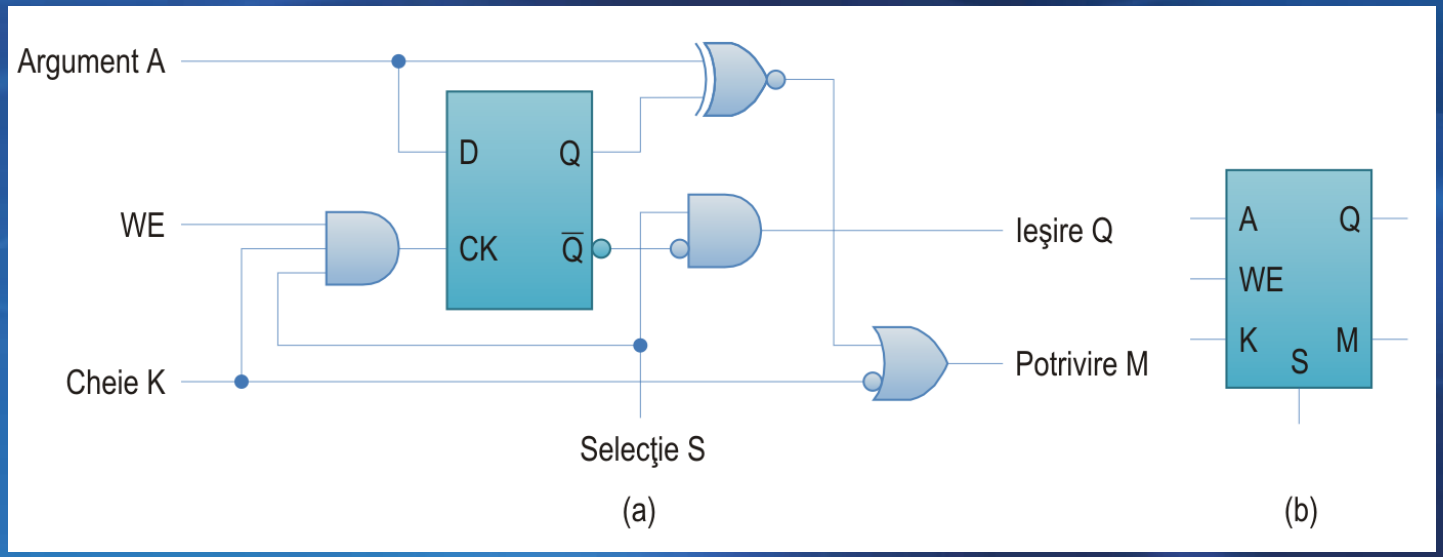
Memoria asociativă (3)

- Organizarea logică: fixă sau variabilă
- *Organizare fixă*:
  - Un cuvânt este împărțit în segmente fixe
  - *Segment cheie*: fiecare bit al acestui segment trebuie utilizat pentru interogare
- *Organizare variabilă*:
  - Un cuvânt poate fi împărțit în segmente fixe
  - Orice parte a unui segment poate fi utilizată pentru interogare → complet interogabilă

Memoria asociativă (4)



Memoria asociativă (5)



Celulă de memorie asociativă

- Sunt necesare în jur de 10 tranzistoare

Memoria asociativă (6)

- Prelucrare asociativă
  - Executată cu un procesor asociativ
  - Procesorul asociativ conține o memorie asociativă și o logică de control suplimentară
  - Memoria asociativă poate executa în paralel instrucțiuni primitive
- Exemple de instrucțiuni executate de o memorie asociativă:
  - SET: setarea la 1 a tuturor biților de potrivire

Memoria asociativă (7)

- COMPARE: comparații paralele ale argumentului mascat cu toate cuvintele
- READ: transmiterea pe liniile de date a cuvintelor care au biții de potrivire setați
- WRITE: scrierea în paralel a argumentului mascat în toate cuvintele care au biții de potrivire setați
- REPORT: raportarea unității centrale dacă există o potrivire după o căutare precedentă

Memoria asociativă (8)

- Avantajele prelucrării asociative:
  - Viteza ridicată: pentru anumite probleme, timpul de execuție se poate reduce cu un factor de *n*
  - Un mod mai natural de a soluționa anumite probleme de calcul
  - Operațiile sunt executate local
- Aplicații: prelucrarea imaginilor; urmărirea semnalelor radar; inteligența artificială în timp real

Sisteme de memorie

- Ierarhia memoriilor
- Tipuri de memorii
- Memorii semiconductoare
- Memoria cu unități multiple
- Memoria asociativă
- Memoria *cache*
- Memoria virtuală

Memoria *cache*

- Memoria *cache*
  - Principiul memoriei *cache*
  - Organizarea memoriei *cache*
  - Funcționarea memoriei *cache*
  - Translatarea adreselor
    - Translatarea asociativă
    - Translatarea directă
    - Translatarea cu seturi asociative
  - Strategii de înlocuire
  - Performanța memoriei *cache*

Principiul memoriei *cache* (1)

- Memorie rapidă, utilizată temporar pentru păstrarea unei porțiuni a datelor și instrucțiunilor în vederea utilizării imediate
- Memoria *cache* este mai costisitoare ⟶ dimensiunea acestei memorii într-un sistem de calcul este limitată
  - O mare parte din cererile de acces vor fi satisfăcute de memoria *cache* ⟶ proprietatea de localitate a referințelor

Principiul memoriei *cache* (2)

- Amplasare: între memoria principală și UCP
  - Conține cópii ale unor blocuri din memoria principală
  - Dacă un cuvânt solicitat de UCP se află în memoria *cache*, nu va fi necesar accesul la memoria principală mai lentă
- Îmbunătățirea performanței: amplasarea memoriei *cache* în același circuit integrat ca și procesorul

