SUBIECT Memorie CACHE

Folosind cunoștințele predate online la cursurile 12.1 și Curs practic – Arhitectura și performanța subsistemului memoriei CACHE postate pe cursul CN1 de pe Moodle și prezentate live prin Curs 12 video Memoria CACHE, pe baza exemplului de la pagina 14/22 din cursul 12.1, pentru o memorie CACHE (MC), inițial goală, de capacitate 16 cuvinte și lungimea blocului de 2 cuvinte, pentru secvența de accese la memorie cu adresa zecimală a blocului din memoria principală (MP) în ordinea LD 0, LD 8, LD 16, ST 0, LD 12, LD 8, ST 20, LD 8, ST 16, LD 22, LD 24, LD 26, LD 30, LD 32, ST 16, ST 0, LD 14, se cer următoarele:

a) Dacă memoria CACHE are corespondență directă, să se calculeze frecvența de eșec (Miss Rate) afișând și harta memoriei CACHE și precizând tipul de Miss de acces (de prim acces sau COMPULSORY, de conflict de bloc sau CONFLICT sau de capacitate sau CAPACITY).
b) Dacă memoria CACHE are corespondență set-asociativă pe (1<N<număr blocuri) căi, unde N este gradul de asociativitate și că algoritmul de înlocuire în caz de Miss al blocului din memoria CACHE este de tip LRU (Least Recently Used), să se calculeze frecvența de eșec (Miss Rate) pentru toate cazurile de asociativitate posibile afișând, de asemenea și harta memoriei CACHE și precizând tipul de Miss de acces (de prim acces sau COMPULSORY, de conflict de bloc sau CONFLICT sau de capacitate sau CAPACITY).
c) Dacă memoria CACHE are corespondență complet asociativă și că algoritmul de înlocuire în caz de Miss al blocului din memoria CACHE este de tip LRU (Least Recently Used), să se calculeze frecvența de eșec (Miss Rate) afișând și în acest caz harta memoriei CACHE și precizând tipul de Miss de acces (de prim acces sau COMPULSORY, de conflict de bloc sau CONFLICT sau de capacitate sau CAPACITY).
d) Dacă se aplică o strategie de scriere de tip WRITE-THROUGH prin care se scrie în același timp în memoria CACHE dar și în memoria principală MAIN, presupunând că timpul de acces la memoria CACHE este de 1 ciclu de ceas iar la memoria principală este de 100 de cicluri de ceas, să se calculeze timpul mediu de acces AMAT (Average Memory Access Time) la subsistemul de memorie CACHE-principală, pentru toate cazurile prevăzute la punctele a), b) și c). Să se facă o discuție succintă pe baza rezultatelor obținute.
e) Cum influențează Hit sau Miss la memoria CACHE accesele la memorie cerute de instrucțiunile din secvența de instrucțiuni de la subiectul de examen anterior?

NOTARE: 1,5 puncte pentru subpunctele a), c), d) și e), 3 puncte pentru subpunctul b) plus 1 punct din oficiu.

Precizări:

1. Dacă adresele pe care vi le-am dat, din motive de simplificare a lucrului în scop didactic, sunt de bloc de 2 cuvinte din memoria principală, adică LD 8 sau ST 8, etc, atunci nu vă luați după figura aceea cu 4 cuvinte de 4 octeți și vă luați după figurile de la pagina 21/87, apoi 26/87 și 29/87 având grijă că în acele figuri seturile sunt pe verticală iar blocurile (căile) sunt pe orizontală iar în cursul 12.1 în română seturile sunt de fapt căile conform exemplului de la pagina 16/22 unde, dacă ar fi fost și accese la adrese impare, s-ar fi dublat blocurile setului cerut așa cum am vorbit noi la cursul video de joi 21.05.2020 (aseară).
   Adică, considerați că se dă adresa în valoare absolută a blocului de 2 cuvinte din pagina memoriei principale în loc de Byte Offset, doar în scop didactic ca să învățați strategia de plasare a blocurilor din memoria principală în memoria CACHE.

