



ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ  
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

---

Οργάνωση Υπολογιστών

**Εργαστήριο 11:**  
**Κρυφές Μνήμες και η Επίδοσή τους**

Μανόλης Γ.Η. Κατεβαίνης

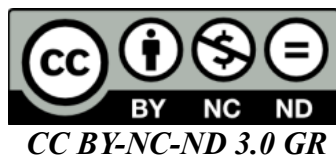
Τμήμα Επιστήμης Υπολογιστών

---

## Άδειες Χρήσης

• Το παρόν εκπαιδευτικό υλικό υπόκειται στην άδεια χρήσης **Creative Commons** και ειδικότερα

*Αναφορά – Μη εμπορική Χρήση – Όχι Παράγωγο Έργο 3.0 Ελλάδα  
(Attribution – Non Commercial – Non-derivatives 3.0 Greece)*



• Εξαιρείται από την ως άνω άδεια υλικό που περιλαμβάνεται στις διαφάνειες του μαθήματος, και υπόκειται σε άλλου τύπου άδεια χρήσης. Η άδεια χρήσης στην οποία υπόκειται το υλικό αυτό αναφέρεται ρητώς.

## Χρηματοδότηση

• Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.

• Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.

• Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



Ευρωπαϊκή Ένωση  
Ευρωπαϊκό Κοινωνικό Ταμείο



Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



# Σειρά Ασκήσεων 11:

## Κρυφές Μνήμες και η Επίδοσή τους

**Βιβλίο** (4η έκδοση, Ελληνικό): διαβάστε το πρώτο ήμισυ του κεφαλαίου 5 – § 5.1 έως και 5.3, σελ. 524-570. Διαβάστε ιδιαίτερα προσεκτικά τις σελίδες 524-542 και 551-564.

### Άσκηση 11.1: Μέσος Χρόνος Προσπέλασης Ιεραρχίας Μνήμης

Όπως είδαμε στο μάθημα, για τον υπολογισμό των επιδόσεων μίας ιεραρχίας μνήμης (π.χ. κρυφή μνήμη (cache memory) με κύρια μνήμη (main memory)) ορίζουμε τις εξής παραμέτρους:

- *hit ratio* (ποσοστό ευστοχίας): τι ποσοστό του συνολικού πλήθους προσπελάσεων στην ιεραρχία μνήμης ήταν εύστοχες προσπελάσεις.
- *miss\_ratio* (ποσοστό αστοχίας): τι ποσοστό του συνολικού πλήθους προσπελάσεων στην ιεραρχία μνήμης ήταν άστοχες προσπελάσεις. Προφανώς,  $hit\_ratio + miss\_ratio = 1$ .
- $t_{hit}$  (hit time - χρόνος ευστοχίας): χρόνος εύστοχης προσπέλασης, δηλαδή πόση ώρα παίρνει μία προσπέλαση που βρίσκει αυτό που ζητά στην μικρή και γρήγορη μνήμη (π.χ. στην κρυφή μνήμη).
- $t_{miss}$  (miss time - χρόνος αστοχίας): χρόνος άστοχης προσπέλασης, δηλαδή πόση ώρα παίρνει (συνολικά) μία προσπέλαση που δεν βρίσκει αυτό που ζητά στην μικρή και γρήγορη μνήμη (π.χ. στην κρυφή μνήμη), και αναγκάζεται να το διαβάσει από την μεγάλη και αργή μνήμη (π.χ. κύρια μνήμη).
- $t_{miss\_penalty} = t_{miss} - t_{hit}$  (miss penalty - επιπλέον κόστος της αστοχίας): πόση ώρα παραπάνω μας κοστίζει η αστοχία από την ευστοχία.

Τότε, ο μέσος χρόνος προσπέλασης στην ιεραρχία μνήμης,  $t_{eff}$  (effective access time), θα είναι, προφανώς:

$$t_{eff} = hit\_ratio * t_{hit} + miss\_ratio * t_{miss} = t_{hit} + miss\_ratio * t_{miss\_penalty}$$

**Άσκηση:** Θεωρήστε μία κρυφή μνήμη με ποσοστό αστοχίας ενάμισι (1.5) %, χρόνο ευστοχίας 1 κύκλο ρολογιού, και κόστος αστοχίας (miss penalty) 40 κύκλους ρολογιού. Πόσος είναι ο μέσος χρόνος προσπέλασης σε αυτή την ιεραρχία μνήμης (σε κύκλους ρολογιού);