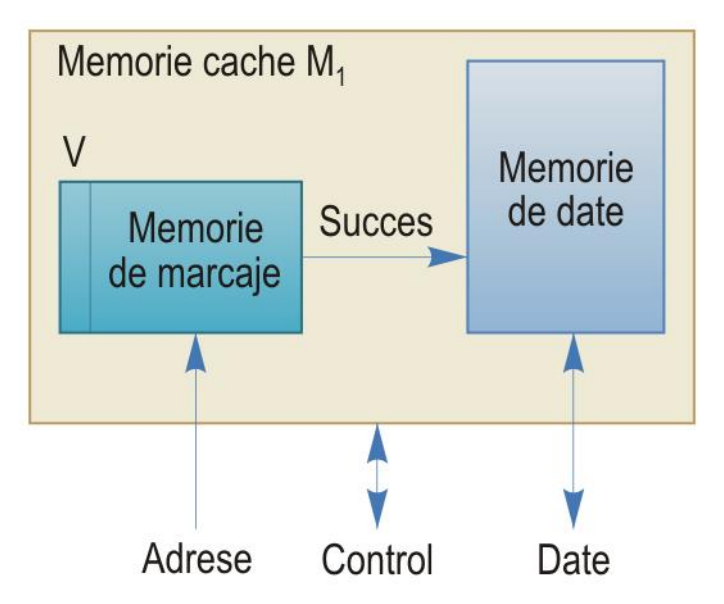
Memoria *cache*

- Memoria *cache*
  - Principiul memoriei *cache*
  - Organizarea memoriei *cache*
  - Funcționarea memoriei *cache*
  - Translatarea adreselor
    - Translatarea asociativă
    - Translatarea directă
    - Translatarea cu seturi asociative
  - Strategii de înlocuire
  - Performanța memoriei *cache*

Organizarea memoriei *cache* (1)

- Cuvintele de memorie: păstrate într-o *memorie de date*
  - Sunt grupate în pagini → blocuri sau linii
- Fiecare bloc al memoriei *cache* este marcat cu adresa sa de bloc → marcaj (“*tag*”)
- Colecția adreselor de marcaj asignate momentan memoriei *cache*: păstrată într-o *memorie de marcaje*

Organizarea memoriei *cache* (2)



- Bit de validare (V) adăugat fiecărui marcaj
  - Indică prin valoarea 1 că marcajul conține o adresă validă

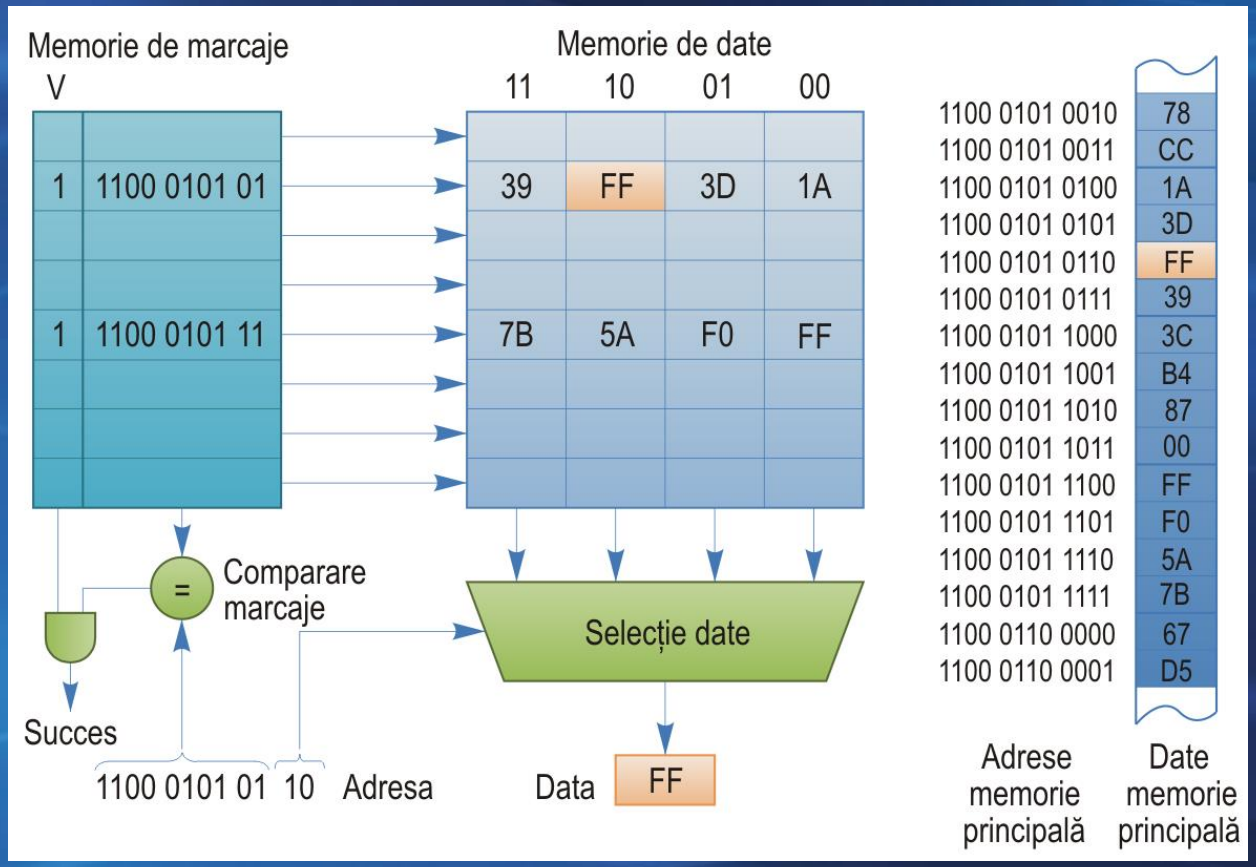
Memoria *cache*

- Memoria *cache*
  - Principiul memoriei *cache*
  - Organizarea memoriei *cache*
  - Funcționarea memoriei *cache*
  - Translatarea adreselor
    - Translatarea asociativă
    - Translatarea directă
    - Translatarea cu seturi asociative
  - Strategii de înlocuire
  - Performanța memoriei *cache*

Funcționarea memoriei *cache* (1)

- Atunci când UCP generează o cerere de citire:
  - Cererea este trimisă la memoria *cache*
  - Dacă blocul cuvântului nu este găsit în memoria *cache*: cuvântul solicitat este furnizat de memoria principală
  - Dacă blocul cuvântului este găsit în memoria *cache*: nu este necesar accesul la memoria principală

Funcționarea memoriei *cache* (2)

