Σειρά Ασκήσεων 15: Κρυφές Μνήμες και η Επίδοσή τους

Προθεσμία έως Τετάρτη 12 Μαΐου 2010 (εβδομάδα 12)

[**Up** - Table of Contents] [printer version - **PDF**]
[**Prev** - 14. Processor Performance] [16. Virtual Memory - **Next**]

Βιβλίο (Ελλ. έκδοση): διαβάστε το πρώτο ήμισυ του κεφαλαίου 7 –§ 7.1 έως και 7.3, σελ. 486–528.

Άσκηση 15.1: Μέσος Χρόνος Προσπέλασης Ιεραρχίας Μνήμης

Οπως είδαμε στο μάθημα, γιά τον υπολογισμό των επιδόσεων μιάς 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 μιά 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) μιά 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:

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

Θεωρήστε μιά κρυφή μνήμη με ποσοστό αστοχίας ενάμισι (1.5) %, χρόνο ευστοχίας 1 κύκλο ρολογιού, και κόστος αστοχίας (miss penalty) 60 κύκλους ρολογιού. Πόσος είναι ο μέσος χρόνος προσπέλασης σε αυτή την ιεραρχία μνήμης (σε κύκλους ρολογιού);

Άσκηση 15.2: Επίδοση Επεξεργαστή με Ιεραρχία Μνήμης

Το CPI των ασκήσεων **14.2** - **14.4** θεωρούσε ότι όλες οι προσπελάσεις μνήμης (instruction fetch, data lw, data sw) 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, γιά ευνόητους λόγους...). Ας δούμε πόσο μπορεί να χειροτερέψουν τα πράγματα με μιά μη ιδανική μνήμη.

Θεωρήστε έναν επεξεργαστή MIPS με ομοχειρία, μ' ένα "ιδανικό" CPI (CPI με ιδανική μνήμη) ίσο με 1 (όπως έχουμε πει στο μάθημα για έναν ιδανικό επεξεργαστή με ομοχειρία). Θεωρήστε ότι αυτός τρέχει ένα πρόγραμμα που απαιτεί την εκτέλεση 1 δισεκατομμυρίου εντολών. **(α)** Πόσους κύκλους ρολογιού θα χρειάζονταν αυτό το πρόγραμμα γιά να τρέξει αν η ιεραρχία μνήμης ήταν ιδανική (όλες οι προσπελάσεις εύστοχες);

Ας υπολογίσουμε τώρα πόσες προσπελάσεις μνήμης κάνει αυτό το πρόγραμμα. Κάθε έντολη κάνει μία προσπέλαση μνήμης γιά i_fetch, και, επιπλέον, οι εντολές lw και sw κάνουν μία ακόμα προσπέλαση μνήμης καθεμία. **(β)** Βάσει των ποσοστών εντολών της άσκησης **14.2**, υπολογίστε το πλήθος προσπελάσεων μνήμης που κάνει το παραπάνω πρόγραμμα.

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 μνήμης κάνει το παραπάνω πρόγραμμα, εάν το miss_ratio είναι 1.5 % (όπως και στην άσκηση **15.1**); **(δ)** Πόσους κύκλους ρολογιού (επιπλέον του ιδανικού) μας κοστίζουν αυτές οι άστοχες προσπελάσεις, εάν το miss_penalty είναι 60 κύκλοι ρολογιού (όπως και στην άσκηση 15.1);

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 γιά να τρέξει το παραπάνω πρόγραμμα του 1 δισεκατομμυρίου 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 $t_{πραγματικός} / t_{ιδανικός}$).

