7ο Εργαστήριο Αρχιτεκτονικής Η/Υ: Κρυφές μνήμες (cache)

Α. Ευθυμίου

Παραδοτέο: Βλ. ecourse 2023, 23:59

Ο σκοπός αυτής της άσκησης είναι η εμβάθυνση της κατανόησης της λειτουργίας των κρυφών μνημών. Θα πρέπει να έχετε μελετήσει τα μαθήματα για το υποσύστημα μνήμης που αντιστοιχούν στις ενότητες 5.1-5.3 και τμήμα του 5.5 του συγγράμματος των Patterson, Hennessy.

Οι ερωτήσεις είναι σχεδιασμένες για να απαντηθούν με τη σειρά που βρίσκονται στο κείμενο γιατί ακολουθούν τη σειρά των «πειραμάτων». Αν δεν μπορείτε να απαντήσετε σε κάποια, συνεχίστε στην επόμενη. Μη δοκιμάσετε όμως να εξετάσετε μια ερώτηση χωρίς τουλάχιστον να προσπαθήσετε την προηγούμενή της.

Αν και παρακάτω υπάρχουν πίνακες, ώστε αν σας βολεύει να δουλεύετε σε χαρτί να μπορείτε να σημειώνετε τις απαντήσεις, οι τελικές απαντήσεις θα πρέπει να ανεβούν στο αποθετήριο στο GitHub σε ένα αρχείο. Δεν υπάρχουν απαιτήσεις για τη μορφη του αρχείου. Χρησιμοποιείστε ό,τι σας είναι βολικό. Με την ολοκλήρωση της άσκησης θα δοθούν λύσεις για να συγκρίνετε με τις δικές απαντήσεις.

1 Προσομοιωτής κρυφής μνήμης του Ripes

Θα χρησιμοποιήσετε τον Ripes και κυρίως το “Cache” tab, την τρίτη από τις επιλογές στα αριστερά, που είναι ένας προσομοιωτής κρυφής μνήμης. Αν και μπορεί να προσομοιώσει και την κρυφή μνήμη εντολών, στην άσκηση θα περιοριστούμε στη κρυφή μνήμη δεδομένων (L1 Data Cache). Μπορεί κανείς να ορίσει τον τρόπο οργάνωσης και τις βασικές παραμέτρους μιας κρυφής μνήμης και να δεί πως γίνονται οι προσπελάσεις στις γραμμές της καθώς και να πάρει πληροφορίες σχετικά με τον αριθμό και το ποσοστό ευστοχιών και άλλων μετρικών επίδοσης κρυφής μνήμης. **Γι᾽αυτή την άσκηση είναι απαραίτητο να ορίσετε τον επεξεργαστή ώστε να μην γίνεται διοχέτευση (single-cycle processor)**. Διαβάστε το https://github.com/mortbopet/Ripes/blob/master/docs/cache_sim.md για πληροφορίες.

Οι παράμετροι οργάνωσης του cache simulator βρίσκονται στο επάνω αριστερά μέρος του παραθύρου. Από κάτω υπάρχουν επιλογές για παρουσίαση γραφημάτων (plot configutation) και ένα πλαίσιο που εμφανίζονται μετρήσεις (ονομάζεται statistics). Στο μεγάλο τμήμα στα δεξιά εμφανίζεται σχηματικά η κρυφή μνήμη. Στον παραπάνω σύνδεσμο περιγράφεται πως εμφανίζονται direct mapped και setassociative οργανώσεις.