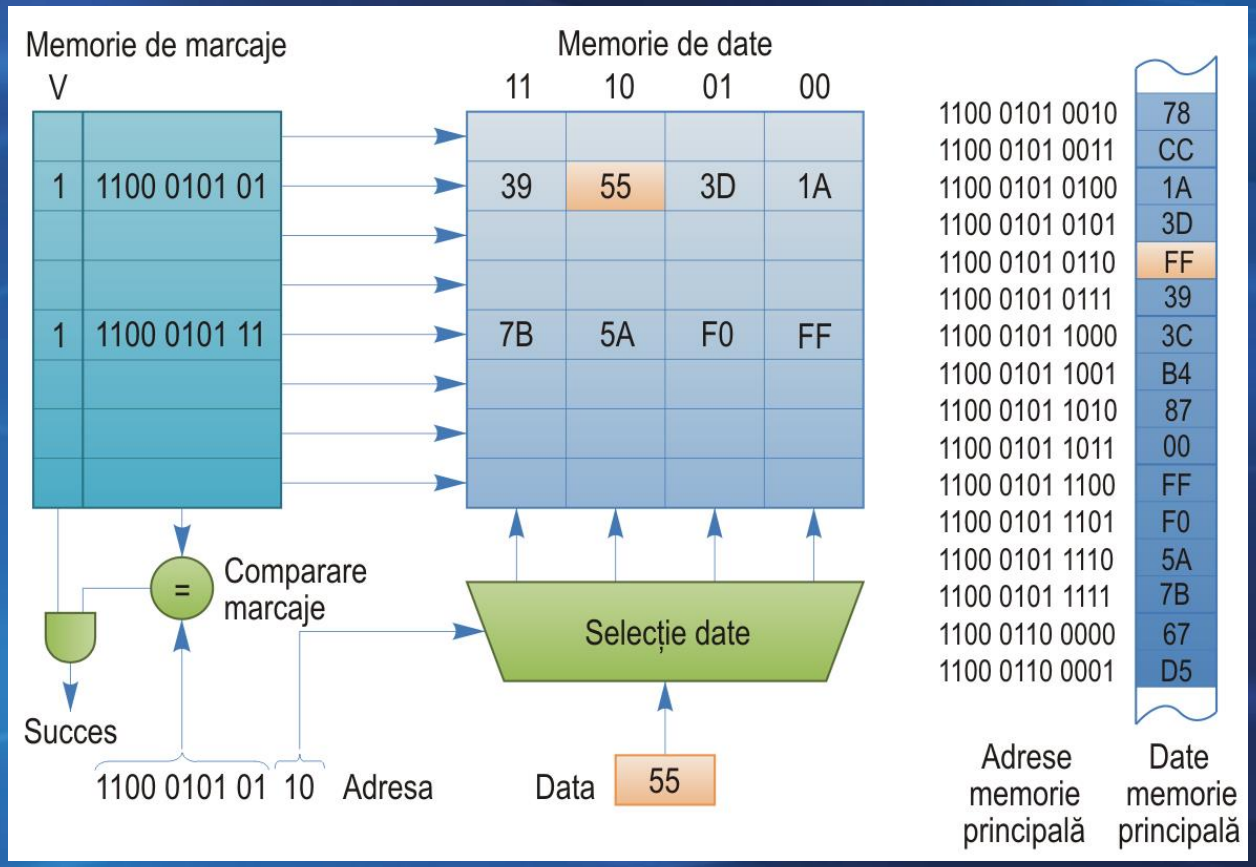


Execuția unei operații de citire

Funcționarea memoriei *cache* (3)

- Atunci când UCP generează o cerere de scriere:
  - Cererea este trimisă la memoria *cache*
  - Dacă blocul cuvântului nu este găsit în memoria *cache*: se încarcă o copie a blocului cuvântului din memoria principală în memoria *cache*
    - Se execută o operație de scriere
  - Dacă blocul cuvântului este găsit în memoria *cache*: se execută o operație de scriere

Funcționarea memoriei *cache* (4)



Execuția unei operații de scriere

Funcționarea memoriei *cache* (5)

- Poate apare incoerența datelor:
  - Data din memoria *cache* diferă de data din memoria principală cu aceeași adresă
- O incoerență temporară este acceptabilă
- Problema coerenței memoriilor *cache*: prevenirea utilizării improprii a datei vechi
  - Sisteme multiprocesor, dacă fiecare procesor are o memorie *cache* proprie
  - Sisteme uni-procesor, cu controlere care au acces direct la memoria principală

Funcționarea memoriei *cache* (6)

- Strategii de scriere:
  - *Write-back*, *write-through*
- Strategia *write-back*
  - Un bloc este scris doar în memoria *cache*
  - Un bloc modificat în memoria *cache* este scris în memoria principală doar la înlocuirea acestuia
  - Fiecărui bloc din memoria *cache* i se asociază un *bit de modificare* (“*dirty bit*”)
    - Dacă un bloc trebuie înlocuit, acesta este scris în memoria principală doar dacă bitul său de modificare este setat

Funcționarea memoriei *cache* (7)

- Avantaje:
  - Operații multiple de scriere în memoria *cache* necesită o singură scriere în memoria principală → se reduce încărcarea magistralei
  - Se reduce consumul de energie
- Dezavantaje:
  - Memoria *cache* și memoria principală pot fi temporar incoerente
  - Se complică recuperarea în cazul defectelor de sistem

Funcționarea memoriei *cache* (8)

- Strategia *write-through*
  - Un bloc este modificat atât în memoria *cache*, cât și în memoria principală
  - Avantaje:
    - Implementare mai simplă
    - Se păstrează coerența datelor
    - Se simplifică gestiunea memoriilor *cache* cu nivele multiple
  - Dezavantaj:
    - Încetinește UCP

Memoria *cache*

- Memoria *cache*
  - Principiul memoriei *cache*
  - Organizarea memoriei *cache*
  - Funcționarea memoriei *cache*
  - Translatarea adreselor
    - Translatarea asociativă
    - Translatarea directă
    - Translatarea cu seturi asociative
  - Strategii de înlocuire
  - Performanța memoriei *cache*

Translatarea adreselor

- Translatarea adreselor de memorie specificate de UCP în locațiile din memoria *cache*
- Tipuri de translatare a adreselor:
  - Translatare asociativă
  - Translatare directă
  - Translatare cu seturi asociative
- Tipul de translatare determină locația din memoria *cache* în care se poate amplasa un bloc din memoria principală

Memoria *cache*

- Memoria *cache*
  - Principiul memoriei *cache*
  - Organizarea memoriei *cache*
  - Funcționarea memoriei *cache*
  - Translatarea adreselor
    - Translatarea asociativă
    - Translatarea directă
    - Translatarea cu seturi asociative
  - Strategii de înlocuire
  - Performanța memoriei *cache*

Translatarea asociativă (1)

