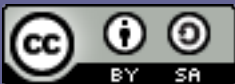


Ιεραρχίες Μνήμης (II)

(οργάνωση, λειτουργία και απόδοση κρυφής μνήμης)

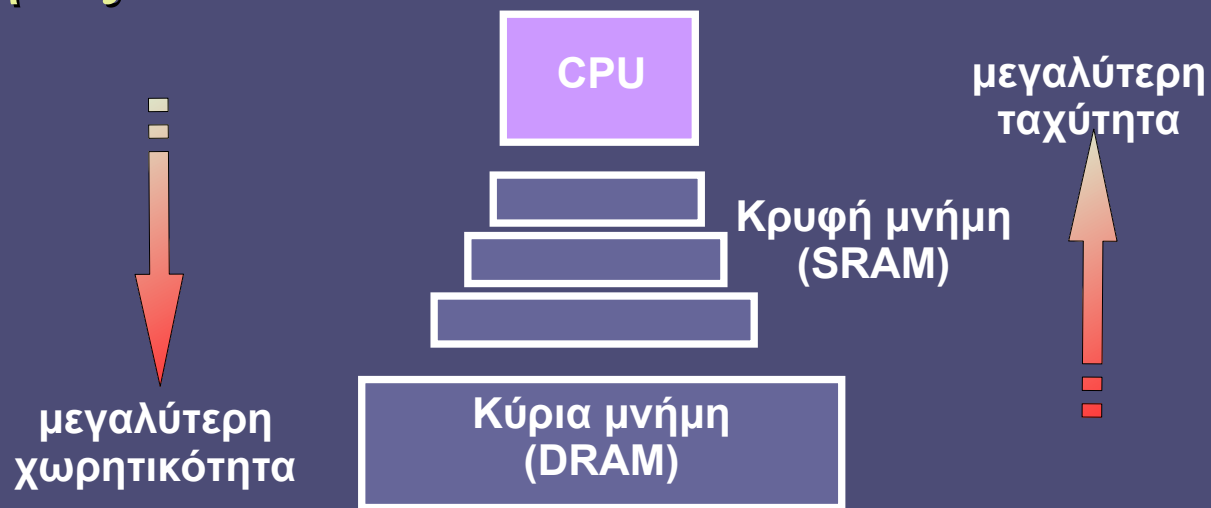
<https://mixstef.github.io/courses/comparch/>

Μ.Στεφανιδάκης



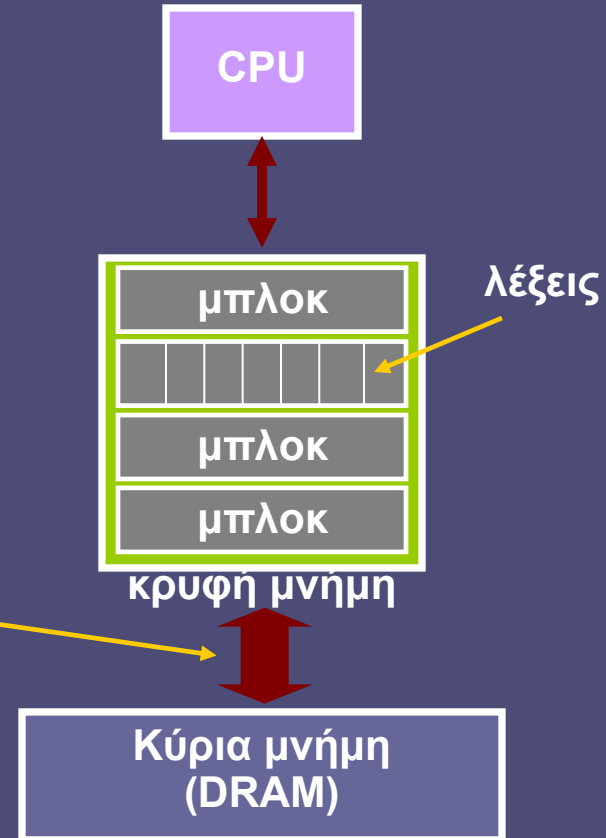
Σκοπός της Ιεραρχίας Μνήμης

- Προσέγγιση της ιδανικής μνήμης
 - Ο επεξεργαστής να βλέπει “μνήμη”
 - Με την ταχύτητα του υψηλότερου επιπέδου
 - Και το μέγεθος του χαμηλότερου επιπέδου
- Η ιεραρχία μνήμης εκμεταλλεύεται την αρχή της τοπικότητας



Μπλοκ (γραμμές) κρυφής μνήμης

- Για την εκμετάλλευση της **χωρικής τοπικότητας**
- Όταν πρέπει να μεταφερθεί μια λέξη, μεταφέρεται **το μπλοκ που την περιέχει**
- Το σύστημα κύριας μνήμης έχει βελτιστοποιηθεί αρχιτεκτονικά για **μεταφορές μπλοκ**
- Οι σημερινοί επεξεργαστές διαθέτουν κρυφές μνήμες με μέγεθος μπλοκ ίσο με 64 bytes



Τοποθέτηση ενός μπλοκ

- Η κύρια μνήμη περιέχει πολύ περισσότερα «μπλοκ» από όσα χωρούν στην κρυφή μνήμη
 - Συνεπώς, στην ίδια θέση της κρυφής μνήμης πρέπει να τοποθετηθούν διαφορετικά μπλοκ από την κύρια μνήμη (προφανώς όχι ταυτόχρονα!)
 - Σύγκρουση μπλοκ
- Πώς αποφασίζεται η θέση ενός μπλοκ στην κρυφή μνήμη;
 - Η απλή λύση: άμεση απεικόνιση (direct mapped cache)
 - Κάθε μπλοκ πηγαίνει σε μία μόνο θέση
(αριθμός μπλοκ) mod (θέσεις στην κρυφή μνήμη)
 - Υπολογίζεται πολύ εύκολα αν οι θέσεις είναι δύναμη του 2

Μέρη διεύθυνσης στην άμεση απεικόνιση

- Η μονάδα επεξεργασίας κάνει αιτήσεις ανάγνωσης/εγγραφής από/σε **διεύθυνση μνήμης**



- Με τη μέθοδο της άμεσης απεικόνισης η διεύθυνση χωρίζεται σε 3 μέρη
 - **byte offset**: σε ποιο byte μέσα στο μπλοκ αρχίζει η ζητούμενη λέξη
 - Για μπλοκ με c bytes, το byte offset είναι $\log_2(c)$ bits
 - **index**: σε ποια θέση της κρυφής μνήμης θα τοποθετηθεί το μπλοκ
 - Σε κρυφή μνήμη με k θέσεις, το index είναι $\log_2(k)$ bits
 - **tag**: τα υπόλοιπα bits της διεύθυνσης
 - Τη χρησιμότητα του tag (ετικέτας) θα δούμε σε λίγο