Παρατηρήστε ιδιαίτερα τον τρόπο που ορίζονται το πλήθος των γραμμών, η συσχετιστικότητα (αριθμός ways), και το μέγεθος κάθε γραμμής: δίνονται ως ο εκθέτης μιας δύναμης του 2. Δείτε τον παραπάνω σύνδεσμο για λεπτομέρειες. **Προσοχή, αυτό που αναφέρεται ως γραμμές-Lines στο πλαίσιο cache configuration, είναι ο αριθμός των σετ!** Για παράδειγμα, αν βάλετε ως 2^N Lines το 4 και 2^N Ways 1, θα δείτε ότι σχηματίζονται 16 σετ των 2 γραμμών το καθένα. **Επίσης προσέξτε ότι το μέγεθος γραμμής είναι σε λέξεις όχι bytes.**

2 Το πρόγραμμα cache.s

Ακολουθήστε τον σύνδεσμο που δίνεται στο ecourse για να δημιουργηθει το δικό σας αποθετήριο της άσκησης και κλωνοποιήστε το.

Εκεί θα βρείτε το cache.s που εκτελεί τον παρακάτω ψευτοκώδικα.

```
int array[arraySize]; // sizeof(int) == 4 bytes ==
1 word j = repCount; do {
   // Step through the selected array with the given step
   size.
   i = 0;
   do { if
```

```
    (option
    == 0)
            // Option 0: One cache access - write
            array[i] = 0;
        else
            // Option 1: Two cache accesses - read AND write
            array[i] = array[i] + 1;
        i += stepSize; }
    while (i <
    arraySize) j--;
} while (j > 0) { // repeat the inner loop repCount times
```

Στις εκτελέσεις του προγράμματος θα γίνονται αλλαγές στις εξής παραμέτρους[1] του.

arraySize - το μέγεθος του πίνακα array σε λέξεις. Αντιστοιχεί στον καταχωρητή s0.

option - Ελέγχει τί θα κάνει ο εσωτερικός βρόχος: 0 μηδενίζει τα στοιχεία του πίνακα και 1 τα αυξάνει κατά 1. Αντιστοιχεί στον καταχωρητή s1.

stepSize - το βήμα με το οποίο προσπελαύνεται ο πίνακας array. Αντιστοιχεί στον καταχωρητή s2. repCount - ο αριθμός επαναλήψεων του εσωτερικού βρόχου. Αντιστοιχεί στον καταχωρητή s3.

Στην αρχή του τμήματος δεδομένων (.data), υπάρχει το **label padding** που έχει ως σκοπό να αλλάξει την διεύθυνση από όπου αρχίζει ο πίνακας array. Η οδηγία .zero X, που χρησιμοποιείται σε αυτό το label, δεσμεύει και μηδενίζει X bytes μνήμης. Αρχικά είναι μηδέν, αλλά σε κάποια ερωτήματα θα το αλλάξετε σε 4 ή 8. Με αυτό τον τρόπο το array θα ξεκινά από την διεύθυνση 0x10000000, 0x10000004, 0x10000008, και αυτό θα προκαλεί κάποιες διαφορές στην αντιστοίχιση λέξεων του array σε γραμμές της κρυφής μνήμης.

Το σημαντικό στο cache.s είναι ο ψευτοκώδικας, οι παράμετροι και το padding, και το γεγονός ότι υπάρχουν δύο επίπεδα επαναλήψεων (φωλιασμένη επανάληψη). Επίσης προσέξτε ότι όταν το option είναι 1 γίνονται **δύο** προσπελάσεις μνήμης ανά (εσωτερική) επανάληψη: ανάγνωση και εγγραφή στην μνήμη.

2.1 Παρατηρήσεις

Στα παρακάτω πειράματα προσπαθήσετε να κάνετε μια εκτίμηση για τα αποτελέσματα **πριν** τρέξετε την προσομοίωση. Μετά την προσομοίωση, σιγουρευτείτε ότι καταλαβαίνετε **γιατί** βλέπετε ό,τι βλέπετε.