- Memorie *cache* cu *asociativitate totală*
- Un bloc de memorie poate fi amplasat în orice locație a memoriei *cache*
- La căutarea unui bloc, ar trebui parcurse toate locațiile memoriei *cache*
  - Este necesară o memorie asociativă
- Organizarea: combinație între o memorie asociativă și o memorie RAM
  - Marcajele: păstrate în memoria asociativă
  - Datele: păstrate în memoria RAM

Translatarea asociativă (2)

- Fiecare marcaj conține doar o parte a adresei din memoria principală
  - Nu conține deplasamentul octetului din bloc
- Adresele de memorie conțin două părți:
  - Deplasament: *d* biți de ordin inferior ai adresei din memoria principală → identifică octetul dintr-un bloc
  - Marcaj: *t* biți de ordin superior rămași ai adresei din memoria principală → identifică blocul
- Dezavantaj: necesită o memorie asociativă

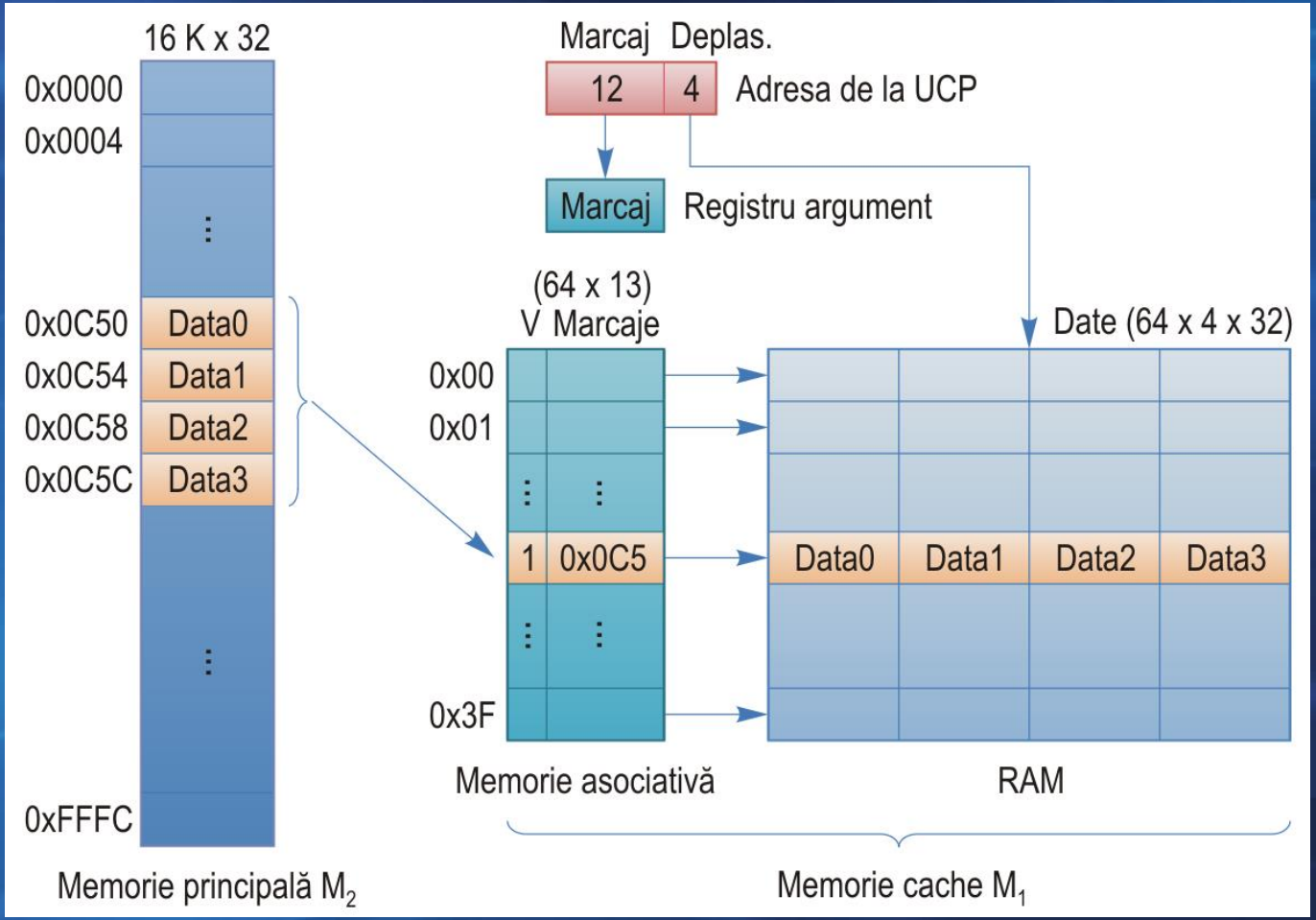
Translatarea asociativă (3)

- Exemplul 3.1
  - Structura unei memorii *cache* cu translatare asociativă
  - Memorie principală: 64 KB
    - Adresabilă la nivel de octet (B)
    - Adrese de memorie de 16 biți
  - Cuvinte de date: 32 biți (4 B)
  - Dimensiunea memoriei cache: 1 KB
  - Dimensiunea unui bloc: 4 cuvinte (16 B)

Translatarea asociativă (4)

- Adresarea unui B din blocul de 16 B: cu 4 biți ($2^4 = 16$)
  - Deplasament: $d = 4$
- Număr de blocuri din memoria principală: 64 KB / 16 B = 2^{16} B / 2^4 B = 2^{12}
  - Marcaj: $t = 12$ ($t = 16 - d = 16 - 4$)
- Număr de blocuri din memoria *cache*: 1 KB / 16 B = 2^{10} B / 2^4 B = $2^6 = 64$
- Număr de cuvinte din memoria de date: 1 KB / 4 B = 2^{10} B / 2^2 B = $2^8 = 256$

Translatarea asociativă (5)



Memoria *cache*

- Memoria *cache*
  - Principiul memoriei *cache*
  - Organizarea memoriei *cache*
  - Funcționarea memoriei *cache*
  - Translatarea adreselor
    - Translatarea asociativă
    - Translatarea directă
    - Translatarea cu seturi asociative
  - Strategii de înlocuire
  - Performanța memoriei *cache*

Translatarea directă (1)