Παράδειγμα άμεσης απεικόνισης

0x226 = 001 00010 0110

tag byte offset

θέση μπλοκ (index) = 2

0x7E9 = 011 11110 1001

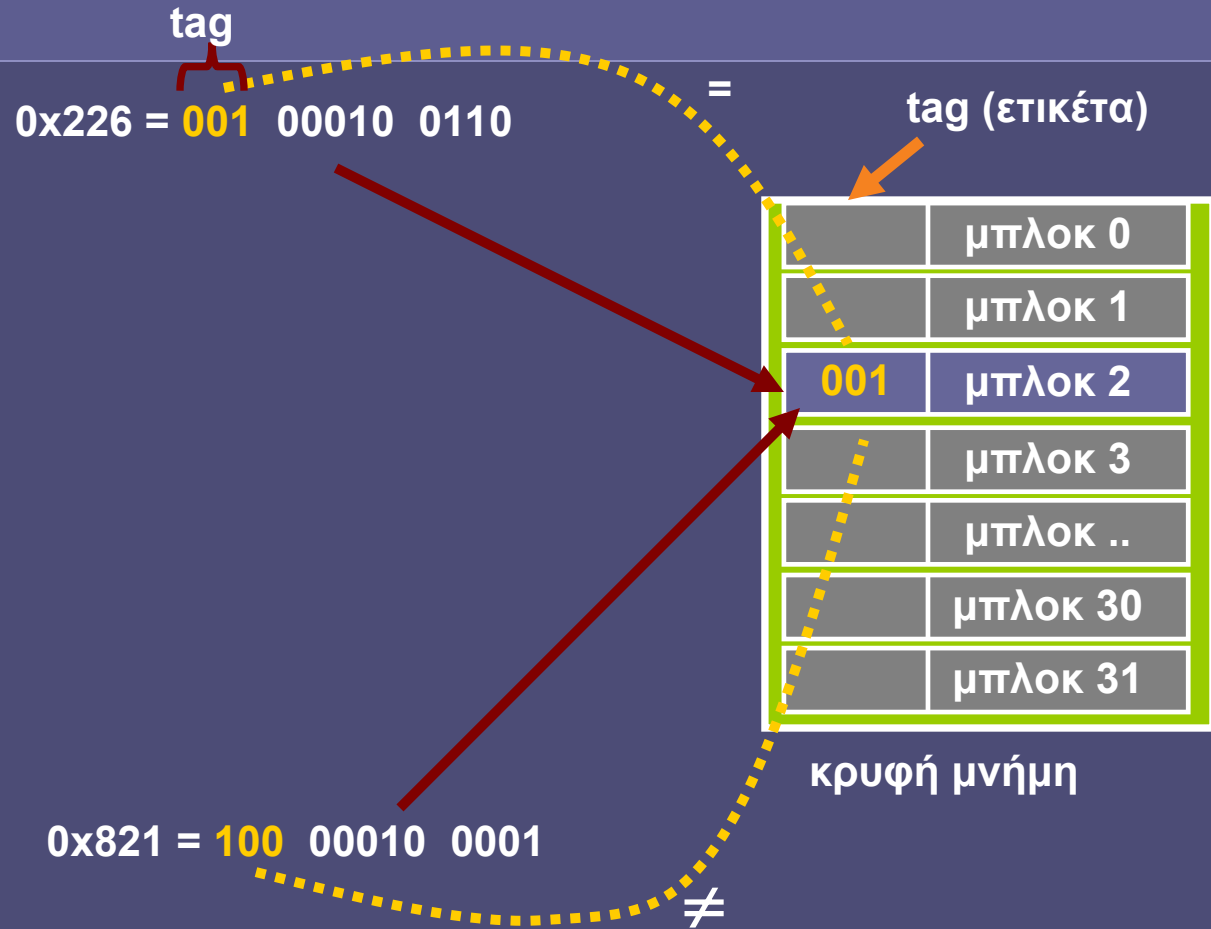
0x821 = 100 00010 0001



κρυφή μνήμη

- Σε κρυφή μνήμη με 16 bytes/μπλοκ, 32 θέσεις για μπλοκ, βρείτε τη θέση όπου τοποθετούνται τα μπλοκ που περιέχουν τις διευθύνσεις 226 (hex), 7E9 (hex) και 821 (hex)
 - block offset = 4 bits ($\log_2 16$), index = 5 bits ($\log_2 32$)

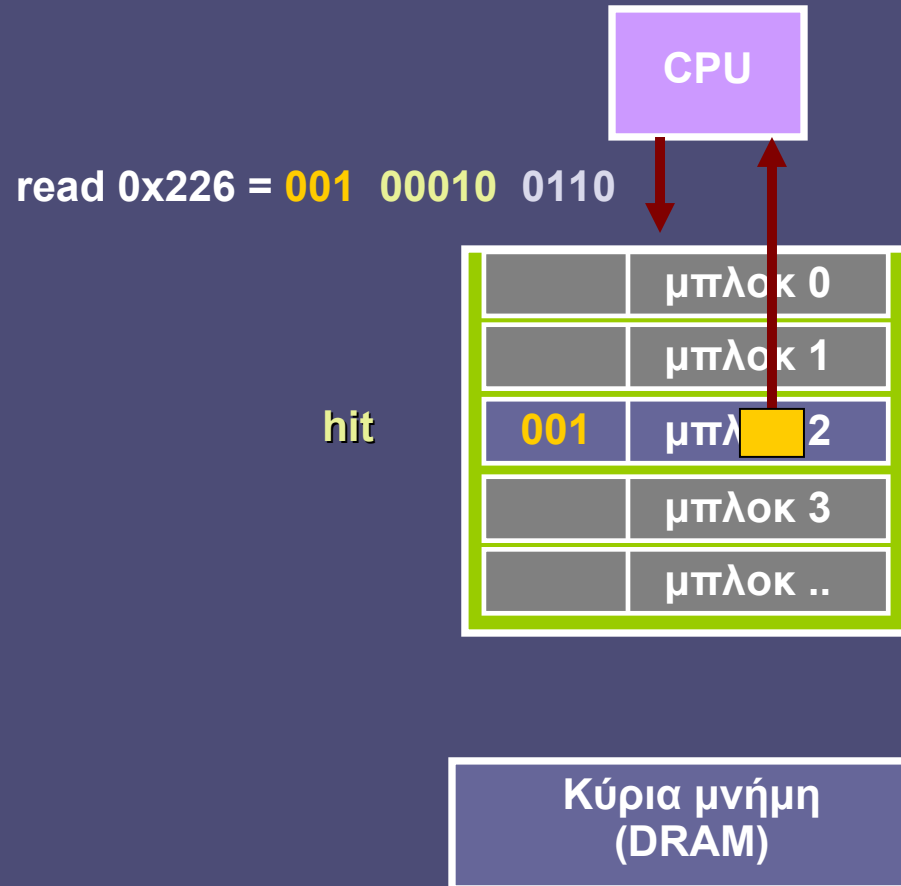
Ποιο μπλοκ βρίσκεται τώρα σε κάθε θέση;