### Άσκηση 11.2: Επίδοση Επεξεργαστή με Ιεραρχία Μνήμης

Το CPI των ασκήσεων 10.2 - 10.4 θεωρούσε ότι όλες οι προσπελάσεις μνήμης (instruction fetch, data load, data store) είναι εύστοχες, δηλαδή βρίσκουν αυτό που ζητούν στην κρυφή μνήμη (η οποία έχει χρόνο προσπέλασης 1 κύκλο ρολογιού). Δυστυχώς, η πραγματικότητα διαφέρει σημαντικά από αυτή την ιδανική εικόνα (και, δυστυχώς, τέτοιους εξιδανικευμένους και εξωπραγματικούς αριθμούς επιδόσεων δημοσιεύουν όχι σπάνια πολλές εταιρείες, για ευνόητους λόγους...). Ας δούμε πόσο μπορεί να χειροτερέψουν τα πράγματα με μία μη ιδανική μνήμη.

Θεωρήστε έναν επεξεργαστή MIPS με ομοχειρία, μ' ένα "ιδανικό" CPI (CPI με

ιδανική μνήμη) ίσο με 1.3 (όπως είχαμε συζητήσει στην άσκηση 10.3). Θεωρήστε ότι αυτός τρέχει ένα πρόγραμμα που απαιτεί την εκτέλεση 1 εκατομμυρίου εντολών. (α) Πόσους κύκλους ρολογιού θα χρειάζονταν αυτό το πρόγραμμα για να τρέξει αν η ιεραρχία μνήμης ήταν ιδανική (όλες οι προσπελάσεις εύστοχες);

Ας υπολογίσουμε τώρα πόσες προσπελάσεις μνήμης κάνει αυτό το πρόγραμμα μη ιδανική ιεραρχία μνήμης. Κάθε έντολη κάνει μία προσπέλαση μνήμης για `i_fetch`, και, επιπλέον, οι εντολές `load` και `store` κάνουν μία ακόμα προσπέλαση μνήμης καθεμία.

(β) Θεωρήστε ότι κατά την εκτέλεση του παραπάνω προγράμματος 25% των εκτελούμενων εντολών είναι `load` και 15% είναι `store`. Υπολογίστε το πλήθος προσπελάσεων που κάνει αυτό το πρόγραμμα στην κρυφή μνήμη εντολών (ICache) (για προσκόμιση εντολών - IFetch) και το πλήθος των προσπελάσεων που αυτό κάνει στην κρυφή μνήμη δεδομένων (DCache) (για αναγνώσεις δεδομένων από εντολές `load` ή για εγγραφές δεδομένων από εντολές `store`).

Από τις προσπελάσεις αυτές (β), άλλες είναι εύστοχες, και άλλες άστοχες. Με τις εύστοχες προσπελάσεις, τα πράγματα έχουν όπως και στον ιδανικό υπολογιστή (α). Κάθε άστοχη προσπέλαση, όμως, μας κοστίζει `miss_penalty` κύκλους ρολογιού επιπλέον αυτών που ξοδεύει ο ιδανικός υπολογιστής.

(γ) Πόσες άστοχες προσπελάσεις στην κρυφή μνήμη εντολών κάνει το παραπάνω πρόγραμμα, εάν το `miss_ratio` αυτής της ICache είναι 2.0 % και πόσες άστοχες προσπελάσεις κάνει αυτό στην κρυφή μνήμη δεδομένων, εάν το `miss_ratio` της DCache είναι 4.0 % ;

(δ) Πόσους κύκλους ρολογιού (επιπλέον του ιδανικού) μας κοστίζουν αυτές οι άστοχες προσπελάσεις, εάν το `miss_penalty` είναι 20 κύκλοι ρολογιού; (το ίδιο, 20 κύκλοι, από οιαδήποτε από τις δύο αυτές κρυφές μνήμες πρώτου επιπέδου, L1 Caches, μέχρι την κρυφή μνήμη δευτέρου επιπέδου, L2 Cache, που είναι κοινή για εντολές και για δεδομένα, και που ας θεωρήσουμε εδώ, για απλότητα, ότι ποτέ δεν αστοχεί...) (μιά άλλη υπόθεση που κάνουμε εδώ είναι ότι, όσο διαρκεί ο εξυπηρέτηση μιάς αστοχίας, ο υπόλοιπος επεξεργαστής "κάθεται άπραγος", μη κάνοντας τίποτα χρήσιμο, όσο περιμένει να τελειώσει ο χειρισμός της αστοχίας (πράγμα που δεν είναι αλήθεια σε πολλούς σημερινούς προχωρημένους επεξεργαστές)).