2. Dacă adresele vi le dădeam sub forma nu LD 8 sau ST 8 ci LW s1, 0x8 sau SW s1, 0x8 sau LD 0x8, ST 0x8, etc atunci, având în vedere că lucrăm cu blocuri de 8 octeți (2 cuvinte de câte 4 octeți), trebuie să observați că cuvântul 0x8 aparține blocului 1 (nu 8 ca în primul caz) pentru că blocul 0 are 8 octeți (0x0000 – 0x0007) iar blocul 1 are tot 8 octeți (0x0008-0x000F) deci are octetul nostru din cuvântul cerut prin LW (SW) s1, 0x8. În CACHE se aduce întotdeauna întregul bloc de 8 octeți, adică la 0x8 se aduce blocul 1 din pagina 0 a memoriei principale format din 8 octeți (8-15) astfel încât că la un acces următor de forma LW (SW) 0x12 se detectează HIT pentru că la LW (SW) a fost adus și acest cuvânt prin LW (SW) 0x8. În cazul 1 se consideră că se face LOAD sau STORE pentru un bloc întreg de lungimea dată (în cazul nostru de 2 cuvinte cu 8 octeți) și nu mai trebuie să calculăm adresa efectivă de bloc cerut pentru acces de către procesor ca în limbajul de asamblare.
   Adică, în cazul 2, pentru problema noastră, considerați că se dă Byte Offset, adică pentru LD 0x08 (unde adresa este în hexazecimal pe 32 de biți) se dă adresă de forma în binar 0…00001000 unde primii 2 biți din dreapta reprezintă Byte Offset, următorul bit reprezintă adresa cuvântului din bloc adică Block Offset (numai pentru blocuri cu mai mult de 1 cuvânt lungime), următorii biți reprezintă adresa setului (3 biți pentru mapare directă sau, pentru maparea set-asociativă pe 4 căi, 1 bit deoarece căile din fiecare set sunt implicit date) iar restul, până la 32 de biți, reprezintă adresa paginii din memoria principală.

Și la 1) și la 2) se lucrează la fel, diferența este cum interpretăm adresele de bloc cerute de procesor, direct în cazul LD (ST) 8, etc sau calculând-o ca mai sus ținând cont de cuvântul (semi cuvântul sau octetul) cerut prin LW (SW) sau LH (SH) sau LB (SB) sau LD 0x8 (ST 0x8). Totul se întâmplă considerând adresare imediată, în caz de alt mod de adresare trebuie să calculați adresa efectivă și apoi să continuați ca la punctul 1 sau 2.

La test, **dat în scop didactic**, se rezolva cu cazul 1 adică este vorba de adresă de bloc din memoria principală și se face modulo cu numărul de seturi pentru a calcula adresa blocului din CACHE.

În cazul 2, **cum se va da la examen**, nu se mai face cu modulo cu toată adresa ci se calculează adresa ținând cont de modul de adresare (în slide-urile în engleză este imediată dar se putea și cu alt mod de adresare) și apoi se face modulo cu partea din adresa de 32 de biți dată de procesor, mai puțin biții din dreapta (3 biți pentru problema noastră), care reprezintă adresa octetului din blocul de memorie CACHE.
La examen se rezolvă după cum vi se dau accesele conform celor 2 cazuri de mai sus.
După interpretarea corectă a adreselor de bloc cerute de accesele pe care vi le voi da, veți continua ca la punctul 1 ținând cont de mapare iar pentru cazul 2 veți scrie cuvântul(sau semi-cuvântul sau octetul) cerut în blocul corespunzător din CACHE.