Αναρωτηθείτε τα παρακάτω σε κάθε πείραμα - αλλαγή παραμέτρων προγράμματος, κρυφής μνήμης:

- Ποιό είναι το μέγεθος γραμμής (cache block size); Στον Ripes δίνεται σε λέξεις (Words/Line). Προσοχή μην μπερδεύετε το μέγεθος γραμμής με τον συνολικό αριθμό γραμμών της cache.
- Πόσες συνεχόμενες προσπελάσεις ``χωράνε'' σε μιά γραμμή κρυφής μνήμης; Εδώ πρέπει να πάρετε υπόψη και το stepSize: οι προσπελάσεις μπορεί να μην είναι σε συνεχόμενα στοιχεία του πίνακα, αλλά να έχουν κάποιο βήμα (stride στα Αγγλικά).
- Σε ποιά θέση στην κρυφή μνήμη τοποθετείται μια (ή κάθε) συγκεκριμένη γραμμή;
- Πόσα δεδομένα του προγράμματος χωράνε σε **ολόκληρη** την κρυφή μνήμη;
- Πόσο απέχουν στην μνήμη οι γραμμές που αντιστοιχίζονται στο ίδιο set (και που μπορούν να προκαλέσουν συγκρούσεις);

[1] Προσοχή να μην συγχέονται με τις παραμέτρους της κρυφής μνήμης!

- Όταν εξετάζετε αν μια προσπέλαση θα ευστοχίσει ή όχι, σκεφτείτε αν έχει ξαναγίνει προσπέλαση σε αυτήν την γραμμή στο παρελθόν. Αν έχει ήδη προσπελαστεί, υπάρχει ακόμα στην κρυφή μνήμη, ή έχει αντικατασταθεί;

3 Μέρος Α: Αντιστοίχιση γραμμών σε θέσεις της cache

Φορτώστε το cache.s στον Ripes και ανοίξτε το cache tab από τα εικονίδια στα αριστερά. Οι παράμετροι του cache.s θα πρέπει να είναι: arraySize=32, option=0, stepSize=1, repCount=1. Ο αριθμός μετά το .zero στο label padding θα πρέπει να είναι 0 (η αρχική τιμή). Στο cache configuration, επάνω αριστερά στο tab Cache, επιλέξτε τις κατάλληλες τιμές ώστε να καθορίσετε μια κρυφή μνήμη **direct mapped, με 8 γραμμές, μέγεθος γραμμής (cache block size) 2 λέξεων**.

Σημαντικό. Αν τρέξετε όλο το πρόγραμμα θα δείτε την τελική κατάσταση της κρυφής μνήμης και τα τελικά αποτελέσματα ευστοχιών - αστοχιών. Στις ερωτήσεις θα χρειαστούν και ενδιάμεσα αποτελέσματα. Αντί να τρέχετε τις εντολές μία-μία, είναι καλύτερο να θέσετε breakpoints αμέσως μετά (ή πριν, αν σας βολεύει περισσότερο) από κάθε εντολή προσπέλασης μνήμης για να βλέπετε τις αλλαγές που προκαλεί στην cache κάθε μία από αυτές. Αυτό γίνεται στο Editor tab του Ripes, κάνοντας κλικ στο μπλε πλαίσιο (στήλη) στα αριστερά της εντολής που σας ενδιαφέρει. Η εκτέλεση πλέον θα σταματά πρίν εκτελεστεί η ``σημαδεμένη'' εντολή, όταν τρέχετε το πρόγραμμα με την επιλογή "Execute simulation (F8)..." πατώντας το εικονίδιο που μοιάζει με "»", που κανονικά εκτελεί ολόκληρο το πρόγραμμα. Έχω παρατηρήσει ότι αυτό **δεν συμβαίνει** όταν τρέχει κανείς το πρόγραμμα με το "Clock the circuit with the selected freuency (F6)" (το πράσινο "play" εικονίδιο). Για να προχωρήσει η εκτέλεση μετά το breakpoint, δυστυχώς χρειάζεται μια φορά να γίνει το "Clock the Circuit (F5)", πατώντας το εικονίδιο που μοιάζει με ">". Πατώντας κατευθείαν το "»" δεν προχωράει στο επόμενο breakpoint, αν είναι ήδη σταματημένο σε ένα breakpoint. Επειδή θα χρειαστεί αυτό να γίνει πολλές φορές, χρησιμοποιείστε τις συντμήσεις πληκτρολογίου, F8 και F5.