Ο πραγματικός υπολογιστής, λοιπόν, θα αργεί περισσότερο από τον ιδανικό (α) κατά τόσους κύκλους ρολογιού όσοι χάθηκαν λόγω άστοχων προσπελάσεων (δ). Επομένως,

(ε) πόσους κύκλους ρολογιού θα χρειαστεί ο πραγματικός υπολογιστής για να τρέξει το παραπάνω πρόγραμμα του 1 εκατομμυρίου εντολών;

(στ) Πόσο είναι, συνεπώς, το ισοδύναμο CPI του πραγματικού υπολογιστή;

(ζ) Πόσες φορές αργότερος είναι ο πραγματικός από τον ιδανικό υπολογιστή; (δηλαδή πόσες φορές ταχύτερος είναι ο ιδανικός από τον πραγματικό –δηλαδή  $t_{\text{πραγματικός}} / t_{\text{ιδανικός}}$ ).

### Άσκηση 11.3: Κρυφή Μνήμη Μονοσήμαντης Απεικόνισης (Direct Mapped)

Όταν τοποθετούμε ένα αντίτυπο μιάς λέξης από την κύρια μνήμη στην κρυφή μνήμη, αυτό μπορεί να τοποθετηθεί σε **ορισμένες μόνο "επιτρεπτές" θέσεις** της κρυφής μνήμης. Εάν όλες οι θέσεις της κρυφής μνήμης είναι επιτρεπτές (για την κάθε λέξη της κύριας μνήμης), τότε η κρυφή μνήμη λέγεται **πλήρως προσεταιριστική** (fully associative cache). Το μειονέκτημα όμως της πλήρως προσεταιριστικής κρυφής μνήμης είναι το υψηλό της κόστος: για να ψάξουμε να βρούμε αν μιά επιθυμητή λέξη της κύριας μνήμης έχει αυτή τη στιγμή αντίτυπο της στην κρυφή μνήμη, ώστε να το διαβάσουμε από εκεί γρήγορα, πρέπει να ψάξουμε **σε όλες** τις θέσεις της κρυφής

μνήμης, αφού η επιθυμητή λέξη, όταν είχε έλθει στην κρυφή μνήμη (αν είχε έλθει, και αν είναι ακόμα εκεί), ήταν επιτρεπτό να τοποθετηθεί σε οποιαδήποτε θέση της κρυφής μνήμης.

Η οργάνωση κρυφής μνήμης με το μικρότερο δυνατό κόστος είναι η **μονοσήμαντης απεικόνισης** (direct mapped cache). Σε αυτή την οργάνωση, υπάρχει **μία μόνο επιτρεπτή θέση** στην κρυφή μνήμη για την κάθε λέξη της κύριας μνήμης. Έτσι, το ψάξιμο για μία λέξη είναι απλό: πηγαίνουμε στην μοναδική επιτρεπτή θέση για αυτή τη λέξη, και κοιτάμε να δούμε ποιός βρίσκεται εκεί αυτή τη στιγμή: η επιθυμητή λέξη, άλλη λέξη, ή καμία λέξη; (Η απάντηση δίδεται από το address tag και το valid bit, όπως είπαμε στο μάθημα).

Φυσικά, το πρόβλημα με την μονοσήμαντη απεικόνιση είναι ότι μόνο μία από τις πολλές λέξεις της κύριας μνήμης που έχουν όλες την ίδια επιτρεπτή θέση στην κρυφή μνήμη μπορεί να βρίσκεται στην κρυφή μνήμη σε οποιαδήποτε δεδομένη στιγμή –αν το πρόγραμμά μας τύχει να θέλει να δουλέψει (σχεδόν) "ταυτόχρονα" με δύο ή περισσότερες από αυτές τις λέξεις, τότε θα έχει συνεχείς αστοχίες: η μία λέξη θα διώχνει την άλλη. Για να μειώσουμε όσο μπορούμε τις αρνητικές συνέπειες αυτού του φαινομένου, επιδιώκουμε η απεικόνιση των λέξεων της κύριας μνήμης στην κρυφή μνήμη –μια συνάρτηση διασποράς (hashing), ουσιαστικά– να τις σπέρνει όσο πιο ομοιόμορφα μπορεί σε όλη την κρυφή μνήμη, και να τις σπέρνει χωρίς αρνητικές συσχετίσεις με τις συνηθισμένες ιδιότητες τοπικότητας των προγραμμάτων.