- Se utilizează memorii RAM în locul memoriilor asociative $\rightarrow$ se reduce costul
- Memoria principală M_2 este divizată în *blocuri*
- Considerăm că memoria *cache* M_1 este împărțită în $b = 2^s$ regiuni $M_1(0)$, $M_1(1)$, …, $M_1(b\text{-}1) \rightarrow$ *seturi*
- Fiecare bloc $M_2(i)$ din M_2 este amplasat într-un set $M_1(j)$ din M_1
  - Adresa j a setului: $j = i \bmod b$

Translatarea directă (2)

- Exemplul 3.2
  - Memorie *cache* cu translatare directă
  - Număr de seturi: 64
  - Dimensiunea unui set: 16 B
  - Care este adresa setului în care se amplasează cuvântul cu adresa de octet 0x1234?
  - Adresa blocului de memorie pentru cuvântul cu adresa 0x1234: $\lfloor$ 0x1234 / 16 $\rfloor$ = 0x123
  - Adresa set = adresa_bloc mod număr_seturi = 0x123 mod 64

Translatarea directă (3)

- 0x123 / 64 = 0001 0010 0011_2 / 2^6 = 100_2, rest 10 0011_2 (0x23)
- Adresa setului: 0x23 (35)

- Adresele de memorie conțin trei părți:
  - Deplasament: *d* biți de ordin inferior ai adresei din memoria principală ⟶ selectează octetul dorit dintr-un set
  - Index: *s* biți următori ai adresei din memoria principală ⟶ identifică setul memoriei *cache* care poate memora blocul

Translatarea directă (4)

- Marcaj: *t* biți de ordin superior rămași ai adresei din memoria principală ⟶ identifică blocul din memoria principală
- La adresa specificată de index:
  - Se memorează un marcaj în memoria de marcaje
  - Se memorează un bloc în memoria de date
- Memoria de marcaje poate fi o memorie RAM obișnuită, adresată de index (*s* biți)
- Caz particular: un set conține un cuvânt

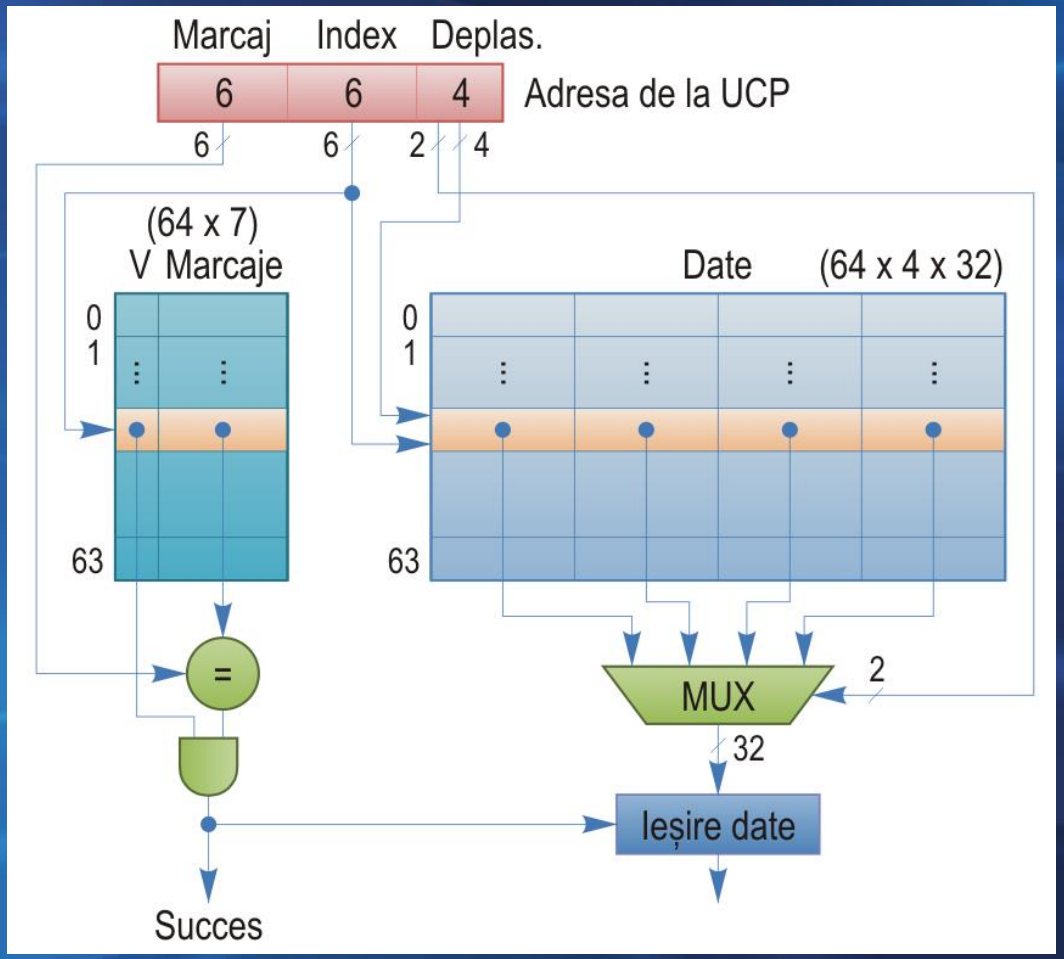
Translatarea directă (5)

- Avantaje ale translatării directe:
  - Necesită un număr mai redus de biți pentru fiecare bloc din memoria *cache*
  - Memorii *cache* cu mai multe cuvinte pe set: scade numărul total de marcaje și biți V
  - Nu necesită memorie asociativă
- Dezavantaj al translatării directe:
  - Performanțele pot fi reduse dacă două sau mai multe cuvinte cu același index, dar marcaje diferite, sunt accesate frecvent

Translatarea directă (6)

- Exemplul 3.3
  - Memoria *cache* din exemplul 3.1, organizată ca memorie *cache* cu translatare directă
  - Adresarea unui B din setul de 16 B: 4 biți
    - Deplasament: $d = 4$
  - Număr de seturi: 1 KB/16 B = 2^{10} B/2^4 B = 2^6
    - Index: $s = 6$
    - Marcaj: $t = 16 – (6 + 4) = 16 – 10 = 6$
  - Număr cuvinte de date: 1 KB/4 B = 2^{10} B/2^2 B = $2^8 = 256$

Translatarea directă (7)