Εκτελέστε το πρόγραμμα μέχρι την πρώτη προσπέλαση μνήμης και συμπληρώστε τις πληροφορίες που ζητούνται στο αριστερό μισό (με επικεφαλίδα direct mapped) του πίνακα απαντήσεων 1 στην γραμμή για padding 0 και την πρώτη προσπέλαση: Αν η κρυφή μνήμη ευστόχησε (Hit ή Miss), τον αριθμό του block που προσπελάστηκε (0 το πρώτο μέχρι 7 το τελευταίο) και το tag της γραμμής (φαίνεται σε ξεχωριστή στήλη σε κάθε cache block). Για το tag χρησιμοποιείστε μόνο δεκαεξαδικούς αριθμούς.

Ο αριθμός του block που ζητείται είναι ίδιος με το index στην περίπτωση της direct mapped οργάνωσης. Παρακάτω ζητείται ο αριθμός block και για associative cache. Εκεί συμπληρώνετε την θέση όπως φαίνεται στον Ripes, με το πρώτο, από επάνω, να είναι το 0 και το τελευταίο να είναι το 7.

Συνεχίστε την εκτέλεση μέχρι και την δεύτερη προσπέλαση μνήμης και συμπληρώστε ξανά τις αντίστοιχες πληροφορίες στον πίνακα.

Ο σκοπός σας είναι να καταλάβετε σε ποιά γραμμή (cache block) αντιστοιχούν οι λέξεις που προσπελαύνονται, πώς ο προσομοιωτής αποφασίζει σε ποιό block της cache αντιστοιχεί - αποθηκεύεται κάθε γραμμή της μνήμης, πώς υπολογίζεται το tag από την διεύθυνση. Αν δυσκολεύεστε, χρησιμοποιείστε χαρτί και μολύβι, ``σπάστε'' την διεύθυνση στα τμήματα tag:index:offset, αφού πρώτα υπολογίσετε πόσα bits θα πρέπει να είναι το καθένα από αυτά. Ο αριθμός που σχηματίζει το index θα πρέπει να είναι ίδιος με του cache simulator και το tag που βρήκατε θα πρέπει να είναι επίσης το ίδιο. Στο επάνω μέρος του Cache tab ο Ripes δείχνει πως ``σπάει'' η διεύθυνση για την τρέχουσα οργάνωση και δείχνει τις συγκεκριμένες τιμές όταν εκτελείται μια εντολή προσπέλασης μνήμης.

Συνεχίστε μέχρι να αναγνωρίσετε το μοτίβο των προσπελάσεων, αλλά δεν υπάρχουν άλλες ερωτήσεις να συμπληρωθούν γι᾽αυτό το πείραμα.

Αλλάξτε το padding σε 4. Δείτε και καταγράψτε στις αντίστοιχες γραμμές του πίνακα ό,τι ζητείται για τις δύο πρώτες προσπελάσεις του προγράμματος.

Οι διαφορές είναι σχετικά μικρές αλλά θα πρέπει να μπορείτε να καταλαβαίνετε γιατί συμβαίνουν.

Επαναλάβετε με padding 8 αυτή τη φορά και συμπληρώστε τον πίνακα.

Αλλάξτε την οργάνωση σε Fully Associative, με αλγόριθμο αντικατάστασης (Replacement Policy) LRU, και τις υπόλοιπες παραμέτρους ίδιες (συνολικό αριθμό γραμμών 8, μέγεθος γραμμής 2 λέξεις). Προσοχή δεν υπάρχει απευθείας τέτοια επιλογή στο cache configuration του Ripes. Πρέπει να σκεφτείτε πως θα θέσετε τις τιμές στα (2^N Lines, 2^N Ways) ώστε να το πετύχετε. Επαναλάβετε τις τρεις παραπάνω παραλλαγές του padding (0, 4, 8). Αυτή τη φορά συμπληρώνετε το δεξί μισό του πίνακα απαντήσεων (με επικεφαλίδα Fully Associative).