Επίσης: περιέχει η θέση κάποιο έγκυρο μπλοκ;
valid bit (V)

- Σύγκριση με **ετικέτα (tag)** της επιλεγμένης θέσης στην κρυφή μνήμη:
- Αυτή τη στιγμή υπάρχει το μπλοκ που περιέχει τη διεύθυνση 226(hex)

Ανάγνωση: Cache Hit

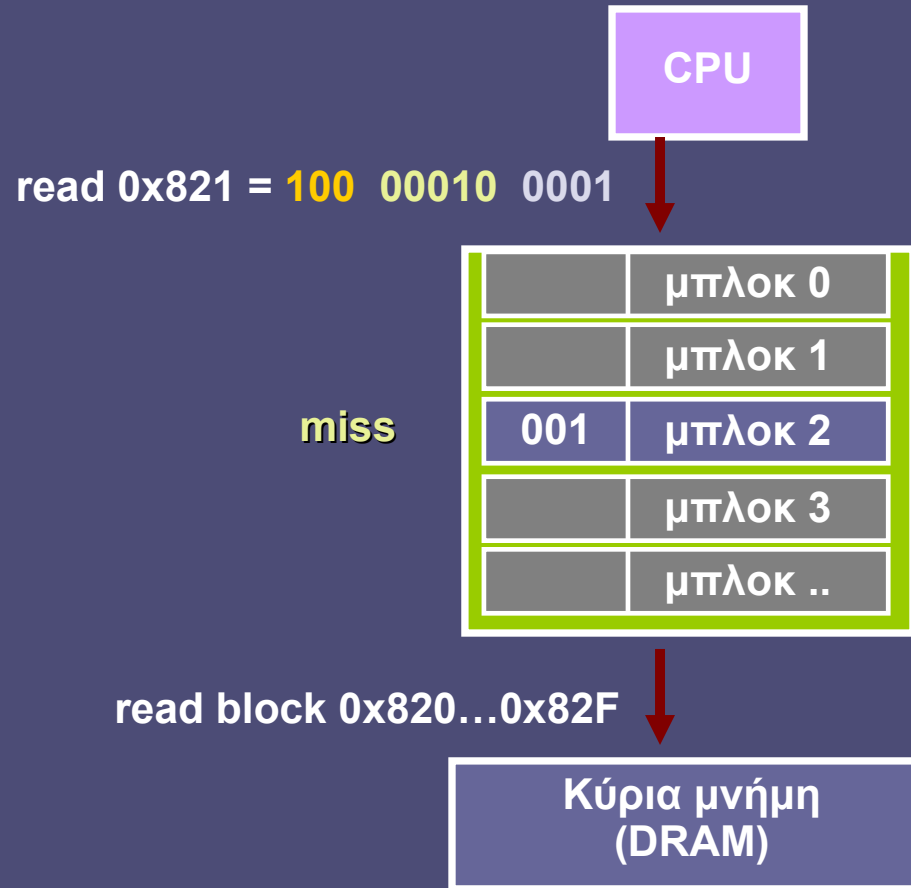


Η ζητούμενη διεύθυνση βρίσκεται στην κρυφή μνήμη, το περιεχόμενο επιστρέφεται στην ΚΜΕ

Ανάγνωση: Cache Miss (1)

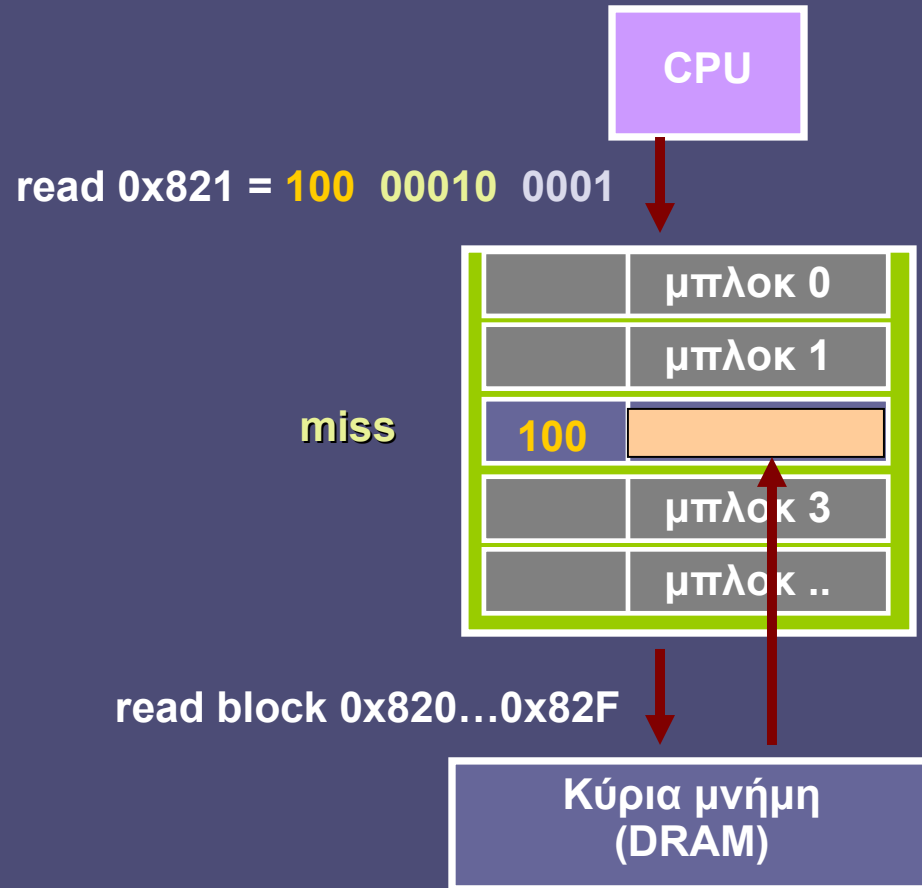


Ανάγνωση: Cache Miss (2)