Ca exemplu pentru testul 8 de curs, LD 16 implică
- pentru maparea directă: 16 mod 8 = 0 (se accesează setul 0)
- pentru maparea set-asociativă pe 2 căi: 16 mod 8/2 = 0 (se accesează setul 0 căutând în ambele căi) (la LD 18 aveam 18 mod 8/2 = 2, adică se accesează setul 2 căutând tot în ambele căi)
- pentru maparea set-asociativă pe 4 căi: 16 mod 8/4 = 0 (se accesează setul 0 căutând în ambele căi) (la LD 18 aveam 18 mod 8/4 = 0, adică se accesează setul 0 căutând tot în ambele căi)
- pentru maparea complet asociativă: 16 mod 8 = 0 (se accesează setul 0 căutând în ambele căi) (la LD 18 aveam 18 mod 8 = tot 2, adică se accesează tot setul 2 căutând tot în ambele căi)

În acest caz nu se gândește pe octeți sau semi-cuvinte sau cuvinte ci pe câte un bloc întreg (de lungimea cerută, în cazul nostru de 2 cuvinte de 8 octeți).

Pentru accese de forma LW 0x16 nu LD 16 se face conform punctului 2 de mai sus. Voi da și un exemplu de rezolvare pentru problema noastră dacă se interpretează accesele la nivel de octet.
Dacă identificăm setul din adresa desfășurată pe câmpuri, atunci nu mai facem modulo pentru că asta obțineam făcând modulo, dar e de calcul pe biți la fiecare load sau store.
Dacă identificăm doar numărul blocului prin împărțire la 8 octeți, atunci e mai ușor și apoi facem iar modulo număr de seturi.

În final, indiferent de tipul de mapare, trebuie să găsiți setul din care aparține cuvântul sau semi-cuvântul sau octetul cerut de procesor. În plus, se mai poate cere un mod de scriere și un alt mod de înlocuire.

Tipuri de MISS:

Miss de tip compulsory: acces eșuat de bloc la memoria CACHE cauzat de primul acces la un bloc care nu a mai fost niciodată în memoria CACHE. Se mai numește și **cold-start miss**.

Miss de tip capacity: acces eșuat de bloc la memoria CACHE care apare deoarece memoria CACHE, chiar și cu mapare complet asociativă, nu poate conține toate blocurile necesare pentru a satisface cererea de acces. În acest caz se face un acces la un bloc care a mai fost în CACHE dar, datorită faptului că memoria a fost plină, a fost înlocuit anterior de un alt bloc cerut de procesor, bloc care nu era atunci în CACHE.

Miss de tip conflict: acces eșuat de bloc la memoria CACHE care apare într-o mapare set-asociativă sau directă a memoriei CACHE când mai multe blocuri concură la același set și care sunt eliminate într-o mapare complet asociativă a memoriei CACHE de aceeași mărime. Se mai numește și **collision miss**.

Exemplu pentru o memorie CACHE set-asociativă pe m = 2 căi, de capacitate C = 64 octeți cu blocuri de lungime L = 8 octeți.

B = C/L = 64/8 = 8 blocuri

S = B/m = 8/2 = 4 seturi cu câte 2 căi (blocuri) fiecare

Adresa procesor este în octal pe 9 biți (primii 6 biți sunt adresa de bloc iar ultimii 3 biți sunt adresa octetului din bloc) iar adresa blocului (dintr-o cale a oricărui set din memoria CACHE) este în octal pe 6 biți.