Οι συνηθισμένες τοπικότητες των προγραμμάτων είναι η **χρονική** (temporal - ίδια λέξη ξανά και ξανά), και η **χωρική** (spatial - γειτονικές λέξεις χρησιμοποιούνται "μαζί"). Για να μην έχει η απεικόνιση των λέξεων αρνητικές συσχετίσεις με την χωρική τοπικότητα, πρέπει **γειτονικές λέξεις** της κύριας μνήμης (που χρησιμοποιούνται "μαζί") να απεικονίζονται σε **διαφορετικές θέσεις** της κρυφής μνήμης (ώστε να μην διώχνει η μία την άλλη). Επιπλέον, η απεικόνιση των λέξεων πρέπει να είναι απλή, προκειμένου να υλοποιείται τάχιστα στο κύκλωμα. Η συνάρτηση διασποράς που έχει αυτές τις ιδιότητες και χρησιμοποιείται στις κρυφές μνήμες μονοσήμαντης απεικόνισης είναι η εξής (όπου "modulo" είναι το υπόλοιπο της διαίρεσης –θυμηθείτε ότι διαίρεση διά δύναμη του 2 ισοδυναμεί με το να μοιράσουμε απλώς τα bits της λέξης στο κατάλληλο πλήθος MS bits (πηλίκο) και LS bits (υπόλοιπο)):

$$(\text{θέση\_κρυφής\_μν}) = (\text{διεύθ\_κύριας\_μν}) \bmod (\text{μέγεθος\_κρυφής\_μν})$$

Θεωρήστε, για απλότητα σε αυτή την άσκηση, ότι η κύρια μνήμη έχει μέγεθος μόνο 128 Bytes, επομένως οι λέξεις της έχουν διευθύνσεις 0, 4, 8, 12,... 120, 124. Θεωρήστε, επίσης για απλότητα, ότι η κρυφή μνήμη έχει μέγεθος μόνο 32 Bytes, δηλαδή 8 λέξεις, επομένως οι θέσεις της είναι οι 0, 4, 8, 12, 16, 20, 24, και 28.

**(α)** Κάντε έναν πίνακα που να δείχνει ποιές λέξεις της κύριας μνήμης απεικονίζονται σε κάθε μία θέση της κρυφής μνήμης (η κάθε λέξη της κύριας μνήμης, προφανώς, θα εμφανίζεται μία μόνο φορά, σε μία μόνο θέση του πίνακα, αφού απεικονίζεται σε μία μόνο θέση της κρυφής μνήμης). Πόσες λέξεις της κύριας μνήμης απεικονίζονται στην ίδια θέση (διώχνουν η μία την άλλη), για κάθε θέση της κρυφής μνήμης; Είναι ομοιόμορφα διασπαρμένες ή συνωστίζονται όλες σε μερικές μόνο θέσεις της κρυφής μνήμης; Υπάρχουν πουθενά γειτονικές λέξεις της κύριας μνήμης που να απεικονίζονται στην ίδια θέση της κρυφής μνήμης, δηλαδή να "διώχνουν" η μία την άλλη;

**(β)** Στο παραπάνω απλό παράδειγμα, η διεύθυνση μνήμης είναι 7 bits (αφού το μέγεθος της κύριας μνήμης είναι 128 Bytes). Η διεύθυνση της κρυφής μνήμης είναι 5 bits (μέγεθος 32 Bytes), αλλά μόνο τα 3 από αυτά τα bits μας ενδιαφέρουν για να επιλέξουμε τη θέση της κρυφής μνήμης όπου απεικονίζεται μία λέξη, αφού η κρυφή

μνήμη έχει 8 μόνο θέσεις (γιά να το πούμε αλλοιώς: η κρυφή μνήμη δεν ενδιαφέρεται γιά Bytes, αλλά μόνο γιά λέξεις). Όταν ο επεξεργαστής μας δίνει τα 7 bits της διεύθυνσης μνήμης της λέξης που θέλει, πώς θα προκύψουν τα 3 bits που υποδηλώνουν τη θέση της κρυφής μνήμης όπου πρέπει να ψάξουμε; Μεταφράστε τον πίνακα της ερώτησης (α) στο δυαδικό γιά να δείτε την απάντηση έτοιμη μπροστά σας.