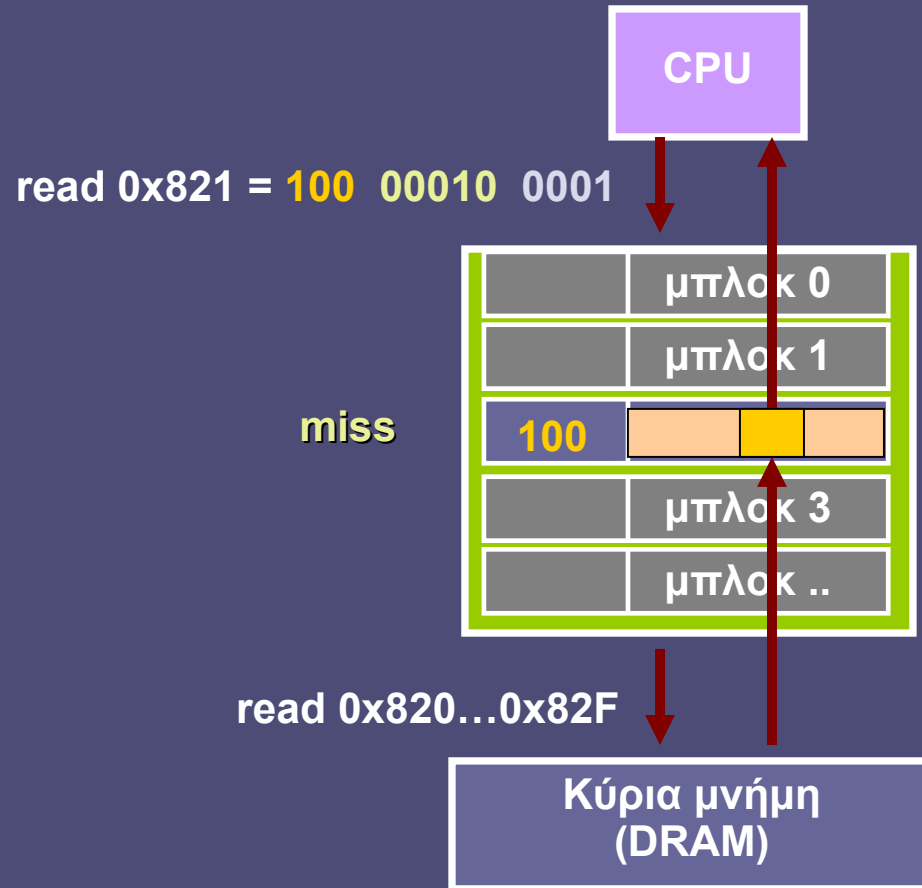
Το μπλοκ που περιέχει τη διεύθυνση 821(hex) ζητείται από την κύρια μνήμη

Ανάγνωση: Cache Miss (3)



Το μπλοκ που περιέχει τη διεύθυνση 821(hex) εισάγεται στην κρυφή μνήμη

Ανάγνωση: Cache Miss (4)



Το περιεχόμενο της διεύθυνσης 821(hex) επιστρέφεται στην ΚΜΕ

Εγγραφή στην κρυφή μνήμη

- **Write Hit**

- Η νέα τιμή ενημερώνεται **μόνο** στην κρυφή μνήμη (**write-back**)
 - Η τιμή στην κύρια μνήμη ενημερώνεται **όταν το μπλοκ εκτοπίζεται από την κρυφή μνήμη**
 - Η εναλλακτική πολιτική **write-through** ενημερώνει και την κύρια μνήμη αλλά είναι πολύ πιο αργή
- Απαιτείται επιπλέον λογική (hardware) για τον έλεγχο της **συνοχής** των δεδομένων
 - Όλοι οι πυρήνες (διαφορετικές κρυφές μνήμες) πρέπει να βλέπουν τα ίδια δεδομένα

- **Write Miss**

- Το μπλοκ έρχεται πρώτα στην κρυφή μνήμη από την κύρια μνήμη (**write-allocate**)

Εγγραφή: Cache Hit

write 0x226 = 001 00010 0110

Η ζητούμενη διεύθυνση βρίσκεται στην κρυφή μνήμη, τα νέα δεδομένα γράφονται στην κρυφή μνήμη. Το μπλοκ σημειώνεται ως (D)irty.

Τα νέα δεδομένα **δεν γράφονται** στην κύρια μνήμη (πολιτική **write-back**)

hit



Κύρια μνήμη
(DRAM)

Εγγραφή: Cache Miss (1)

write 0x821 = 100 00010 0001

CPU

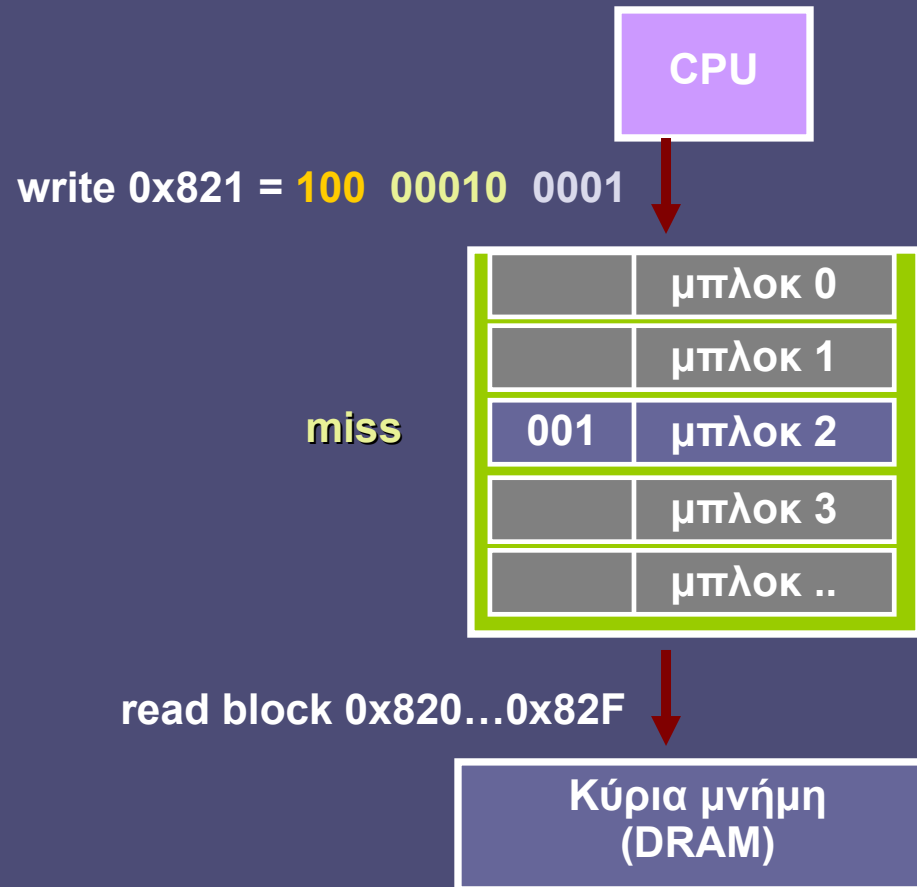
miss

	μπλοκ 0
	μπλοκ 1
001	μπλοκ 2
	μπλοκ 3
	μπλοκ ..