De exemplu:

- Pentru adresa procesor 024 avem **Hit** pentru că blocul cu adresa 02 este în setul 2, calea 0
  În blocul 02 avem octeții 020 – 027, adică 8 octeți ca în toate blocurile

- Pentru adresa procesor 272 avem **Miss** pentru că blocul cu adresa 27_O = 010 111_B = 23_{BCD} nu se găsește nicăieri în memoria CACHE și, cum 23 mod 4 = setul 3, blocul va fi adus în setul 3 pe calea cerută de LRU

Set 0		**Set 1**		**Set 2**		**Set 3**		**Adresă**	**Tip miss**
Cale 0	**Cale 1**	**Cale 0**	**Cale 1**	**Cale 0**	**Cale 1**	**Cale 0**	**Cale 1**	**procesor**	**sau hit**
(00,	04)	(01,	05)	(02,	06)	(03,	07)	024	Hit (update 02 LRU)
(00,	04)	(01,	05)	(06,	02)	(03,	07)	100	Compulsory miss
(04,	10)	(01,	05)	(06,	02)	(03,	07)	270	Compulsory miss
(04,	10)	(01,	05)	(06,	02)	(07,	27)	570	Compulsory miss
(04,	10)	(01,	05)	(06,	02)	(27,	57)	074	Conflict miss
(04,	10)	(01,	05)	(06,	02)	(57,	07)	272	Conflict miss
(04,	10)	(01,	05)	(06,	02)	(07,	27)	004	Capacity miss
(10,	00)	(01,	05)	(06,	02)	(07,	27)	044	Capacity miss
(00,	04)	(01,	05)	(06,	02)	(07,	27)	640	Compulsory miss
(04,	64)	(01,	05)	(06,	02)	(07,	27)	000	Conflict miss
(64,	00)	(01,	05)	(06,	02)	(07,	27)	410	Compulsory miss
(64,	00)	(05,	41)	(06,	02)	(07,	27)	710	Compulsory miss
(64,	00)	(41,	71)	(06,	02)	(07,	27)	550	Compulsory miss
(64,	00)	(71,	55)	(06,	02)	(07,	27)	570	Capacity miss
(64,	00)	(71,	55)	(06,	02)	(27,	57)	410	Conflict miss

REZOLVARE SUBIECT MEMORIE CACHE

C = 16 cuvinte (capacitatea MC)
L = 2 cuvinte (mărimea blocului, aceeași la MP și MC)
B = C/L = 16/2 = 8 blocuri (numărul de blocuri din MC)
m – grad de asociativitate
S = B/m – număr de seturi

a) Directă -> 1 bloc/set (m =1)
b) N căi -> N blocuri/set (m = N)
c) Complet asociativă -> B blocuri/1 set (m = B)

a) LD 0, LD 8, LD 16, ST 0, LD 12, LD 8, ST 20, LD 8, ST 16, LD 22, LD 24, LD 26, LD 30, LD 32, ST 16, ST 0, LD 14
S = 8/1 = 8 seturi

AMP[n0-1:0] = AGP_ABL_BLO_BO și AMC[n1-1:0] = ABL_BLO_BO unde ABL este adresa blocului dintr-un grup din MP, aceeași cu cea din MC iar Tag =AGP (MP are 2^{n0-n1} grupuri de capacitate C)

0 mod 8 = 0
8 mod 8 = 0
16 mod 8 = 0
12 mod 8 = 4
20 mod 8 = 4
22 mod 8 = 6
24 mod 8 =0
26 mod 8 = 2
30 mod 8 = 6
32 mod 8 = 0
14 mod 8 = 6

Adresă set	000	001	010	011	100	101	110	111	Hit sau Miss
LD 0	(0)	-	-	-	-	-	-	-	Miss compulsory
LD 8	(8)	-	-	-	-	-	-	-	Miss compulsory
LD 16	(16)	-	-	-	-	-	-	-	Miss compulsory
ST 0	(0)	-	-	-	-	-	-	-	Miss conflict
LD 12	(0)	-	-	-	(12)	-	-	-	Miss compulsory
LD 8	(8)	-	-	-	(12)	-	-	-	Miss conflict
ST 20	(8)	-	-	-	(20)	-	-	-	Miss compulsory
LD 8	(8)	-	-	-	(20)	-	-	-	Hit
ST 16	(16)	-	-	-	(20)	-	-	-	Miss conflict
LD 22	(16)	-	-	-	(20)	-	(22)	-	Miss compulsory
LD 24	(24)	-	-	-	(20)	-	(22)	-	Miss compulsory
LD 26	(24)		(26)		(20)	-	(22)	-	Miss compulsory
LD 30	(24)	-	(26)	-	(20)	-	(3)	-	Miss compulsory
LD 32	(32)	-	(26)	-	(20)	-	(30)	-	Miss compulsory
ST 16	(16)	-	(26)	-	(20)	-	(30)	-	Miss conflict
ST 0	(0)	-	(26)	-	(20)	-	(30)	-	Miss conflict
LD 14	(0)	-	(26)	-	(20)	-	(14)	-	Miss compulsory