Padding	Mem access	Direct Mapped			Fully Associative		
		Hit?	Αριθμός block	Tag	Hit?	Αριθμός block	Tag
0	1st	Miss	0	400000	Miss	0	2000000
	2nd	Hit	0	400000	Hit	0	2000000
4	1st	Miss	0	400000	Miss	0	2000000
	2nd	Miss	1	400000	Miss	1	2000001
8	1st	Miss	1	400000	Miss	0	2000001
	2nd	Hit	1	400000	Hit	0	2000001

Πίνακας 1: Πινακας απαντήσεων πρώτου μέρους

4 Μέρος Β: Επίδραση μεγέθους γραμμής

Βάλτε τις παρακάτω τιμές παραμέτρων στο cache.s: arraySize 128, option 1, stepSize 1, repCount 1, και padding 0. Ρυθμίστε την κρυφή μνήμη ως εξής: οργάνωση Direct Mapped, αριθμός γραμμών 8, μέγεθος γραμμής 2 λέξεις. Τρέξτε το πρόγραμμα και καταγράψτε το μοτίβο ευστοχιών αστοχιών ως μια ακολουθία από γράμματα M - miss, H - hit. Η απάντησή σας θα πρέπει να είναι το συντομότερο μοτίβο που επαναλαμβάνεται, για παράδειγμα το ”HHMHHM” θα πρέπει να γραφεί ως ”HHM”. Καταγράψτε και το τελικό ποσοστό ευστοχίας του προγράμματος.

Σκεφτείτε πώς το μοτίβο μπορεί άμεσα να σας δώσει το τελικό ποστοστό ευστοχίας.

Μοτίβο	Ποσοστό ευστοχίας
MHHH	75%

Πίνακας 2: Direct mapped, 8 γραμμές, 2 λέξεις ανά γραμμή

Αλλάξτε το μέγεθος γραμμής σε 4 λέξεις, αλλά για να διατηρηθεί η συνολική χωρητικότητα της cache ίδια, μειώστε τον αριθμό γραμμών σε 4. Βρείτε και γράψτε το νέο μοτίβο και το ποσοστό ευστοχίας.

Μοτίβο	Ποσοστό ευστοχίας
MHHHHHHH	87.5%

Πίνακας 3: Direct mapped, 4 γραμμές, 4 λέξεις ανά γραμμή

Σκεφτείτε σε ποιό από τα δύο είδη τοπικότητας αναφορών μνήμης οφείλεται κάθε ευστοχία του μοτίβου. Αντιγράψτε το μοτίβο που βρήκατε στο τελευταίο ερώτημα, αλλάζοντας κάθε H (hit) είτε σε T (temporal), για χρονική τοπικότητα αναφοράς, είτε σε S (spatial), για χωρική τοπικότητα. Αφήστε τα M ως είναι.

Μοτίβο
MTSTSTST

Πίνακας 4: Μοτίβο με είδη τοπικότητας αναφορών

Σκεφτείτε αν το ποσοστό ευστοχίας θα άλλαζε αν αυξάνατε το repCount με όλες τις υπόλοιπες παραμέτρους προγράμματος και cache αμετάβλητες. Αλλάξτε το cache.s και τρέξτε το για να επιβεβαιώσετε ότι το βρήκατε σωστά. Αν η οργάνωση ήταν Fully Associative και όλοι οι άλλοι παράμετροι έμεναν ίδιοι[1] θα άλλαζε κάτι στο μοτίβο ή στο ποσοστό ευστοχίας; Οι απαντήσεις στις παραπάνω ερωτήσεις δεν χρειάζονται καταγραφή στο αρχείο απαντήσεων.