Κύρια μνήμη
(DRAM)

Η ζητούμενη διεύθυνση δεν βρίσκεται στην κρυφή μνήμη

Εγγραφή: Cache Miss (2)



Το μπλοκ που περιέχει τη διεύθυνση 821(hex) ζητείται από την κύρια μνήμη

Εγγραφή: Cache Miss (3)

Το μπλοκ που περιέχει τη διεύθυνση 821(hex) εισάγεται στην κρυφή μνήμη (πολιτική *write-allocate*)

Προσοχή: εάν το μπλοκ που εκτοπίζεται είναι *Dirty*, θα πρέπει να γραφεί πρώτα στην κύρια μνήμη πριν αντικατασταθεί

write 0x821 = 100 00010 0001

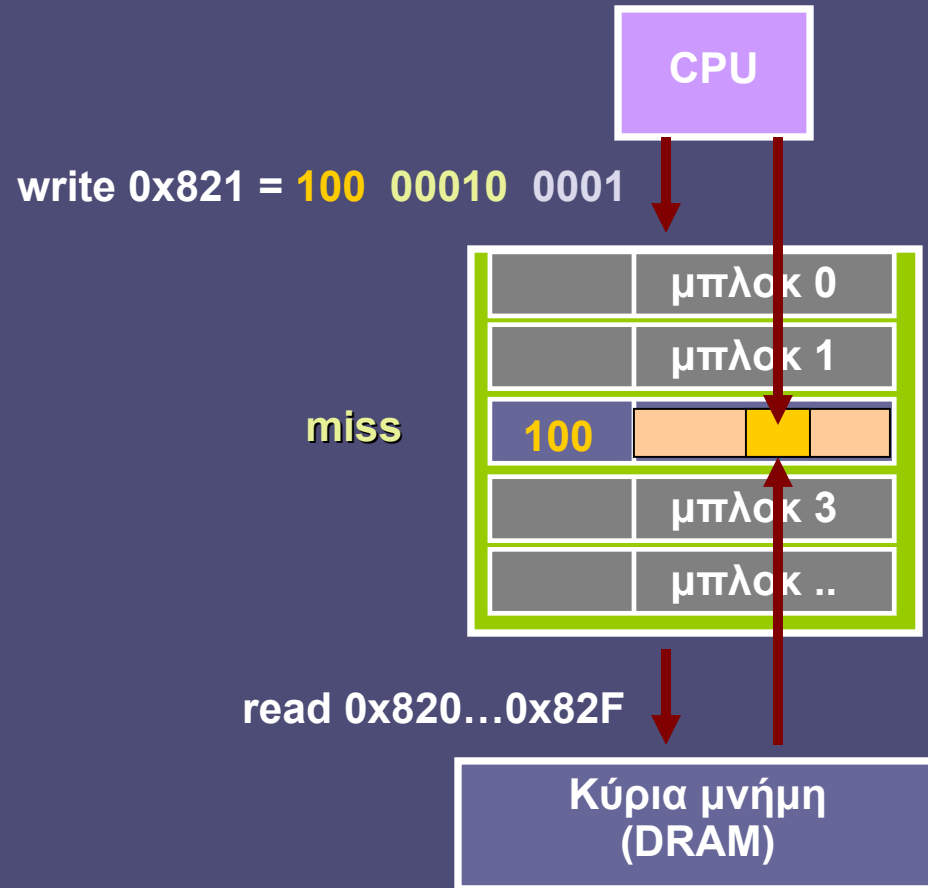
miss



read block 0x820...0x82F

Κύρια μνήμη (DRAM)

Εγγραφή: Cache Miss (4)



Το νέο περιεχόμενο της διεύθυνσης 821(hex) γράφεται στην κρυφή μνήμη

Χαρακτηριστικά απόδοσης κρυφής μνήμης

- **Hit Rate**
 - Ποσοστό προσπελάσεων μνήμης, όπου τα δεδομένα βρίσκονται στην κρυφή μνήμη
- **Miss Rate**
 - Ποσοστό προσπελάσεων μνήμης, όπου τα δεδομένα δεν βρίσκονται στην κρυφή μνήμη
 - (1-hit rate)
- **Hit Time**
 - Ο χρόνος για την προσπέλαση δεδομένων σε hit
- **Miss Penalty**
 - Ο χρόνος για την προσπέλαση, μεταφορά και τοποθέτηση των δεδομένων miss από την κύρια στην κρυφή μνήμη και στην ΚΜΕ

Τι δημιουργεί cache misses;

- Η πρώτη φορά προσπέλασης ενός μπλοκ
 - Όταν ζητούνται από τη μονάδα επεξεργασίας μπλοκ που δεν βρέθηκαν **ποτέ μέχρι τώρα** στην κρυφή μνήμη
- Λόγω της πεπερασμένης χωρητικότητας της κρυφής μνήμης
 - Η κρυφή μνήμη **δεν χωράει** όλα τα μπλοκ ταυτόχρονα
 - Μπλοκ που τοποθετούνται στην **ίδια θέση** στην κρυφή μνήμη, συναγωνίζονται για τη θέση αυτή
 - Ένα νέο μπλοκ όταν τοποθετηθεί στην κρυφή μνήμη **εκτοπίζει** ένα προηγούμενο διαφορετικό μπλοκ που βρισκόταν στην ίδια θέση

Το κόστος των cache misses

- Χαμένοι κύκλοι ρολογιού
 - Σε αναμονή για προσπέλαση κύριας μνήμης

Κύκλοι Αναμονής =

*Προσπελάσεις μνήμης * Miss Rate * Miss Penalty*
- Είναι απλουστευμένο μοντέλο γιατί στην πραγματικότητα:
 - Διαφορετικό Miss Rate ανά κατηγορίες εντολών
 - Διαφορετικό Miss Rate για ανάγνωση-εγγραφή
 - Δυσκολεύουν τον υπολογισμό ενός ακριβούς miss rate
 - Σύνθετη ανάλυση για **εκτέλεση εκτός σειράς**
 - Ο επεξεργαστής “κρύβει” την καθυστέρηση εκτελώντας κάτι άλλο
 - Δυσκολεύει τον υπολογισμό ενός ακριβούς miss penalty

