3η Άσκηση στην Αρχιτεκτονική Υπολογιστών

Στεφανάκης Γεώργιος (el18436)

Ακ. έτος 2020-2021, 5ο Εξάμηνο, Σχολή ΗΜΜΥ

Μέρος Α

Γνωρίζουμε από την εκφώνηση ότι ο μέσος χρόνος πρόσβασης στη μνήμη για το πρώτο επίπεδο κρυφής μνήμης είναι $AMAT_{L1}$ = 3,5 κύκλοι και ότι υπάρχει 98% hit rate με 1 κύκλο per hit. Συνεπώς το miss rate του πρώτου επιπέδου είναι 2%. Επιπλέον ισχύει $\mathrm{AMAT_{L1}} = 1 \cdot \mathrm{hit_{L1}} + \mathrm{MR_{L1}} \cdot \mathrm{MP_{L1}} = 3{,}5 \Rightarrow \mathrm{MP_{L1}} = \frac{3{,}5-1}{\frac{2}{100}} 125$ CC. Το miss penalty της L1 που βρήκαμε είναι ίσο με εκείνο της L2 καθώς αντιστοιχεί στο τελευταίο επίπεδο κρυφής μνήμης ανεξαρτήτως αριθμού επιπέδων. Από την εκφώνηση δίνεται το hit rate του δεύτερου επιπέδου ίσο με 88,8% δηλαδή το miss rate είναι ίσο με 11,2%. Συνεπώς έχουμε:

$$\frac{\mathrm{AMAT_{L1}}}{\mathrm{AMAT_{L1+L2}}} \geq 2{,}5 \Rightarrow$$

$$\frac{3{,}5}{1 \cdot \mathrm{hit_{L1}} + \mathrm{MR_{L1}} \cdot (\mathrm{hit_{L2}} + \mathrm{MR_{L2}} \cdot \mathrm{MP_{L2}})} \geq 2{,}5 \Rightarrow$$

$$\frac{3{,}5}{1 + 0{,}02 \cdot (\mathrm{hit_{L2}} + 0{,}112 \cdot 125)} \geq 2{,}5 \Rightarrow$$

$$\mathrm{hit_{L2}} \leq 6 \text{ CC}$$

Δηλαδή η μέγιστη τιμή του hit_{L2} είναι 6 CC, άρα το μέγιστο κόστος πρόσβασης στην L2 είναι hit_{L1} + hit_{L2} = 7 CC.

Μέρος Β

Α) Οι πίνακες Α και Β έχουν ο καθένας από $16 \cdot 8 = 128$ στοιχεία και άρα μέγεθος $128 \cdot 8 = 104$ bytes, διπλάσιο της cache. Το block size έχει μέγεθος 32 bytes άρα χωράει 4 διαδοχικά στοιχεία συνολικά, συνεπώς το block offset θα είναι 5 bits. Επίσης, η cache έχει μέγεθος 512 bytes και διαιρώντας με το μέγεθος του ενός block προκύπτει ο συνολικός αριθμός $\frac{512}{32} = 16$ blocks. Επειδή έχουμε 2 ways ο αριθμός των sets θα είναι $\frac{16}{2} = 8$ sets. Δηλαδή το index είναι $\log_2 8 = 3$ bits. Στην κύρια μνήμη είναι αποθηκευμένος πρώτος ο πίνακας Α και μετά ο πίνακας Β. Συμπεραίνουμε λοιπόν ότι κάθε τετράδα στοιχείων πχ Α00, Α01, Α02, Α03 θα έχουν το ίδιο index αφού στην αρχή στο στοιχείο Α00 το block offset θα δείχνει στο τελευταίο byte (8ο byte ξεκινώντας από το 0 μέχρι το 7) του στοιχείου αυτού δηλαδή θα έχει τιμή 00111 = 7. Το τελευταίο byte του στοιχείου Α03 θα έχει block offset 11111 = 31 συμπληρώνοντας τα 32 bytes που είναι το block size έχοντας ξεκινήσει από το byte 0 μέχρι το byte 31. Έπειτα, θα αυξηθεί το index κατά 1 bit πηγαίνοντας μας σε νέο set. Επειδή το μέγεθος των πινάκων είναι διπλάσιο της cache καταλαβαίνουμε ότι παίρνοντας τα στοιχεία με την σειρά πρώτα θα χωρέσει το ¼ του πίνακα Α στο 1ο way και στη συνέχεια το υπόλοιπο ¼ του Α στο 2ο way. Η ίδια διαδικασία θα γίνει και για τον υπόλοιπο μισό πίνακα Α. Έπειτα, τα ήδη υπάρχοντα στοιχεία του Α στο 1ο way θα αρχίσουν να αντικαθίστανται από τα πρώτα στοιχεία του Β και θα ακολουθηθεί η ίδια διαδικασία με προηγουμένως. Από τα παραπάνω συμπεραίνουμε ότι τα στοιχεία Α00, Α01, Α02, Α03 και Β00, Β01, Β02, Β03 θα μπαίνουν στο ίδιο set και άρα θα έχουν το ίδιο index. Επίσης, τα Α00, Α01, Α02, Α03 θα είναι στο ίδιο set με τα Α40, Α41, Α42,