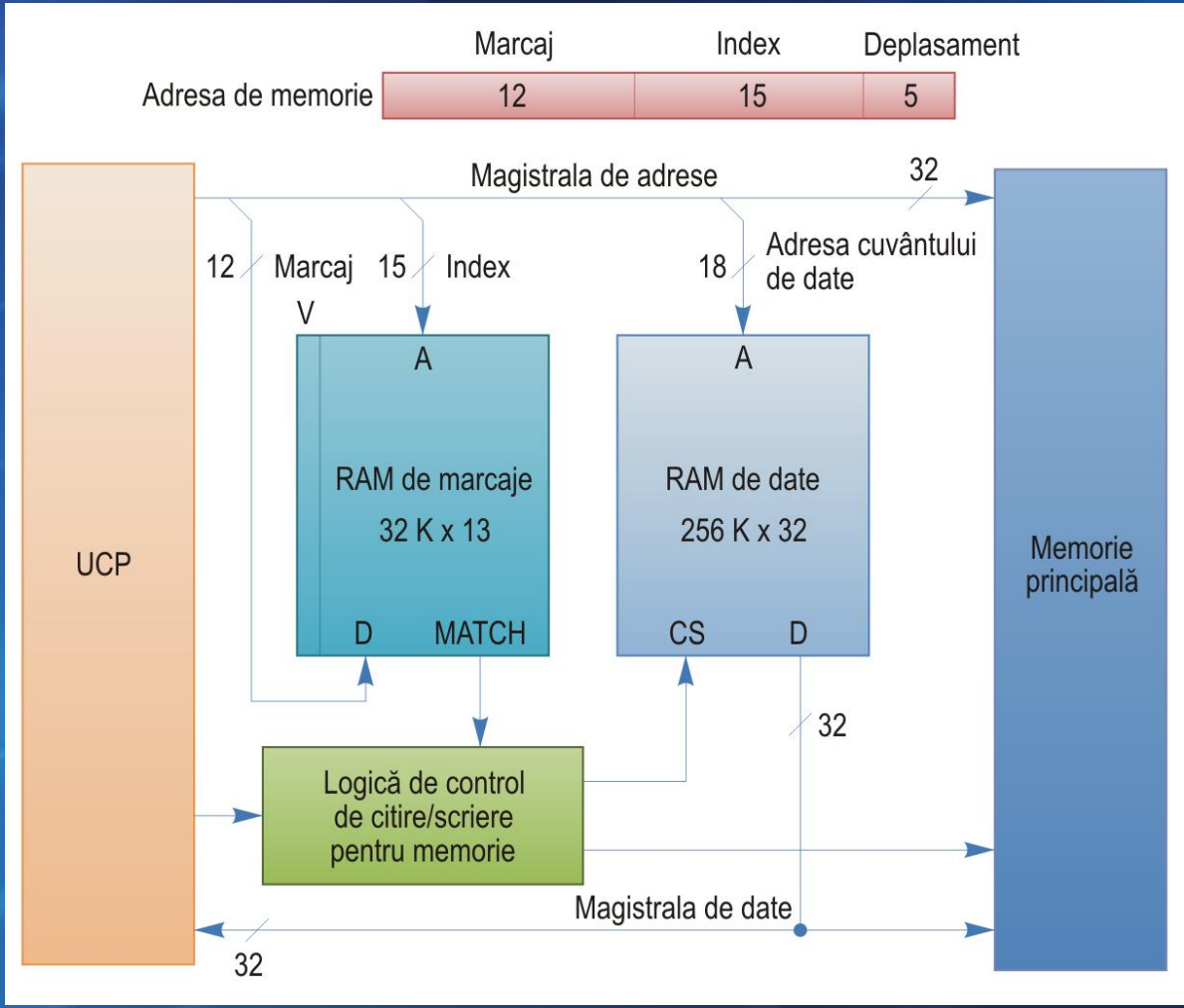
Translatarea directă (8)

- Exemplul 3.4
  - Proiectarea unei memorii *cache* cu translatare directă
  - Procesor conectat cu o memorie externă adresabilă la nivel de octet (B)
  - Magistrala de adrese: 32 biți
  - Magistrala de date: 32 biți
  - Dimensiunea memoriei *cache*: 1 MB
  - Dimensiunea unui set: 32 B

Translatarea directă (9)

- Adresarea unui B din setul de 32 B: $d = 5$
- Nr. de seturi: 1 MB / 32 B = 2^{20} B / 2^5 B = 2^{15}
  - Selectarea seturilor din memoria de date: $s = 15$ (biți pentru index)
- Marcaj: $t = 32 - (15 + 5) = 12$
- Dimensiunea memoriei de marcaje: 2^{15} x 13 biți
- Nr. de cuvinte din memoria de date: 1 MB / 4 B = 2^{20} B / 2^2 B = 2^{18} = 256 K cuvinte de 32 biți
  - Adrese de 18 biți pentru memoria de date

Translatarea directă (10)



Memoria *cache*

- Memoria *cache*
  - Principiul memoriei *cache*
  - Organizarea memoriei *cache*
  - Funcționarea memoriei *cache*
  - Translatarea adreselor
    - Translatarea asociativă
    - Translatarea directă
    - Translatarea cu seturi asociative
  - Strategii de înlocuire
  - Performanța memoriei *cache*

Translatarea cu seturi asociative (1)

- Permite memorarea mai multor blocuri cu același index
- Memoria *cache* M_1 este divizată în $b = 2^s$ seturi
- Fiecare set poate memora $k = 2^m$ blocuri
- Un bloc este translatat într-un set (cu funcția mod); blocul poate fi amplasat oriunde în set
- Translatarea asociativă și cea directă: cazuri speciale ale translatării cu seturi asociative
  - $k = 1$: translatare directă
  - $b = 1$: translatare complet asociativă