Miss rate = 16/17

b) Adresa AMP din memoria principală dată de procesor este:

AMP = adresa AGP a grupului din MP_ adresa ABLP a blocului din MP_adresa setului ASET din MP_ block offset BLO_byte offset BO (AMP = AGP_ABLP_ASET_BLO_BO)

unde: AGP reprezintă adresa unui grup din MP de unde se cere accesul (fiecare grup din MP are capacitatea C a MC)

ASET adresa setului din MP (aceeași ca la MC)

ABLP reprezintă adresa, nu neapărat aceeași ca a blocului din MC, a blocului din MP (de aceeași lungime ca a blocurilor din MC), bloc adus complet asociativ pe o cale din același set din MC (maparea complet asociativă în cadrul fiecărui set) corespunzător setului din MP cerut de procesor (mapare directă la nivel de seturi).

BLO reprezintă numărul de cuvinte dintr-un bloc (lungimea LC în cuvinte a fiecărui bloc

BO reprezintă numărul de octeți dintr-un cuvânt (adresa unuia dintre cei 4 octeți ai unui cuvânt)

BLO + BO = lungimea LO în octeți a fiecărui bloc (se mai numește și deplasamentul în cadrul blocului

Eticheta Tag = AGP + ABLP

Numărul de seturi se calculează ca mai sus.

Adresa AMC din memoria cache este:

AMC = ASET_ABLC_BLO_BO (MC are același număr de seturi și același număr de blocuri din set ca și MP)

unde: ASET, BLO și BO sunt aceleași ca la AMP

ABLC reprezintă adresa blocului din MC (a unei căi dintr-un set) unde există sau unde se aduce blocul din MP cu adresa ABLP cerută de procesor (adresa unde se aduce nu e neapărat aceeași cu cea de unde a fost înlocuit prin LRU sau alte metode dacă mai fusese în MC)

b1) m = 2, de rezolvat conform exemplului de la b2)

…

b2) m = 4

S = 8/4 = 2 seturi pe 4 căi

0 mod 2 = 0 (calea 0 care conține un întreg bloc)

8 mod 2 = 0

16 mod 2 = 0 … toate dau 0 că sunt pare

	Adresa Cale:	000	001	010	011	Hit sau Miss
LD 0	Set 1					
	Set 0	(0)				Miss compulsory
LD 8	Set 1					
	Set 0	(0)	(8)			Miss compulsory
LD 16	Set 1					
	Set 0	(0)	(8)	(16)		Miss compulsory
ST 0	Set 1					
	Set 0	(0)	(8)	(16)		Hit
LD 12	Set 1					
	Set 0	(0)	(8)	(16)	(12)	Miss compulsory