A43 και τα A04, A05, A06, A07 στο ίδιο set με τα A44, A45, A46, A47. Δηλαδή ανά 4 γραμμές τα 4 πρώτα από τα 8 στοιχεία της κάθε γραμμής μπαίνουν στο ίδιο set και τα άλλα 4 στοιχεία μπαίνουν επίσης σε ίδιο set. Άρα και οι τετράδες που ξεκινάνε από τα στοιχεία A[0][0], A[4][0], A[8][0], A[12][0] θα μπαίνουν στο ίδιο set και συγκεκριμένα στο set 0 που είναι το πρώτο. Τα ίδια ισχύουν και για τον πίνακα B. Σύμφωνα με τα παραπάνω και ξεκινώντας να γεμίζουμε πρώτο το way 0 έχουμε από την αρχή του κώδικα:

Για i=0, j=0:

Read A[4][0]: m (compulsory)

Read A[0][0]: m (compulsory)

Read B[0][0]: m (compulsory)

Write A[4][0]: m (conflict)

Η cache στο τέλος θα έχει την παρακάτω μορφή:

SETS	LRU	WAY 0	WAY 1
0	0	**B[0][0] B[0][1] B[0][2] B[0][3]**	**A[4][0] A[4][1] A[4][2] A[4][3]**
1			
2			
3			
4			
5			
6			
7			

(i = 0, j = 1)

SETS	LRU	WAY 0	WAY 1
0	0	**A[4][0] A[4][1] A[4][2] A[4][3]**	**B[0][0] B[0][1] B[0][2] B[0][3]**
1			
2			
3			
4			
5			
6			
7			

(i = 0, j = 2)

Όμοια για j = 3. Άρα για κάθε τετράδα j έχουμε 3 hits και 13 misses. Στο τέλος κάθε τετράδας j το αντίστοιχο set θα έχει στο way 0 την τετράδα A[4+j] και στο way 1 την τετράδα B[i][j] και το LRU εναλλάσσεται άρα θα έχει την τιμή 1. Αυτά θα ισχύουν για τα πρώτα i (0 έως 3). Για κάθε τετράδα j οι τετράδες των πινάκων A, B θα έχουν το ίδιο index και άρα θα μπαίνουν στα ίδια sets. Μόλις αλλάζει η τετράδα των j πάμε σε νέο set και έτσι εν τέλει για τα i από 0 έως 3 η cache θα γεμίσει και θα έχει την μορφή:

SETS	LRU	WAY 0	WAY 1
0	1	**A[4][0] A[4][1] A[4][2] A[4][3]**	**B[0][0] B[0][1] B[0][2] B[0][3]**
1	1	**A[4][4] A[4][5] A[4][6] A[4][7]**	**B[0][4] B[0][5] B[0][6] B[0][7]**
2	1	**A[5][0] A[5][1] A[5][2] A[5][3]**	**B[1][0] B[1][1] B[1][2] B[1][3]**
3	1	**A[5][4] A[5][5] A[5][6] A[5][7]**	**B[1][4] B[1][5] B[1][6] B[1][7]**
4	1	**A[6][0] A[6][1] A[6][2] A[6][3]**	**B[2][0] B[2][1] B[2][2] B[2][3]**
5	1	**A[6][4] A[6][5] A[6][6] A[6][7]**	**B[2][4] B[2][5] B[2][6] B[2][7]**
6	1	**A[7][0] A[7][1] A[7][2] A[7][3]**	**B[3][0] B[3][1] B[3][2] B[3][3]**
7	1	**A[7][4] A[7][5] A[7][6] A[7][7]**	**B[3][4] B[3][5] B[3][6] B[3][7]**

Άρα για τα 4 πρώτα i έχουμε 8·3 = 24 hits και 8·13 = 104 misses

Για i = 4, j = 0:

A[4][0] h

A[5][0] h

B[4][0] m (comp)

A[4][0] h

Για i = 4, j = 1:

A[4][1] h

A[5][1] h

B[4][1] h

A[4][1] h

Το ίδια θα συνεχιστεί για j = 2 και j = 3

Άρα τώρα για κάθε τετράδα j θα έχω 15 hits και 1 miss. Ύστερα από τα 4 πρώτα j η cache θα έχει την εξής μορφή:

SETS	LRU	WAY 0	WAY 1
0	1	**A[4][0] A[4][1] A[4][2] A[4][3]**	**B[4][0] B[4][1] B[4][2] B[4][3]**
1	1	**A[4][4] A[4][5] A[4][6] A[4][7]**	**B[0][4] B[0][5] B[0][6] B[0][7]**
2	1	**A[5][0] A[5][1] A[5][2] A[5][3]**	**B[1][0] B[1][1] B[1][2] B[1][3]**
3	1	**A[5][4] A[5][5] A[5][6] A[5][7]**	**B[1][4] B[1][5] B[1][6] B[1][7]**
4	1	**A[6][0] A[6][1] A[6][2] A[6][3]**	**B[2][0] B[2][1] B[2][2] B[2][3]**
5	1	**A[6][4] A[6][5] A[6][6] A[6][7]**	**B[2][4] B[2][5] B[2][6] B[2][7]**
6	1	**A[7][0] A[7][1] A[7][2] A[7][3]**	**B[3][0] B[3][1] B[3][2] B[3][3]**
7	1	**A[7][4] A[7][5] A[7][6] A[7][7]**	**B[3][4] B[3][5] B[3][6] B[3][7]**

Στην επόμενη 4άδα από j = 4 μέχρι j = 7 έχουμε:

SETS	LRU	WAY 0	WAY 1
0	1	**A[4][0] A[4][1] A[4][2] A[4][3]**	**B[4][0] B[4][1] B[4][2] B[4][3]**
1	1	**A[4][4] A[4][5] A[4][6] A[4][7]**	**B[4][4] B[4][5] B[4][6] B[4][7]**
2	1	**A[5][0] A[5][1] A[5][2] A[5][3]**	**B[1][0] B[1][1] B[1][2] B[1][3]**
3	1	**A[5][4] A[5][5] A[5][6] A[5][7]**	**B[1][4] B[1][5] B[1][6] B[1][7]**
4	1	**A[6][0] A[6][1] A[6][2] A[6][3]**	**B[2][0] B[2][1] B[2][2] B[2][3]**
5	1	**A[6][4] A[6][5] A[6][6] A[6][7]**	**B[2][4] B[2][5] B[2][6] B[2][7]**
6	1	**A[7][0] A[7][1] A[7][2] A[7][3]**	**B[3][0] B[3][1] B[3][2] B[3][3]**
7	1	**A[7][4] A[7][5] A[7][6] A[7][7]**	**B[3][4] B[3][5] B[3][6] B[3][7]**

Συνεχίζοντας την επανάληψη μέχρι και i = 6 θα πάρουμε:

SETS	LRU	WAY 0	WAY 1
0	1	**A[4][0] A[4][1] A[4][2] A[4][3]**	**B[4][0] B[4][1] B[4][2] B[4][3]**
1	1	**A[4][4] A[4][5] A[4][6] A[4][7]**	**B[4][4] B[4][5] B[4][6] B[4][7]**
2	1	**A[5][0] A[5][1] A[5][2] A[5][3]**	**B[5][0] B[5][1] B[5][2] B[5][3]**
3	1	**A[5][4] A[5][5] A[5][6] A[5][7]**	**B[5][4] B[5][5] B[5][6] B[5][7]**
4	1	**A[6][0] A[6][1] A[6][2] A[6][3]**	**B[6][0] B[6][1] B[6][2] B[6][3]**
5	1	**A[6][4] A[6][5] A[6][6] A[6][7]**	**B[6][4] B[6][5] B[6][6] B[6][7]**
6	1	**A[7][0] A[7][1] A[7][2] A[7][3]**	**B[3][0] B[3][1] B[3][2] B[3][3]**
7	1	**A[7][4] A[7][5] A[7][6] A[7][7]**	**B[3][4] B[3][5] B[3][6] B[3][7]**

Μέχρι εδώ έχουμε $3 \cdot 2 = 6$ misses και $30 \cdot 3 = 90$ hits.

Για i = 7, j = 0:

A[7][0] h

A[8][0] m

B[7][0] m

A[7][0] h

Για i = 7, j = 1:

A[7][1] h

A[8][1] h

B[7][1] h

A[7][1] h

Επομένως για μία τετράδα j και i = 7, έχουμε 2 misses και 14 hits. Η cache θα έχει την παρακάτω μορφή:

SETS	LRU	WAY 0	WAY 1
0	0	**A[4][0] A[4][1] A[4][2] A[4][3]**	**A[8][0] A[8][1] A[8][2] A[8][3]**
1	1	**A[4][4] A[4][5] A[4][6] A[4][7]**	**B[4][4] B[4][5] B[4][6] B[4][7]**
2	1	**A[5][0] A[5][1] A[5][2] A[5][3]**	**B[5][0] B[5][1] B[5][2] B[5][3]**
3	1	**A[5][4] A[5][5] A[5][6] A[5][7]**	**B[5][4] B[5][5] B[5][6] B[5][7]**
4	1	**A[6][0] A[6][1] A[6][2] A[6][3]**	**B[6][0] B[6][1] B[6][2] B[6][3]**
5	1	**A[6][4] A[6][5] A[6][6] A[6][7]**	**B[6][4] B[6][5] B[6][6] B[6][7]**
6	1	**A[7][0] A[7][1] A[7][2] A[7][3]**	**B[7][0] B[7][1] B[7][2] B[7][3]**
7	1	**A[7][4] A[7][5] A[7][6] A[7][7]**	**B[3][4] B[3][5] B[3][6] B[3][7]**

Για την 2η τετράδα από j = 4 έως j = 7 η cache θα έχει την μορφή:

SETS	LRU	WAY 0	WAY 1
0	0	**A[4][0] A[4][1] A[4][2] A[4][3]**	**A[8][0] A[8][1] A[8][2] A[8][3]**
1	0	**A[4][4] A[4][5] A[4][6] A[4][7]**	**A[8][4] A[8][5] A[8][6] A[8][7]**
2	1	**A[5][0] A[5][1] A[5][2] A[5][3]**	**B[5][0] B[5][1] B[5][2] B[5][3]**
3	1	**A[5][4] A[5][5] A[5][6] A[5][7]**	**B[5][4] B[5][5] B[5][6] B[5][7]**
4	1	**A[6][0] A[6][1] A[6][2] A[6][3]**	**B[6][0] B[6][1] B[6][2] B[6][3]**
5	1	**A[6][4] A[6][5] A[6][6] A[6][7]**	**B[6][4] B[6][5] B[6][6] B[6][7]**
6	1	**A[7][0] A[7][1] A[7][2] A[7][3]**	**B[7][0] B[7][1] B[7][2] B[7][3]**
7	1	**A[7][4] A[7][5] A[7][6] A[7][7]**	**B[7][4] B[7][5] B[7][6] B[7][7]**

Για i = 7 έχουμε $2 \cdot 2 = 4$ misses και $2 \cdot 14 = 28$ hits, συνολικά $4 + 6 + 104 = 114$ misses και $28 + 90 + 24 = 142$ hits.