Παράδειγμα υπολογισμού κόστους misses

- Σύστημα έχει ιδανικό $CPI = 1$
 - Όταν έχουμε cache hits
- 40% των εντολών διαβάζουν ή γράφουν δεδομένα από/στη μνήμη
- Miss rate = 2%
- Miss penalty = 20 κύκλοι ρολογιού
 - Πόσες προσπελάσεις μνήμης ανά εντολή;
 - Πόσα misses ανά εντολή;
 - Ποιο το πραγματικό CPI αν λάβουμε υπόψη και τα misses;

Παράδειγμα υπολογισμού κόστους misses

- Σύστημα έχει ιδανικό $CPI = 1$
 - Όταν έχουμε cache hits
- 40% των εντολών διαβάζουν ή γράφουν δεδομένα από/στη μνήμη
- Miss rate = 2%
- Miss penalty = 20 κύκλοι ρολογιού
 - Πόσες προσπελάσεις μνήμης ανά εντολή;
 1 (ανάκληση εντολής) + 0.4×1 (ανάγνωση/εγγραφή δεδομένων) = 1.4 προσπ./εντολή
 - Πόσα misses ανά εντολή;
 - Ποιο το πραγματικό CPI αν λάβουμε υπόψη και τα misses;

Παράδειγμα υπολογισμού κόστους misses

- Σύστημα έχει ιδανικό $CPI = 1$
 - Όταν έχουμε cache hits
- 40% των εντολών διαβάζουν ή γράφουν δεδομένα από/στη μνήμη
- Miss rate = 2%
- Miss penalty = 20 κύκλοι ρολογιού
 - Πόσες προσπελάσεις μνήμης ανά εντολή;
 1 (ανάκληση εντολής) + $0,4 \times 1$ (ανάγνωση/εγγραφή δεδομένων) = $1,4$ προσπ./εντολή
 - Πόσα misses ανά εντολή;
 $1,4$ προσπελάσεις/εντολή $\times 0,02 = 0,028$ misses/εντολή
 - Ποιο το πραγματικό CPI αν λάβουμε υπόψη και τα misses;

Παράδειγμα υπολογισμού κόστους misses

- Σύστημα έχει ιδανικό $CPI = 1$
 - Όταν έχουμε cache hits
- 40% των εντολών διαβάζουν ή γράφουν δεδομένα από/στη μνήμη
- Miss rate = 2%
- Miss penalty = 20 κύκλοι ρολογιού
 - Πόσες προσπελάσεις μνήμης ανά εντολή;
 1 (ανάκληση εντολής) + $0,4 \times 1$ (ανάγνωση/εγγραφή δεδομένων) = $1,4$ προσπ./εντολή
 - Πόσα misses ανά εντολή;
 $1,4$ προσπελάσεις/εντολή $\times 0,02 = 0,028$ misses/εντολή
 - Ποιο το πραγματικό CPI αν λάβουμε υπόψη και τα misses;
 1 κύκλος (ιδανικό CPI) + $0,028$ misses $\times 20$ cycles (κύκλοι αναμονής) = $1 + 0,56 = 1,56$

Μειώνοντας το κόστος των cache misses

- Βελτίωση της απόδοσης

Κύκλοι Αναμονής =

*Προσπελάσεις μνήμης * Miss Rate * Miss Penalty*

- Παράγοντες βελτίωσης
 - Μείωση του miss rate
 - Μείωση του miss penalty

Τεχνικές μείωσης miss rate

- Αντιμετώπιση αιτιών που προκαλούν misses
- Αύξηση χωρητικότητας κρυφής μνήμης
 - *Αλλά:* μια μεγάλη κρυφή μνήμη μπορεί να είναι πιο αργή (αύξηση hit time)
- Αύξηση του μεγέθους του μπλοκ
 - Προσπάθεια εκμετάλλευσης της χωρικής τοπικότητας
 - *Αλλά:* αυξάνει το miss penalty
 - Πιθανόν να αυξάνει τελικά το miss rate, λόγω των *λιγότερων* μπλοκ στην κρυφή μνήμη
- Ευέλικτες τεχνικές τοποθέτησης των μπλοκ
 - Όστε να παραμένουν περισσότερο στην κρυφή μνήμη

Το πρόβλημα με την άμεση απεικόνιση

- Η τοποθέτηση των μπλοκ στις θέσεις της κρυφής μνήμης με τη μέθοδο της άμεσης απεικόνισης
 - Είναι γρήγορη και απαιτεί απλούστερο κύκλωμα
 - Κατάλληλη για τις κρυφές μνήμες **κοντά** στη μονάδα επεξεργασίας (1^ο επιπέδου, L1)
- Επειδή όμως κάθε μπλοκ τοποθετείται ανελαστικά σε μια και μόνο θέση
 - Μπορεί να προκαλέσει αυξημένες συγκρούσεις μπλοκ μέσα στο ίδιο εκτελούμενο πρόγραμμα
 - Με αποτέλεσμα τη συνεχή αντικατάσταση μπλοκ που έτυχε να απεικονιστούν στην ίδια θέση της κρυφής μνήμης
 - Ακόμα κι αν υπάρχουν άλλες θέσεις που δεν χρησιμοποιούνται τη στιγμή εκείνη

Παράδειγμα

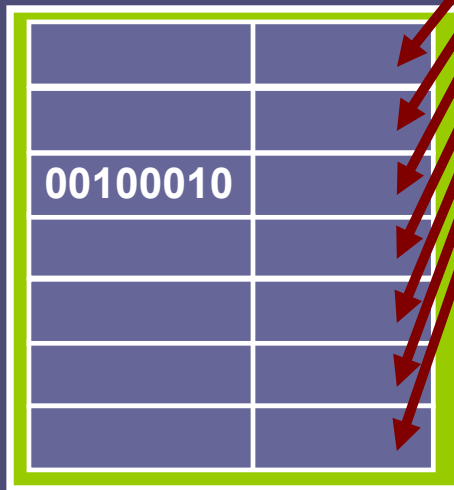
```
for (i=0;i<N;i++) {  
    a[i] = b[i]+c[i];  
}
```

- Τι θα συμβεί αν οι πίνακες a,b,c, βρίσκονται στη μνήμη σε τέτοιες διευθύνσεις ώστε τα μπλοκ που περιέχουν τα a[i], b[i], c[i] να τοποθετούνται στην ίδια θέση της κρυφής μνήμης;