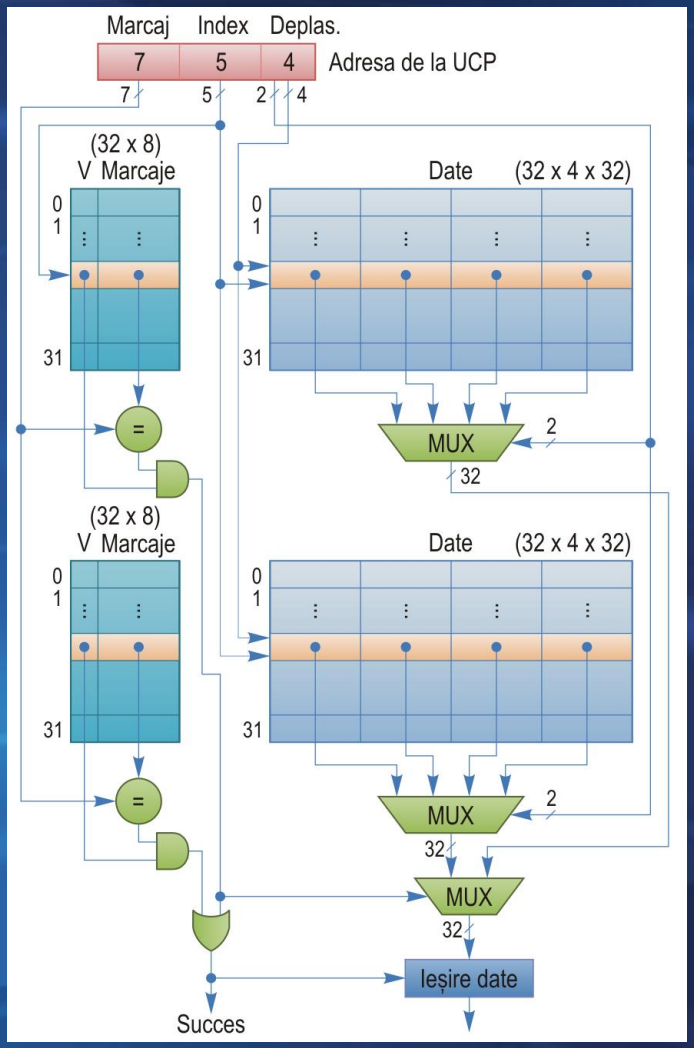
Translatarea cu seturi asociative (2)

- În practică, se utilizează valori mici ale numărului *k* (de ex., *k* = 4)
  - Permite utilizarea unor memorii RAM pentru memorarea marcajelor
- Memorie *cache* cu seturi asociative cu *k* blocuri pe set: set asociativ cu *k* căi
  - Poate memora *k* blocuri cu același index
- Fiecare bloc de memorie: aceeași structură ca și memoria *cache* cu translatare directă

Translatarea cu seturi asociative (3)

- Exemplul 3.5
  - Memoria *cache* din exemplul 3.1, organizată ca un set asociativ cu două căi
  - Adresarea unui B din setul de 16 B: $d = 4$
  - Număr de seturi din fiecare cale: 1 KB/2x16 B = 2^{10} B/2x2^4 B = $2^{10}/2^5 = 2^5 = 32$
    - Index: $s = 5$
    - Marcaj: $t = 16 – (5 + 4) = 16 – 9 = 7$
  - Număr de cuvinte din blocurile memoriei de date: 1 KB/2x4 B = 2^{10} B/2^3 B = $2^7 = 128$

Translatarea cu seturi asociative (4)



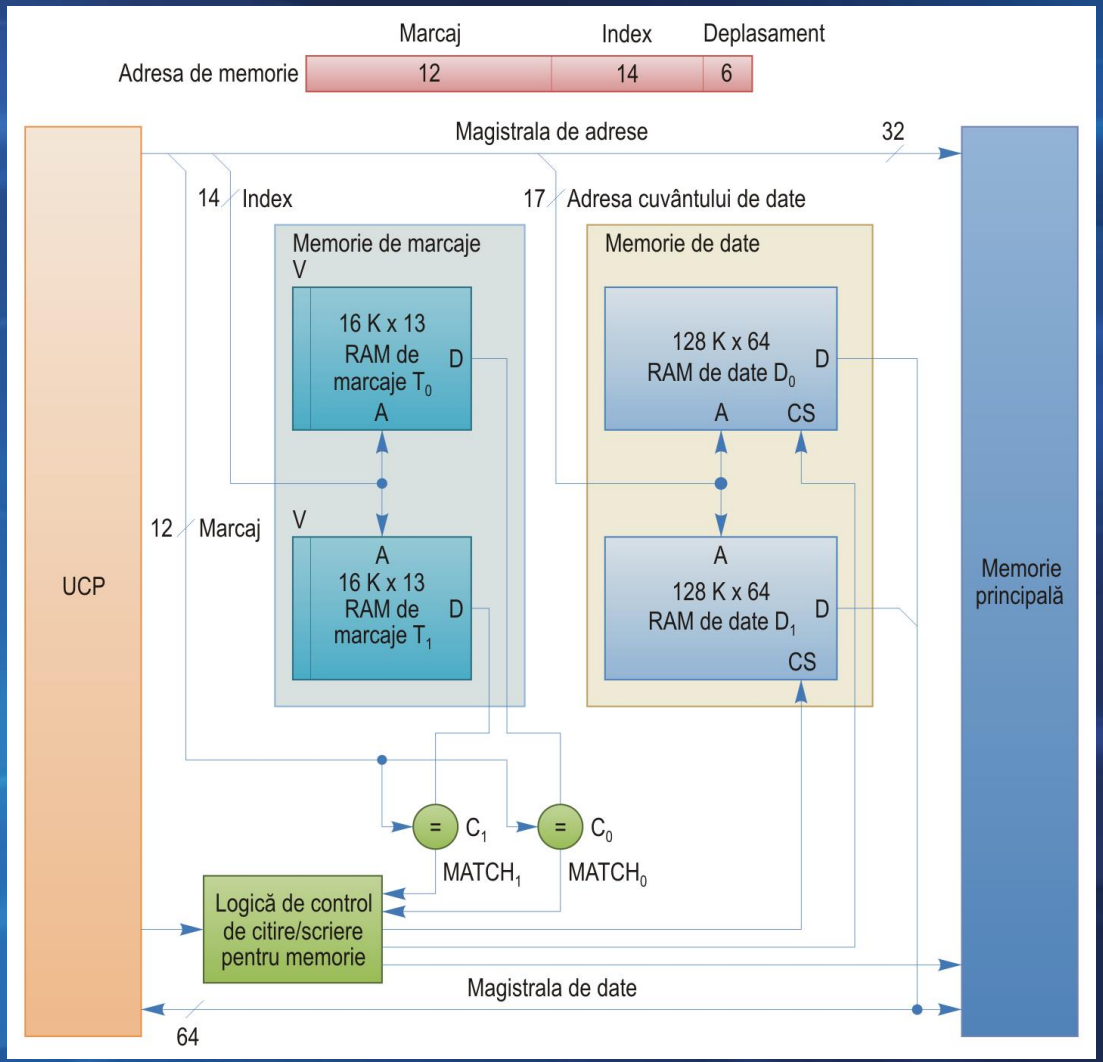
Translatarea cu seturi asociative (5)