Β) Τα block offset και block size παραμένουν ίδια. Με την 4 way cache αυτό που θα αλλάξει είναι ο αριθμός των sets τα οποία θα γίνουν $\frac{16}{4} = 4$ και άρα το index ίσο με 2 bits. Με την ίδια λογική με προηγουμένως τα πρώτα 16 στοιχεία του Α θα γεμίσουν το πρώτο way μέχρι το 4ο way. Μόλις γεμίσει το 4ο way θα έχει γεμίσει η cache με τον μισό πίνακα Α. Με την ίδια λογική τώρα θα αντικατασταθούν τα ήδη υπάρχοντα στοιχεία της cache με τα υπόλοιπα στοιχεία του Α. Όταν ολοκληρωθεί αυτή η διαδικασία θα αρχίσουμε να φέρνουμε στοιχεία από τον πίνακα Β και η πρώτη 4αδα που θα φέρουμε θα είναι η B[0][0] B[0][1] B[0][2] B[0][3] η οποία θα πάει στο 1ο set και γεμίζουμε με τον ίδιο τρόπο όπως και με τον πίνακα Α.

Για i = 0, j = 0:

A[4][0] m (comp)

A[0][0] m (comp)

B[0][0] m (comp)

A[4][0] h

Για i = 0, j = 1:

A[4][1] h

A[0][1] h

B[0][1] h

A[4][1] h

Το ίδιο θα συνεχιστεί για j = 2, 3. Για μία τετράδα j έχουμε 3 misses και 13 hits. Είναι φανερό ότι μειώνονται σημαντικά τα conflict misses. Στην επόμενη τετράδα των j θα γίνει το ίδιο στο set 1. Τα ίδια και για i = 1 όπου εκεί θα γεμίσουν τα set 2, 3. Τελικά για i από 0 έως και 1 έχουμε 12 misses και 52 hits. Παρακάτω φαίνεται η μορφή της cache αφού τρέξουμε τον κώδικα για i =0 και j = 0:

SETS	LRU	WAY-0	WAY-1	WAY-2	WAY-3
0		A[4][0] A[4][1] A[4][2] A[4][3]	A[0][0] A[0][1] A[0][2] A[0][3]	B[0][0] B[0][1] B[0][2] B[0][3]	
1					
2					
3					

Γεμίζοντας με αυτήν την λογική τον πίνακα μέχρι και i = 1 έχουμε:

SETS	LRU	WAY-0	WAY-1	WAY-2	WAY-3
0		A[4][0] A[4][1] A[4][2] A[4][3]	A[0][0] A[0][1] A[0][2] A[0][3]	B[0][0] B[0][1] B[0][2] B[0][3]	
1		A[4][4] A[4][5] A[4][6] A[4][7]	A[0][4] A[0][5] A[0][6] A[0][7]	B[0][4] B[0][5] B[0][6] B[0][7]	
2		A[5][0] A[5][1] A[5][2] A[5][3]	A[1][0] A[1][1] A[1][2] A[1][3]	B[1][0] B[1][1] B[1][2] B[1][3]	
3		A[5][4] A[5][5] A[5][6] A[5][7]	A[1][4] A[1][5] A[1][6] A[1][7]	B[1][4] B[1][5] B[1][6] B[1][7]	

Για i = 2:

A[6][0] m (LRU 01) (compulsory)

A[2][0] m (LRU 10) (compulsory)

B[2][0] m (LRU 00) (compulsory)

A[6][0] h

(Άρα κάθε τετράδα θα τελειώνει με LRU = 00)

i = 2, j = 1:

A[6][1] h

A[2][1] h

B[2][1] h

A[6][1] h

Το ίδιο θα συνεχιστεί για j = 2, 3.Επομένως για την μία τετράδα j έχουμε ξανά 3 misses 13 hits. Για την επόμενη τετράδα j (4-7) θα ακολουθηθεί η ίδια διαδικασία στο set 1. Ακόμα, η ίδια διαδικασία θα ακολουθηθεί και για το i = 3 στα sets 2 και 3.Τελικά για i = 2 και i = 3 μαζί θα έχουμε 12 misses 52 hits. Τελικά για i < 4 , έχουμε 24 misses 104 hits. Η μορφή της cache όταν θα έχει τελείωσει και το i = 3 φαίνεται παρακάτω:

SETS	LRU	WAY-0	WAY-1	WAY-2	WAY-3
0	00	A[4][0] A[4][1] A[4][2] A[4][3]	A[2][0] A[2][1] A[2][2] A[2][3]	B[2][0] B[2][1] B[2][2] B[2][3]	A[6][0] A[6][1] A[6][2] A[6][3]
1	00	A[4][4] A[4][5] A[4][6] A[4][7]	A[2][4] A[2][5] A[2][6] A[2][7]	B[2][4] B[2][5] B[2][6] B[2][7]	A[6][4] A[6][5] A[6][6] A[6][7]
2	00	A[5][0] A[5][1] A[5][2] A[5][3]	A[3][0] A[3][1] A[3][2] A[3][3]	B[3][0] B[3][1] B[3][2] B[3][3]	A[7][0] A[7][1] A[7][2] A[7][3]
3	00	A[5][4] A[5][5] A[5][6] A[5][7]	A[3][4] A[3][5] A[3][6] A[3][7]	B[3][4] B[3][5] B[3][6] B[3][7]	A[7][4] A[7][5] A[7][6] A[7][7]

Για i = 4, j = 0:

A[4][0] h (LRU 01)

A[5][0] h (LRU 01)

B[4][0] m (LRU 10) (comp)

A[4][0] h

(Άρα κάθε τετράδα θα τελειώνει με LRU = 10)

Για i = 4, j = 1:

A[4][1] h

A[5][1] h

B[4][1] h

A[4][1] h

Το ίδιο επαναλαμβάνεται για j = 2, 3. Επομένως για μία τετράδα j έχουμε συνολικά 1 misses 15 hits. Γενικεύουμε αυτή τη διαδικασία για i = 4 και i = 5. Τελικά έχουμε 4 misses 60 hits για i = 4 και i = 5. Παρακάτω φαίνεται η cache μόλις τελειώσει και το i = 5:

SETS	LRU	WAY-0	WAY-1	WAY-2	WAY-3
0	10	A[4][0] A[4][1] A[4][2] A[4][3]	B[4][0] B[4][1] B[4][2] B[4][3]	B[2][0] B[2][1] B[2][2] B[2][3]	A[6][0] A[6][1] A[6][2] A[6][3]
1	10	A[4][4] A[4][5] A[4][6] A[4][7]	B[4][4] B[4][5] B[4][6] B[4][7]	B[2][4] B[2][5] B[2][6] B[2][7]	A[6][4] A[6][5] A[6][6] A[6][7]
2	10	A[5][0] A[5][1] A[5][2] A[5][3]	B[5][0] B[5][1] B[5][2] B[5][3]	B[3][0] B[3][1] B[3][2] B[3][3]	A[7][0] A[7][1] A[7][2] A[7][3]
3	10	A[5][4] A[5][5] A[5][6] A[5][7]	B[5][4] B[4][5] B[5][6] B[5][7]	B[3][4] B[3][5] B[3][6] B[3][7]	A[7][4] A[7][5] A[7][6] A[7][7]

Για i = 6 και j = 0:

A[6][0] h

A[7][0] h

B[6][0] m (LRU 01) (comp)

A[6][0] h

Για i = 6 και j = 1:

A[6][0] h

A[7][0] h

B[6][0] h

A[6][0] h

Όμοια και για j = 2, 3. Όμοια και για j = 4, 5, 6, 7 στο set 2. Άρα για μια τετράδα j έχουμε 1 miss 15 hits. Συνολικά για i = 6 έχουμε 2 misses 30 hits. Παρακάτω φαίνεται η cache όταν θα έχει τελείωσει και το i = 6:

SETS	LRU	WAY-0	WAY-1	WAY-2	WAY-3
0	01	A[4][0] A[4][1] A[4][2] A[4][3]	B[4][0] B[4][1] B[4][2] B[4][3]	B[6][0] B[6][1] B[6][2] B[6][3]	A[6][0] A[6][1] A[6][2] A[6][3]
1	01	A[4][4] A[4][5] A[4][6] A[4][7]	B[4][4] B[4][5] B[4][6] B[4][7]	B[6][4] B[6][5] B[6][6] B[6][7]	A[6][4] A[6][5] A[6][6] A[6][7]
2	10	A[5][0] A[5][1] A[5][2] A[5][3]	B[5][0] B[5][1] B[5][2] B[5][3]	B[3][0] B[3][1] B[3][2] B[3][3]	A[7][0] A[7][1] A[7][2] A[7][3]
3	10	A[5][4] A[5][5] A[5][6] A[5][7]	B[5][0] B[5][1] B[5][2] B[5][3]	B[3][4] B[3][5] B[3][6] B[3][7]	A[7][4] A[7][5] A[7][6] A[7][7]

Για i = 7 και j = 0:

A[7][0] hit

A[8][0] m (LRU 00) (comp)

B[7][0] m (LRU 01) (comp)

A[7][0]

Για i = 7 και j = 1:

A[7][0] hit

A[8][0] hit

B[7][0] hit

A[7][0] hit

Ομοίως και για j = 2, 3. Όλη η παραπάνω διαδικασία για i = 7 και j = 0, 1, 2, 3 επαναλαμβάνεται και για i = 7 και j = 4, 5, 6, 7 στο set 1.

Επομένως για i = 7 έχουμε συνολικά $2 \cdot 2 = 4$ misses και $2 \cdot 14 = 28$ hits γιατί έχουμε 2 τετράδες από j. Η μορφή της cache αφού τελειώσει και το i = 7 θα είναι:

SETS	LRU	WAY-0	WAY-1	WAY-2	WAY-3
0	00	A[4][0] A[4][1] A[4][2] A[4][3]	A[8][0] A[8][1] A[8][2] A[8][3]	B[6][0] B[6][1] B[6][2] B[6][3]	A[6][0] A[6][1] A[6][2] A[6][3]
1	00	A[4][4] A[4][5] A[4][6] A[4][7]	A[8][4] A[8][5] A[8][6] A[8][7]	B[6][4] B[6][5] B[6][6] B[6][7]	A[6][4] A[6][5] A[6][6] A[6][7]
2	01	A[5][0] A[5][1] A[5][2] A[5][3]	B[5][0] B[5][1] B[5][2] B[5][3]	B[7][0] B[7][1] B[7][2] B[7][3]	A[7][0] A[7][1] A[7][2] A[7][3]
3	01	A[5][4] A[5][5] A[5][6] A[5][7]	B[5][0] B[5][1] B[5][2] B[5][3]	B[7][4] B[7][5] B[7][6] B[7][7]	A[7][4] A[7][5] A[7][6] A[7][7]

Τελικά έχουμε συνολικά 4 + 2 + 4 + 24 = 34 misses και 28 + 30 + 60 + 104 = 222 hits. Είναι φανερό ότι με την 4 way cache μειώθηκαν σε μεγάλο βαθμό τα misses κάτι το οποίο οφείλεται στην εξάλειψη των conflict misses. Πλέον το πρόγραμμα δεν διώχνει από την μνήμη τα δεδομένα που θα χρειαστεί αργότερα αλλά αντίθετα διώχνει τα άχρηστα δεδομένα. Άρα η 4 way Cache είναι πολύ πιο αποδοτική για τον συγκεκριμένο κώδικα από την 2-way.