LD 8	Set 1					
	Set 0	(0)	(8)	(16)	(12)	Hit
ST 20	Set 1					
	Set 0	(0)	(8)	(20)	(12)	Miss compulsory
LD 8	Set 1					
	Set 0	(0)	(8)	(20)	(12)	Hit
ST 16	Set 1					
	Set 0	(16)	(8)	(20)	(12)	Miss conflict
LD 22	Set 1					
	Set 0	(16)	(8)	(20)	(22)	Miss compulsory
LD 24	Set 1					
	Set 0	(16)	(8)	(24)	(22)	Miss compulsory
LD 26	Set 1					
	Set 0	(16)	(26)	(24)	(22)	Miss compulsory
LD 30	Set 1					
	Set 0	(30)	(26)	(24)	(22)	Miss conflict
LD 32	Set 1					
	Set 0	(30)	(26)	(24)	(32)	Miss compulsory
ST 16	Set 1					
	Set 0	(30)	(26)	(16)	(32)	Miss conflict
ST 0	Set 1					
	Set 0	(30)	(0)	(16)	(32)	Miss conflict
LD 14	Set 1					
	Set 0	(14)	(0)	(16)	(32)	Miss compulsory

Miss rate = 14/17, mai bună ca la a)

c) … pentru acasă
   AMP = ABLP_BLO_BO și AMC = ABLC_BLO_BO unde eticheta Tag = ABLP
d) … pentru acasă
e) … pentru acasă

b2a) Rezolvare b2) în cazul unor cereri de acces la memorie cu adresa în hexazecimal (gen examen) - cazurile a și c pentru acasă

Dacă adresele de cereri de acces sunt date în hexazecimal ca adrese de octet sau semi-cuvânt sau cuvânt, atunci, de exemplu pentru maparea set-asociativă pe 4 căi avem:

m = 4 => S = 8/4 = 2 seturi pe 4 căi (blocuri), ca mai sus

AMP[31:0] = B7hB6hB5hB4hB3hB2hB1hB0h unde B este octetul (Byte) din AMP în hexazecimal

Vom detalia acum adresele procesor pe biți punând în evidență ultimii 3 biți din dreapta care reprezintă offset-ul blocului de 1 bit și offset-ul octetului de 2 biți (sunt 2 cuvinte cu câte 4 octeți fiecare în oricare bloc), bitul anterior celor 3 biți din dreapta reprezintă setul de 4 căi (blocuri de 8 octeți fiecare) iar toți biții din stânga celor 4 biți din dreapta reprezintă câmpul Tag al din memoria principală (MP).

Pentru calculul setului din MC în care se plasează sau se găsește blocul din MP cerut de procesor, se face modulo 4 cu câmpul AGP + ABLP +ASET din adresa procesor AMP iar acel câmp se va plasa sau accesa într-o/dintr-o cale a setului calculat ținând cont de algoritmul de înlocuire atunci când este cazul și de faptul că pe o cale se plasează întotdeauna 8 octeți (0-7h, 8-Fh, …) specificând doar Tag-ul blocului din MP.

00h = 0…0_00_0_0_00b (adresa blocului 0 din MP) => 0…0_00_0 = 0 mod 2 = set 0 (octetul 0 din cuvântul 0 din blocul 0 din setul 0 din grupul 0 al memoriei principale)

08h = 0…0_00_1_0_00b (adresa blocului 1 din MP) => 0…0_00_1 = 1 mod 2 = set 1 (octetul 0 din cuvântul 0 din blocul 0 din setul 1 din grupul 0 al memoriei principale)

12h = 0…0_01_0_0_10b (adresa blocului 2 din MP) => 0…0_01_0 = 2 mod 2 = set 0 (octetul 2 din cuvântul 0 din blocul 1 din setul 0 din grupul 0 al memoriei principale)

14h = 0…0_01_0_1_00b (adresa blocului 2 din MP) => 0…0_01_0 = 2 mod 2 = set 0 (octetul 0 din cuvântul 1 din blocul 1 din setul 0 din grupul 0 al memoriei principale)

16h = 0…0_01_0_1_10b (adresa blocului 2 din MP) => 0…0_01_0 = 2 mod 2 = set 0 (octetul 2 din cuvântul 1 din blocul 1 din setul 0 din grupul 0 al memoriei principale)