5 Μέρος Γ: Επίδραση repCount

Βάλτε τις παρακάτω τιμές παραμέτρων στο cache.s: arraySize 32, option 1, stepSize 4, repCount 1, και padding 0. Ρυθμίστε τον cache simulator ως εξής: οργάνωση Direct Mapped, αριθμός γραμμών 16, μέγεθος γραμμής 2 λέξεις. Καταγράψτε τον αριθμό προσπελάσεων που συμβαίνουν σε μία επανάληψη του **εξωτερικού βρόχου** και το τελικό ποσοστό ευστοχίας (στο τέλος της εκτέλεσης όλου του προγράμματος).

Αριθμ. προσπελάσεων μνήμης	Ποσοστό ευστοχίας
8	50%

Πίνακας 5: repCount 1

Αλλάξτε το repCount σε 2 και επαναλάβετε το πείραμα. Καταγράψτε το νέο (τελικό) ποσοστό ευστοχίας.

Ποσοστό ευστοχίας
75%

Πίνακας 6: repCount 2

Σκεφτείτε τί θα συμβεί αν συνεχίσετε να αυξάνετε το repCount. Συμπληρώστε το μικρότερο δυνατό μοτίβο που επαναλαμβάνεται, καταγράφοντας και το είδος τοπικότητας αναφορών μνήμης σε κάθε ευστοχία, **από την δεύτερη επανάληψη και έπειτα**. (Όπως παραπάνω, γράψτε το μοτίβο, αλλά στις ευστοχίες, γράψτε S ή T, ενώ τυχόν αστοχίες συμβολίζονται με M.)

Μοτίβο
T

Πίνακας 7: Μοτίβο με είδη τοπικότητας αναφορών για τη δεύτερη επανάληψη και έπειτα Συμπληρώστε τον μαθηματικό τύπο που δίνει το ποσοστό αστοχιών σε σχέση με το repCount. (Στο αρχείο απαντήσεων) γράψτε το σαν να το γράφατε π.χ. σε Java. Μπορείτε να το επιβεβαιώσετε με μερικά πειράματα, αλλά προσέξτε ότι ο Ripes παρουσιάζει το hit rate (στο πλαίσιο statistics) ενώ εδώ ζητείται το miss rate.

[1] Ο αλγόριθμος αντικατάστασης μπορεί να είναι LRU. Παίζει ρόλο στο συγκεκριμένο πρόγραμμα και cache;

Ποσοστό αστοχιών σε σχέση με το repCount
1/(2*repCount)

Πίνακας 8: repCount 2

6 Μέρος Δ: Επίδραση stepSize και associativity

Βάλτε τις παρακάτω τιμές παραμέτρων στο cache.s: arraySize 128, option 1, stepSize 16, repCount 4, και padding 0. Ρυθμίστε τον cache simulator ως εξής: οργάνωση Direct Mapped, αριθμός γραμμών 16, μέγεθος γραμμής 4 λέξεις. Καταγράψτε τον αριθμό προσπελάσεων **της πρώτης επανάληψης του εξωτερικού βρόχου** και το ποσοστό ευστοχίας. Επιπλέον, γράψτε το μοτίβο προσπελάσεων στην μνήμη της **πρώτης επανάληψης του εξωτερικού βρόχου**, αλλά κάθε σε αστοχία θα γράψετε το είδος της αστοχίας που συμβαίνει: Υ-υποχρεωτική, Σ σύγκρουσης, Χ - Χωρητικότητας και Ε για ευστοχία. Χρησιμοποιούμε Ελληνικούς χαρακτήρες εδώ γιατί στα Αγγλικά όλα τα είδη αστοχιών ξεκινούν με C! **Προσοχή** ενώ προηγουμένως είχε ζητηθεί το συντομότερο δυνατό μοτίβο, εδώ ζητείται πλήρης καταγραφή - χαρακτηρισμός όλων των προσπελάσεων της πρώτης επανάληψης του εξωτερικού βρόχου.