- Exemplul 3.6
  - Proiectarea unei memorii *cache* cu seturi asociative
  - Procesor de 32/64 biți
  - Magistrala de adrese: 32 biți
  - Magistrala de date: 64 biți
  - Dimensiunea memoriei *cache*: 2 MB
  - Dimensiunea unui set: 64 B
  - Set asociativ cu două căi

Translatarea cu seturi asociative (6)

- Adresarea unui B din setul de 64 B: $d = 6$
- Nr. de seturi din fiecare cale: 2 MB / 2x64 B = 2^{21} B / 2x2^6 B = 2^{21} / 2^7 = 2^{14}
  - Selectarea seturilor din memoria de date: $s = 14$ (biți pentru index)
- Marcaj: $t = 32 - (14 + 6) = 12$
- Memoria de marcaje: 2 x (2^{14} x 13 biți)
- Nr. de cuvinte din blocurile memoriei de date: 2 MB / 2x8 B = 1 MB / 8 B = 2^{20} B / 2^3 B = 2^{17}
  - Adrese de 17 biți pentru memoria de date
- Memoria de date: 2 x (2^{17} x 64 biți)

Translatarea cu seturi asociative (7)



Memoria *cache*

- Memoria *cache*
  - Principiul memoriei *cache*
  - Organizarea memoriei *cache*
  - Funcționarea memoriei *cache*
  - Translatarea adreselor
    - Translatarea asociativă
    - Translatarea directă
    - Translatarea cu seturi asociative
  - Strategii de înlocuire
  - Performanța memoriei *cache*

Strategii de înlocuire (1)

- Dacă nu mai există spațiu în memoria *cache*, trebuie să se înlocuiască un bloc
- Înlocuire aleatorie
  - Alege un bloc în mod aleatoriu și îl înlocuiește cu blocul care conține data nou accesată
  - Avantaj: implementare simplă
  - Dezavantaj: blocurile cu probabilitatea cea mai mare de a fi utilizate din nou au aceeași șansă de a fi eliminate ca și cele care nu vor fi utilizate din nou

Strategii de înlocuire (2)

- Cel mai puțin frecvent utilizat (LFU – *Least Frequently Used*)
  - Pentru fiecare bloc, se păstrează un contor al numărului total de utilizări
  - Avantaj: un bloc utilizat frecvent are o probabilitate mai mare de a rămâne în memoria *cache*
  - Dezavantaj: blocurile care au fost încărcate recent au o valoare mică a contorului

Strategii de înlocuire (3)

- Cel mai puțin recent utilizat (LRU – *Least Recently Used*)
  - Un bloc care nu a fost utilizat pentru o perioadă lungă de timp are o șansă mai redusă de a fi utilizat în viitorul apropiat
  - Se păstrează evidența acelor blocuri care au fost accesate cel mai recent → contor
  - Are performanța cea mai bună raportată la cost comparativ cu celelalte metode

Strategii de înlocuire (4)

- Aproximare a strategiei LRU (Pseudo-LRU)
  - Fiecărui set din memoria *cache* i se alocă un număr de biți, câte un bit pentru fiecare cale
  - La accesarea unui bloc, este setat bitul asociat căii care conține blocul accesat
  - Dacă toți biții asociați unui set sunt setați, aceștia se resetează, exceptând bitul setat cel mai recent
  - Se înlocuiește blocul dintr-o cale cu bitul asociat resetat $\rightarrow$ o altă metodă dacă există mai multe opțiuni

Strategii de înlocuire (5)

- Exemplul 3.7
  - Memorie *cache* cu 4 seturi asociative cu 2 căi
  - Blocuri de memorie de un cuvânt
  - Strategia de înlocuire LRU

Adresa cuvântului	1	2	8	9	20	26	27	28	42	1	19	52
Set 0			8	8	8*	8*	8*	28	28	28	28	28*
					20	20	20	20*	20*	20*	20*	52
Set 1	1	1	1	1*	1*	1*	1*	1*	1*	1	1	1
				9	9	9	9	9	9	9*	9*	9*
Set 2		2	2	2	2	2*	2*	2*	42	42	42	42
						26	26	26	26*	26*	26*	26*
Set 3							27	27	27	27	27*	27*
											19	19

Rezumat (1)

- Memoria asociativă permite determinarea rapidă a prezenței unui cuvânt în memorie
  - Conține o logică suplimentară pentru compararea în paralel a tuturor cuvintelor cu cuvântul căutat
  - Poate fi cu potrivire exactă sau cu comparație, cu organizare fixă sau organizare variabilă
- Un procesor asociativ conține o memorie asociativă și poate executa în paralel instrucțiuni primitive
  - Este avantajos pentru anumite probleme

Rezumat (2)

- Memoria *cache* este o memorie rapidă care păstrează o parte a instrucțiunilor și a datelor
  - Permite reducerea numărului de accese la memoria principală
  - Este formată dintr-o memorie de date și o memorie de marcaje
  - Se poate utiliza una din două strategii de scriere: *write-back* sau *write-through*
- Translatarea asociativă a adreselor asigură flexibilitatea maximă, dar necesită o memorie asociativă

Rezumat (3)

- Translatarea directă permite utilizarea unei memorii RAM, dar reduce flexibilitatea prin amplasarea unui bloc într-o anumită locație fixă
- Translatarea cu seturi asociative este cea mai utilizată; permite memorarea într-un set a mai multor blocuri cu același index
- Există diferite strategii de înlocuire a unui bloc din memoria *cache*: înlocuire aleatorie; cel mai puțin frecvent utilizat (LFU); cel mai puțin recent utilizat (LRU); pseudo-LRU

Noțiuni, cunoștințe (1)

- Principiul memoriei asociative
- Tipuri de memorii asociative
- Schema bloc a unei memorii asociative
- Schema unei celule de memorie asociativă
- Prelucrarea asociativă
- Principiul memoriei *cache*
- Organizarea memoriei *cache*
- Execuția unei operații de citire și de scriere într-un sistem cu memorie *cache*

Noțiuni, cunoștințe (2)

- Problema coerenței memoriilor *cache*
- Strategii de scriere pentru memoria *cache*
- Principiul memoriei *cache* cu translatarea asociativă a adreselor
- Principiul memoriei *cache* cu translatare directă a adreselor
- Principiul memoriei *cache* cu seturi asociative
- Strategii de înlocuire pentru memoria *cache*