(γ) Γιά να βρούμε τη λέξη που θέλει ο επεξεργαστής, είδαμε στο (β) πού πρέπει να ψάξουμε. Όταν φτάσουμε εκεί, η επόμενη ερώτηση είναι εαν υπάρχει εκεί αντίτυπο κάποιας λέξης μνήμης, εαν ναι ποιός, και εαν είναι αυτής που εμείς ψάχνουμε. Το εαν υπάρχει αντίτυπο κάποιας λέξης μνήμης, το απαντάει το **valid bit** (bit εγκυρότητας). Το ποιός λέξης, το απαντάει το **address tag** (ετικέτα διεύθυνσης). Πόσα bits πρέπει να έχει η ετικέτα διεύθυνσης (να κάνετε οικονομία στα bits –κοστίζουν!); Ποιά bits της διεύθυνσης μνήμης που θέλει ο επεξεργαστής πρέπει να συγκρίνουμε με την ετικέτα διεύθυνσης γιά να ξέρουμε αν βρήκαμε τη λέξη που ζητάει (ευστοχία) ή όχι (αστοχία); Γιά να απαντήστε αυτές τις ερωτήσεις, μελετήστε προσεκτικά τον πίνακα που μεταφράσατε στο δυαδικό στην ερώτηση (β), και δείτε μέσω ποιών bits διεύθυνσης μπορεί κανείς να ξεχωρίσει μεταξύ τους τις λέξεις της κύριας μνήμης που επιτρέπεται να απεικονίζονται στη θέση της κρυφής μνήμης όπου ψάχνουμε, και που επομένως θέλουμε να ξέρουμε ποιά απ' όλες αυτές βρίσκεται εκεί αυτή τη στιγμή.

#### Άσκηση 11.4: Ποσοστό Αστοχίας σε Μνήμη Μονοσήμαντης Απεικόνισης

Θεωρήστε την κρυφή μνήμη άμεσης απεικόνισης της άσκησης 11.3, η οποία είναι αρχικά κενή (όλα τα valid bits ψευδή), και θεωρήστε ότι ο επεξεργαστής προσπελαύνει τις εξής διευθύνσεις μνήμης, κατά χρονολογική σειρά:

100,	72,	56,	96,	76,	60,	52,	100,	80,	96,
72,	52,	76,	104,	60,	100,	80,	52,	96,	84,
100,	80,	52,	108,	104,	60,	56,	108,	76,	52,
96,	76,	56,	100,	60,	52,	104,	64,	60,	76

(α) Ποιές από αυτές τις προσπελάσεις είναι άστοχες και ποιές είναι εύστοχες; Γιά να το απαντήστε, γράψτε τις προσπελάσεις στη σειρά, και δίπλα σε καθεμιά σημειώστε σε παρένθεση τη θέση της κρυφής μνήμης όπου αυτή τοποθετείται, και "Α" ή "Ε". Το "Α" ή "Ε" προκύπτει κοιτώντας πίσω στο χρόνο και βλέποντας ποιά λέξη μνήμης είχε μπει σε αυτή τη θέση τελευταία.

(β) Πόσες από τις προσπελάσεις είναι άστοχες; Ποιό είναι το ποσοστό αστοχίας;

(γ) Εστω ότι ο χρόνος εύστοχίας είναι 1 κύκλος ρολογιού, ενώ το επιπλέον κόστος αστοχίας είναι 5 κύκλοι ρολογιού. Τότε, ποιός είναι ο μέσος χρόνος προσπέλασης σε αυτή την ιεραρχία μνήμης; (Βλ. άσκηση 11.1 γιά τους ορισμούς). (Το κόστος αστοχίας που δίδεται εδώ είναι υπερβολικά μικρό γιά σημερινά δεδομένα, όμως και το ποσοστό αστοχίας είναι υπερβολικά μεγάλο λόγω του τεχνητού παραδείγματος που έχουμε).

#### Άσκηση 11.5: Αύξηση του Μεγέθους Block

Στις ασκήσεις 11.3 και 11.4, το μέγεθος **block** της κρυφής μνήμης ήταν μία (1) λέξη μόνο, δηλαδή σε κάθε αστοχία φέρναμε στην κρυφή μνήμη μόνο την μία συγκεκριμένη λέξη που θέλαμε και μας έλειπε. Μιά τέτοια οργάνωση, δεν εκμεταλλεύεται τη χωρική τοπικότητα, δηλαδή ότι τα προγράμματα, μόλις χρειαστούν μία λέξη, έχουν μεγάλη πιθανότητα σε λίγο να χρειαστούν και τις διπλανές της. Γιά να εκμεταλλευτούμε αυτή την ιδιότητα (σε συνδυασμό με το γεγονός ότι προσπελάσεις σε συνεχόμενες διευθύνσεις κύριας μνήμης είναι πολύ

φτηνότερες από προσπελάσεις σε τυχαίες διευθύνσεις) (και για να μειώσουμε και το πλήθος των address tags), αυξάνουμε το μέγεθος των blocks της κρυφής μνήμης.