20h = 0…0_10_0_0_00b (adresa blocului 2 din MP) => 0…0_10_0 = 4 mod 2 = set 0 (octetul 0 din cuvântul 0 din blocul 2 din setul 0 din grupul 0 al memoriei principale)

22h = 0…0_10_0_0_10b (adresa blocului 2 din MP) => 0…0_10_0 = 4 mod 2 = set 0 (octetul 2 din cuvântul 0 din blocul 2 din setul 0 din grupul 0 al memoriei principale)

24h = 0…0_10_0_1_00b (adresa blocului 2 din MP) => 0…0_10_0 = 4 mod 2 = set 0 (octetul 0 din cuvântul 1 din blocul 2 din setul 0 din grupul 0 al memoriei principale)

26h = 0…0_10_0_1_10b (adresa blocului 2 din MP) => 0…0_10_0 = 4 mod 2 = set 0 (octetul 2 din cuvântul 1 din blocul 2 din setul 0 din grupul 0 al memoriei principale)

30h = 0…0_11_0_0_00b (adresa blocului 2 din MP) => 0…0_11_0 = 6 mod 2 = set 0 (octetul 0 din cuvântul 0 din blocul 3 din setul 0 din grupul 0 al memoriei principale)

32h = 0…0_11_0_0_10b (adresa blocului 2 din MP) => 0…0_11_0 = 6 mod 2 = set 0 (octetul 2 din cuvântul 0 din blocul 3 din setul 0 din grupul 0 al memoriei principale)

Set 0				**Set 1**				**Adresă**	**Tip miss**
Cale 0	**Cale 1**	**Cale 2**	**Cale 3**	**Cale 0**	**Cale 1**	**Cale 2**	**Cale 3**	**procesor**	**sau hit**
(0…00h,	-	-	-)	(-	-	-	-)	LD 00h	Compulsory miss
(0…00h,	-	-	-)	(0…01h,	-	-	-)	LD 08h	Compulsory miss
(0…00h,	0…02h,	-	-)	(0…01h,	-	-	-)	LD 16h	Compulsory miss
(0…00h,	0…02h,	-	-)	(0…01h,	-	-	-)	ST 00h	Hit (0…00h)
(0…00h,	0…02h,	-	-)	(0…01h,	-	-	-)	LD 12h	Hit (0…02h)
(0…00h,	0…02h,	-	-)	(0…01h,	-	-	-)	LD 08h	Hit (0…01h)
(0…00h,	0…02h,	0…04h,	-)	(0…01h,	-	-	-)	ST 20h	Compulsory miss
(0…00h,	0…02h,	0…04h,	-)	(0…01h,	-	-	-)	LD 08h	Hit (0…01h)
(0…00h,	0…02h,	0…04h,	-)	(0…01h,	-	-	-)	ST 16h	Hit (0…02h)
(0…00h,	0…02h,	0…04h,	-)	(0…01h,	-	-	-)	LD 22h	Hit (0…04h)
(0…00h,	0…02h,	0…04h,	-)	(0…01h,	-	-	-)	LD 24h	Hit (0…04h)
(0…00h,	0…02h,	0…04h,	-)	(0…01h,	-	-	-)	LD 26h	Hit (0…04h)
(0…00h,	0…02h,	0…04h,	0…06h)	(0…01h,	-	-	-)	LD 30h	Compulsory miss
(0…00h,	0…02h,	0…04h,	0…06h)	(0…01h,	-	-	-)	LD 32h	Hit (0…06h)
(0…00h,	0…02h,	0…04h,	0…06h)	(0…01h,	-	-	-)	ST 16h	Hit (0…02h)
(0…00h,	0…02h,	0…04h,	0…06h)	(0…01h,	-	-	-)	ST 00h	Hit (0…00h)
(0…00h,	0…02h,	0…04h,	0…06h)	(0…01h,	-	-	-)	LD 14h	Hit (0…02h)

Miss rate = 5/17, mai bună ca la b2)