Άσκηση 15.3: 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 (γιά την κάθε λέξη της κύριας μνήμη), τότε η κρυφή μνήμη λέγεται **πλήρως προσεταιριστική** (fully associative cache). Το μειονέκτημα όμως της πλήρως προσεταιριστικής κρυφής μνήμης είναι το υψηλό της κόστος: γιά 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 γιά την κάθε λέξη της κύριας μνήμης. Έτσι, το ψάξιμο γιά μιά λέξη είναι απλό: πηγαίνουμε στην μοναδική επιτρεπτή θέση γιά 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 μιά λέξη θα διώχνει την άλλη. Γιά να μειώσουμε όσο μπορούμε τις αρνητικές συνέπειες αυτού του φαινομένου, επιδιώκουμε η απεικόνιση των λέξεων της κύριας μνήμης στην κρυφή μνήμη –μιά συνάρτηση διασποράς (hashing), ουσιαστικά– να τις σπέρνει όσο πιό 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"). Γιά 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)):

(θέση_κρυφής_μν) = (διεύθ_κύριας_μν) **modulo** (μέγεθος_κρυφής_μν)

Θεωρήστε, γιά απλότητα σε αυτή την άσκηση, ότι η κύρια μνήμη έχει μέγεθος μόνο 512 Bytes, επομένως οι λέξεις της έχουν διευθύνσεις 0, 4, 8, 12,... 504, 508. Θεωρήστε, επίσης γιά απλότητα, ότι η κρυφή μνήμη έχει μέγεθος μόνο 64 Bytes, δηλαδή 16 λέξεις, επομένως οι θέσεις της είναι οι 0, 4, 8, 12, ..., 56, 60.

(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), γιά 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 9 bits (αφού το μέγεθος της κύριας μνήμης είναι 512 Bytes). Η διεύθυνση της κρυφής μνήμης είναι 6 bits (μέγεθος 64 Bytes), αλλά μόνο τα 4 από αυτά τα bits μας ενδιαφέρουν γιά να επιλέξουμε τη θέση της κρυφής μνήμης όπου απεικονίζεται μιά λέξη, αφού η κρυφή μνήμη έχει 16 μόνο θέσεις (γιά να το πούμε αλλοιώς: η κρυφή μνήμη δεν ενδιαφέρεται γιά Bytes, αλλά μόνο γιά λέξεις). Όταν ο επεξεργαστής μας δίνει τα 9 bits της διεύθυνσης μνήμης της λέξης που θέλει, πώς θα προκύψουν τα 4 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. Οταν φτάσουμε εκεί, η επόμενη ερώτηση είναι αν υπάρχει εκεί αντίτυπο κάποιας λέξης μνήμης, αν ναι ποιάς, και αν είναι αυτής που εμείς ψάχνουμε. Το αν υπάρχει αντίτυπο κάποιας λέξης μνήμης, το απαντάει το **valid bit** (bit εγκυρότητας). Το ποιάς 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 15.4: Ποσοστό Αστοχίας σε Μνήμη Μονοσήμαντης Απεικόνισης

Θεωρήστε την κρυφή μνήμη άμεσης απεικόνισης της άσκησης **15.3**, η οποία είναι αρχικά κενή (όλα τα valid bits ψευδή), και θεωρήστε ότι ο επεξεργαστής εκτελεί τις παρακάτω προσπελάσεις στη μνήμη (οι προσπελάσεις αναφέρονται σαν εντολές lw/sw και όλες οι διευθύνσεις είναι στο δεκαδικό), κατά χρονολογική σειρά:

> lw 200, lw 400, lw 56, lw 192, lw 404, lw 120, lw 52, sw 200, lw 408, sw 192,
> sw 400, lw 52, sw 404, lw 208, lw 60, sw 200, lw 412, lw 52, sw 192, lw 416,
> sw 200, sw 408, lw 52, lw 204, sw 208, lw 60, lw 56, sw 208, sw 404, lw 52,
> lw 192, lw 404, lw 56, lw 200, lw 60, lw 52, sw 208, lw 64, lw 60, sw 404

(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 "A" ή "E". Το "A" ή "E" προκύπτει κοιτόντας πίσω στο χρόνο και βλέποντας ποιά λέξη μνήμης είχε μπεί σε αυτή τη θέση τελευταία.