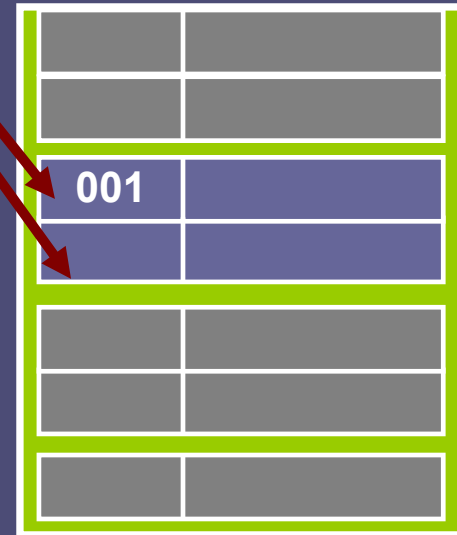
Ευέλικτες τεχνικές τοποθέτησης μπλοκ

0x226 = 001 00010 0110

fully associative



N-way set associative



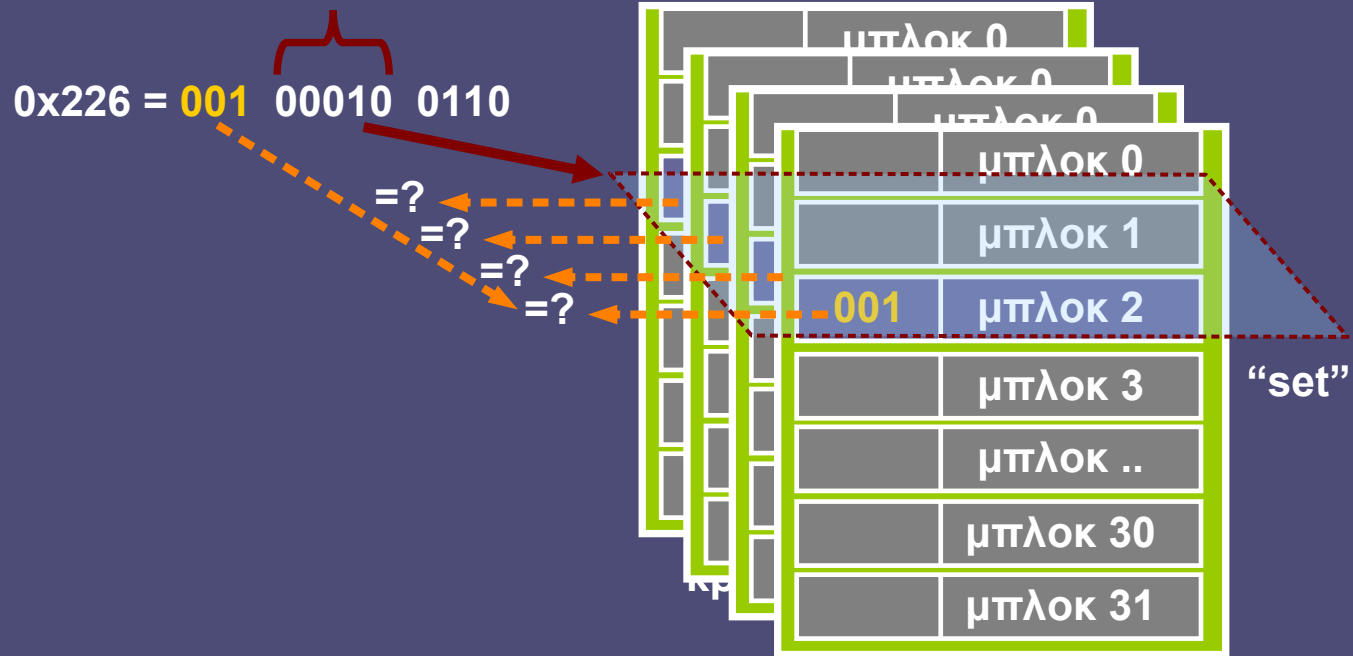
Πιθανή η αύξηση
του hit time λόγω
πιο πολύπλοκου
κυκλώματος

Πιθανή τοποθέτηση
Παράλληλη αναζήτηση

Αντικατάσταση: least recently used (LRU)

Παράδειγμα: 4-way set associativity

προσδιορίζει το set των 4 θέσεων



Η τοποθέτηση γίνεται σε μία από 4 πιθανές θέσεις

Παράδειγμα οργάνωσης κρυφής μνήμης

- Επεξεργαστής έχει 48 bits διεύθυνσης
- 32KB κρυφή μνήμη 1^{ου} επιπέδου (L1)
 - 8-way set associative
 - Μέγεθος μπλοκ = 64 bytes
- Πόσες θέσεις για μπλοκ συνολικά;
- Πόσα sets;
- Ποια τα μέρη της διεύθυνσης και το εύρος τους σε bits;
 - Πώς θα ήταν τα παραπάνω μεγέθη αν είχαμε απλή άμεση απεικόνιση;

Παράδειγμα οργάνωσης κρυφής μνήμης

- Επεξεργαστής έχει 48 bits διεύθυνσης
- 32KB κρυφή μνήμη 1^{ου} επιπέδου (L1)
 - 8-way set associative
 - Μέγεθος μπλοκ = 64 bytes
- Πόσες θέσεις για μπλοκ συνολικά;
 $32\text{KB}/64 = 2^{15} / 2^6 = 2^9 = 512$ θέσεις
- Πόσα sets;
- Ποια τα μέρη της διεύθυνσης και το εύρος τους σε bits;
 - Πώς θα ήταν τα παραπάνω μεγέθη αν είχαμε απλή άμεση απεικόνιση;

Παράδειγμα οργάνωσης κρυφής μνήμης

- Επεξεργαστής έχει 48 bits διεύθυνσης
- 32KB κρυφή μνήμη 1^{ου} επιπέδου (L1)
 - 8-way set associative
 - Μέγεθος μπλοκ = 64 bytes
- Πόσες θέσεις για μπλοκ συνολικά;
 $32\text{KB}/64 = 2^{15} / 2^6 = 2^9 = 512$ θέσεις
- Πόσα sets;
 $512/8 = 2^9 / 2^3 = 2^6 = 64$ sets
- Ποια τα μέρη της διεύθυνσης και το εύρος τους σε bits;
 - Πώς θα ήταν τα παραπάνω μεγέθη αν είχαμε απλή άμεση απεικόνιση;