Αριθμ. προσπελάσεων μνήμης	Ποσοστό ευστοχίας	Μοτίβο
16	50%	ΥΕΥΕΥΕΥΕΥΕΥΕΥΕΥΕ

Πίνακας 9: Πρώτη επανάληψη του εξωτερικού βρόχου

Συνεχίστε την προσομοίωση και ολοκληρώστε την **δεύτερη** επανάληψη του εξωτερικού βρόχου. Σημειώστε το συνολικό προσοστό ευστοχίας από την αρχή της εκτέλεσης μέχρι αυτό το σημείο της εκτέλεσης. (Όχι μεχρι το τέλος του προγράμματος.) Γράψτε το μοτίβο ολόκληρης της **δεύτερης** επανάληψης, όπως και προηγουμένως.

Ποσοστό ευστοχίας	Μοτίβο
50%	ΣΕΣΕΣΕΣΕΣΕΣΕΣΕ

Πίνακας 10: Δεύτερη επανάληψη του εξωτερικού βρόχου

Συνεχίστε την προσομοίωση μέχρι να τερματίσει το πρόγραμμα και παρατηρήστε το τελικό ποσοστό ευστοχίας. Αν αλλάζατε το repCount σε μεγαλύτερο αριθμό, θα βελτιωνόταν το τελικό ποσοστό ευστοχίας;

Απάντηση:
Όχι

Αν είχαμε την παράμετρο option του προγράμματος στην τιμή 0, ποιό θα ήταν το τελικό ποσοστό ευστοχίας;

Ποσοστό ευστοχίας
0%

Πίνακας 11: Ποσοστό ευστοχίας με option 0

Επαναφέρετε τις παραμέτρους του cache.s στις αρχικές του μέρους Γ: arraySize 128, option

1, stepSize 16, repCount 4, και padding 0. Ρυθμίστε την cache ώστε να ακολουθεί την Fully Associative οργάνωση, με όλες τις υπόλοιπες παραμέτρους ίδιες (αριθμός γραμμών 16, μέγεθος γραμμής 4 λέξεις). Τρέξτε τον **πρώτο εξωτερικό** βρόχο και παρατηρείστε πώς τοποθετούνται οι γραμμές δεδομένων στην cache. Καταγράψτε το πλήθος των θέσεων (γραμμών) της cache που χρησιμοποιούνται, και το ποσοστό ευστοχίας. Το μοτίβο προσπελάσεων (ακολουθία ευστοχιών - αστοχιών και το είδος των αστοχιών) είναι ίδιο με την Direct Mapped οργάνωση για την πρώτη (εξωτερική) επανάληψη;

Αριθμός θέσεων	Ποσοστό ευστοχίας	Ίδιο μοτίβο;
8	50%	Ναι

Πίνακας 12: Fully associative, πρώτη επανάληψη εξωτερικού βρόχου

Συνεχίστε την προσομοίωση και ολοκληρώστε την **δεύτερη** επανάληψη του εξωτερικού βρόχου. Σημειώστε το συνολικό προσοστό ευστοχίας μέχρι αυτό το σημείο της εκτέλεσης. Το μοτίβο προσπελάσεων (ακολουθία ευστοχιών-αστοχιών και το είδος των αστοχιών) είναι ίδιο με την δεύτερη επανάληψη της Direct Mapped οργάνωσης;

Συνολικό ποσοστό ευστοχίας	Ίδιο μοτίβο με DM;
75%	Οχι

Πίνακας 13: Fully associative, δεύτερη επανάληψη εξωτερικού βρόχου

Συνεχίστε την προσομοίωση μέχρι να τερματίσει το πρόγραμμα και παρατηρήστε το τελικό ποσοστό ευστοχίας. Αν αλλάζατε το repCount σε μεγαλύτερο αριθμό, θα βελτιωνόταν το ποσοστό ευστοχίας;

Απάντηση:
Ναι

Βρείτε την μικρότερη associativity (αριθμός ways στον cache simulator) που να δίνει το ίδιο ποσοστό ευστοχίας με την Fully Associative, χωρίς να αλλάξετε άλλες παραμέτρους ούτε στο πρόγραμμα ούτε στην κρυφή μνήμη.

Απάντηση:
8