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

(γ) Εστω ότι ο χρόνος ευστοχίας στη μνήμη cache είναι 1 κύκλος ρολογιού, ενώ το κόστος πρόσβασης στη μνήμη DRAM σε περίπτωση αστοχίας είναι 60 κύκλοι ρολογιού. Στους 60 κύκλους αυτούς περιλαμβάνεται το κόστος εγγραφής νέων δεδομένων στη μνήμη cache σε περίπτωση αστοχίας. Ποιός είναι ο μέσος χρόνος προσπέλασης σε αυτή την ιεραρχία μνήμης αν υποθέσουμε ότι η μνήμη cache είναι πολιτικής write-through και χωρίς η μνήμη cache να περιλαμβάνει επιπλέον αποθηκευτικό χώρο (write buffers) για να επιταχύνει τα stores; Ποιος είναι ο μέσος χρόνους προσπέλασης σε αυτή την ιεραρχία μνήμης αν υποθέσουμε ότι η μνήμη cache είναι πολιτικής write-back; (Βλ. άσκηση **15.1** γιά τους ορισμούς και τις επιπλέον σημειώσεις για ορισμούς των πολιτικών write-through και write-back).

Άσκηση 15.5: Αύξηση του Μεγέθους Block

Στις ασκήσεις **15.3** και **15.4**, 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 μιά 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 γιά να μειώσουμε και το πλήθος των address tags), αυξάνουμε το μέγεθος των blocks της κρυφής μνήμης.

Ας πούμε ότι στην κρυφή μνήμη της άσκησης 15.3 κάνουμε το μέγεθος block να είναι 2 λέξεις, δηλαδή 8

Bytes. Θέλουμε να διατηρήσουμε το συνολικό όγκο δεδομένων που αποθηκεύει η μνήμη cache σε 64 bytes, άρα η cache θα έχει 8 μόνο blocks, αντί για 16, αλλά κάθε block θα αποθηκεύει 2 λέξεις των 4 bytes. 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,... 496 με 500, και 504 με 508.

Σε κάθε αστοχία, φέρνουμε στην κρυφή μνήμη όχι μόνο τη λέξη που χρειαζόμαστε, αλλά και το "ταίρι" της (τα ταίρια της, γιά 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 ξερουμε 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 **15.4**. Με τη νέα κρυφή μνήμη, ποιές από αυτές τις προσπελάσεις είναι άστοχες και ποιές είναι εύστοχες; Απαντήστε με τρόπο ανάλογο προς την ερώτηση 15.4(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 στη μνήμη cache είναι 1 κύκλος ρολογιού, ενώ το κόστος πρόσβασης στη DRAM είναι 62 κύκλοι ρολογιού: 60 κύκλοι γιά την πρώτη λέξη που διαβάζουμε η γράφουμε στη DRAM, και 2 επιπλέον κύκλοι γιά τη μία επιπλέον λέξη (παρατηρήστε ότι σε αυτή την υλοποίηση διαβάζουμε και γράφουμε ζευγάρια λέξεων στη DRAM). Η δεύτερη λέξη μας κοστίζει δύο μόνο κύκλους παραπάνω από την πρώτη επειδή οι δύο λέξεις που φέρνουμε ανήκουν στο ίδιο (ευθυγραμμισμένο) block της κύριας μνήμης, και άρα μπορούν να διαβαστούν με επικάλυψη (όπως εξηγήσαμε στο μάθημα για τις μνήμες DRAM με πολλές banks), πολύ γρηγορότερα από δύο τυχαίες, άσχετες αναγνώσεις που θα χρειάζονταν 60+60=120 κύκλους ρολογιού. Τώρα, ποιός είναι ο μέσος χρόνος προσπέλασης σε αυτή την ιεραρχία μνήμης εάν υποθέσουμε ότι η μνήμη cache είναι write-back και αν η μήνη cache είναι write-through;

Τρόπος Παράδοσης

Παραδώστε ηλεκτρονικά τις απαντήσεις σας, σε ένα αρχείο "**`ask15.txt`**".

[**Up** - Table of Contents]
[**Prev** - 14. Performance]
[printer version - **PDF**]
[16. Virtual Memory - **Next**]