Παράδειγμα οργάνωσης κρυφής μνήμης

- Επεξεργαστής έχει 48 bits διεύθυνσης
- 32KB κρυφή μνήμη 1^{ου} επιπέδου (L1)
 - 8-way set associative
 - Μέγεθος μπλοκ = 64 bytes
- Πόσες θέσεις για μπλοκ συνολικά;
 $32\text{KB}/64 = 2^{15} / 2^6 = 2^9 = 512$ θέσεις
- Πόσα sets;
 $512/8 = 2^9 / 2^3 = 2^6 = 64$ sets
- Ποια τα μέρη της διεύθυνσης και το εύρος τους σε bits;
byte offset = 6 bits, (set) index = 6 bits, tag = 48 – 6 – 6 = 36 bits
 - Πώς θα ήταν τα παραπάνω μεγέθη αν είχαμε απλή άμεση απεικόνιση;

Παράδειγμα οργάνωσης κρυφής μνήμης

- Επεξεργαστής έχει 48 bits διεύθυνσης
- 32KB κρυφή μνήμη 1^{ου} επιπέδου (L1)
 - 8-way set associative
 - Μέγεθος μπλοκ = 64 bytes
- Πόσες θέσεις για μπλοκ συνολικά;
 $32\text{KB}/64 = 2^{15} / 2^6 = 2^9 = 512$ θέσεις
- Πόσα sets;
 $512/8 = 2^9 / 2^3 = 2^6 = 64$ sets
- Ποια τα μέρη της διεύθυνσης και το εύρος τους σε bits;
byte offset = 6 bits, (set) index = 6 bits, tag = 48 – 6 – 6 = 36 bits
 - Πώς θα ήταν τα παραπάνω μεγέθη αν είχαμε απλή άμεση απεικόνιση;
byte offset = 6 bits, index = 9 bits, tag = 48 – 6 – 9 = 33 bits

Παράδειγμα οργάνωσης κρυφής μνήμης

- Επεξεργαστής έχει 32 bits διεύθυνσης
- 4KB κρυφή μνήμη δεδομένων 1^{ου} επιπέδου
 - 64-way set associative
 - Μέγεθος μπλοκ = 16 bytes
- Πόσες θέσεις για μπλοκ συνολικά;
- Πόσα sets;
- Ποια τα μέρη της διεύθυνσης και το εύρος τους σε bits;
 - Πώς θα ήταν τα παραπάνω μεγέθη αν είχαμε απλή άμεση απεικόνιση;

Παράδειγμα οργάνωσης κρυφής μνήμης

- Επεξεργαστής έχει 32 bits διεύθυνσης
- 4KB κρυφή μνήμη δεδομένων 1^{ου} επιπέδου
 - 64-way set associative
 - Μέγεθος μπλοκ = 16 bytes
- Πόσες θέσεις για μπλοκ συνολικά;
 $4KB/16 = 2^{12} / 2^4 = 2^8 = 256$ θέσεις
- Πόσα sets;
 $256/64 = 2^8 / 2^6 = 2^2 = 4$ sets
- Ποια τα μέρη της διεύθυνσης και το εύρος τους σε bits;
byte offset = 4 bits, (set) index = 2 bits, tag = 32 – 4 – 2 = 26 bits
 - Πώς θα ήταν τα παραπάνω μεγέθη αν είχαμε απλή άμεση απεικόνιση;
byte offset = 4 bits, index = 8 bits, tag = 32 – 4 – 8 = 20 bits

Τεχνικές μείωσης miss penalty

- Μείωση των χρόνων μεταφοράς μπλοκ
 - Βελτιστοποιήσεις στην επικοινωνία με την κύρια μνήμη
 - Έτσι ώστε ένα ολόκληρο μπλοκ να μεταφέρεται με τη μικρότερη δυνατή καθυστέρηση (**bursts**)
- Πολυεπίπεδες ιεραρχίες κρυφής μνήμης
 - Μείωση miss penalty πρώτου επιπέδου (L1)
 - L1: μικρότερο μέγεθος, μεγαλύτερη ταχύτητα
 - Μεγαλύτερο miss rate αλλά miss penalty μικρότερο
 - L2: μεγαλύτερο μέγεθος, μικρότερη ταχύτητα
 - Αργότερη αλλά δεν επηρεάζει hit time επεξεργαστή
 - L3: κοινή για ομάδες πυρήνων

Οι σύγχρονοι επεξεργαστές έχουν L1, L2 και L3 caches (ίσως και 4ο επίπεδο, ως cache «τελευταίας ευκαιρίας»)

Στην εποχή των multicore συστημάτων

- Ιεραρχία Μνήμης
- Κρυφή Μνήμη
- Απόδοση κρυφής μνήμης

Τύπος	Μέγεθος	Χρόνος προσπέλασης	Ρυθμός μεταφοράς
L1	32KB 8-way	4-6 cycles	192b/cycle
L2 (MLC)	1MB 16-way	14 cycles	64b/cycle
L3 (LLC)	1.375MB /core	50-70 cycles	32b/cycle

- Παράδειγμα: Intel Xeon Scalable Processors
 - max 28 cores
 - L1 και L2 caches: κάθε πυρήνας έχει τις δικές του
 - L3: κοινή για όλους τους πυρήνες
 - δεν περιέχει υποχρεωτικά ότι υπάρχει σε L1, L2

Βελτιστοποίηση απόδοσης κρυφής μνήμης

- **Αρχιτεκτονικές βελτιώσεις**
 - **Pipelining**
 - Διάσπαση σε βαθμίδες, π.χ. σύγκριση tags, ανάγνωση data κ.ο.κ
 - **Non-blocking** – εξυπηρέτηση πολλαπλών αιτήσεων
 - Ένα miss δεν καθυστερεί επόμενα hits
 - Πολλαπλά επίπεδα κρυφής μνήμης στο chip του επεξεργαστή
- **Ο ρόλος του λογισμικού (μεταγλωττιστές)**
 - Αναδιοργάνωση προγραμμάτων για αύξηση της τοπικότητας (κυρίως στους βρόχους επανάληψης)
 - **Prefetching**: μετακίνηση δεδομένων στην κρυφή μνήμη πριν αυτά χρειαστούν στον επεξεργαστή

Η απόδοση της κρυφής μνήμης συνοπτικά

- Καθοριστική για τα σύγχρονα υπολογιστικά συστήματα
- Μείωση του miss rate ή του miss penalty
 - Όμως: η συμπεριφορά της ιεραρχίας μνήμης επηρεάζεται από πολλούς παράγοντες
- Η πραγματική συμπεριφορά
 - Είναι σύνθετη – απαιτούνται εξομοιώσεις πριν τη σχεδίαση νέων συστημάτων
 - Είναι διαφορετική ανά εφαρμογή – δεν υπάρχει ένα μόνο αντιπροσωπευτικό πρόγραμμα
 - Είναι διαφορετική ανά υπολογιστικό σύστημα – desktop, server ή embedded