Ας πούμε ότι στην κρυφή μνήμη της άσκησης 11.3 κάνουμε το μέγεθος block να είναι 2 λέξεις, δηλαδή 8 Bytes. Τότε, η κρυφή μνήμη, μεγέθους πάντα 32 Bytes, θα έχει 4 μόνο blocks, τα 0, 8, 16, και 24. Το block 0 περιέχει τις θέσεις 0 και 4, το block 8 περιέχει τις θέσεις 8 και 12, κ.ο.κ. Επίσης, οι λέξεις μνήμης θεωρούνται ζευγαρωμένες σε blocks ίσου μεγέθους, που είναι ευθυγραμμισμένα στα φυσικά τους όρια (όπως και οι γνωστοί μας περιορισμοί ευθυγράμμισης των προσπελάσεων του επεξεργαστή). Επομένως, τα ζευγάρια των λέξεων μνήμης είναι τα: 0 με 4, 8 με 12, 16 με 20, 24 με 28,... 112 με 116, και 120 με 124.

Σε κάθε αστοχία, φέρνουμε στην κρυφή μνήμη όχι μόνο τη λέξη που χρειαζόμαστε, αλλά και το "ταίρι" της (τα ταίρια της, για μεγαλύτερα blocks), δηλαδή **ολόκληρο το block** στο οποίο αυτή ανήκει. Αυτό το block λέξεων της κύριας μνήμης, το φέρνουμε στο block θέσεων της κρυφής μνήμης όπου αυτό απεικονίζεται, με την ίδια συνάρτηση απεικόνισης όπως πριν.

(α) Πόσες ετικέτες διεύθυνσης (address tags) χρειάζεται τώρα η κρυφή μνήμη; Παρατηρήστε ότι, αφού πάντα φέρνουμε ολόκληρα blocks και ποτέ μεμονωμένες λέξεις, αν ξέρουμε ποιά λέξη βρίσκεται π.χ. στη θέση 24 της κρυφής μνήμης, τότε ξέρουμε αυτόματα και ποιά λέξη βρίσκεται στη θέση 28, δηλαδή στην άλλη θέση του ίδιου block: το "ταίρι" της πρώτης.

(β) Θεωρήστε την ίδια σειρά προσπελάσεων όπως και στην άσκηση [11.4](#). Με τη νέα κρυφή μνήμη, ποιές από αυτές τις προσπελάσεις είναι άστοχες και ποιές είναι εύστοχες; Απαντήστε με τρόπο ανάλογο προς την ερώτηση 11.4(α), αλλά προσέξτε ότι τώρα η πρώτη αστοχία σε μιά λέξη ενός block φέρνει ολόκληρο το block (2 λέξεις), και επομένως η επόμενη προσπέλαση στην άλλη λέξη του ίδιου block θα ευστοχήσει.

(γ) Πόσες από τις προσπελάσεις είναι άστοχες; Ποιό είναι το ποσοστό αστοχίας;

(δ) Έστω ότι ο χρόνος ευστοχίας είναι 1 κύκλος ρολογιού, ενώ το επιπλέον κόστος αστοχίας είναι 6 κύκλοι ρολογιού: 5 κύκλοι για την πρώτη λέξη, και 1 επιπλέον κύκλος για την μία επιπλέον λέξη. Η δεύτερη λέξη μας κοστίζει έναν μόνο κύκλο παραπάνω από την πρώτη επειδή οι δύο λέξεις που φέρνουμε ανήκουν στο ίδιο (ευθυγραμμισμένο) block της κύριας μνήμης, και άρα μπορούν να διαβαστούν "μαζί", πολύ γρηγορότερα από δύο τυχαίες, άσχετες αναγνώσεις που θα χρειάζονταν  $5+5=10$  κύκλους ρολογιού. Τώρα, ποιός είναι ο μέσος χρόνος προσπέλασης σε αυτή την ιεραρχία μνήμης;