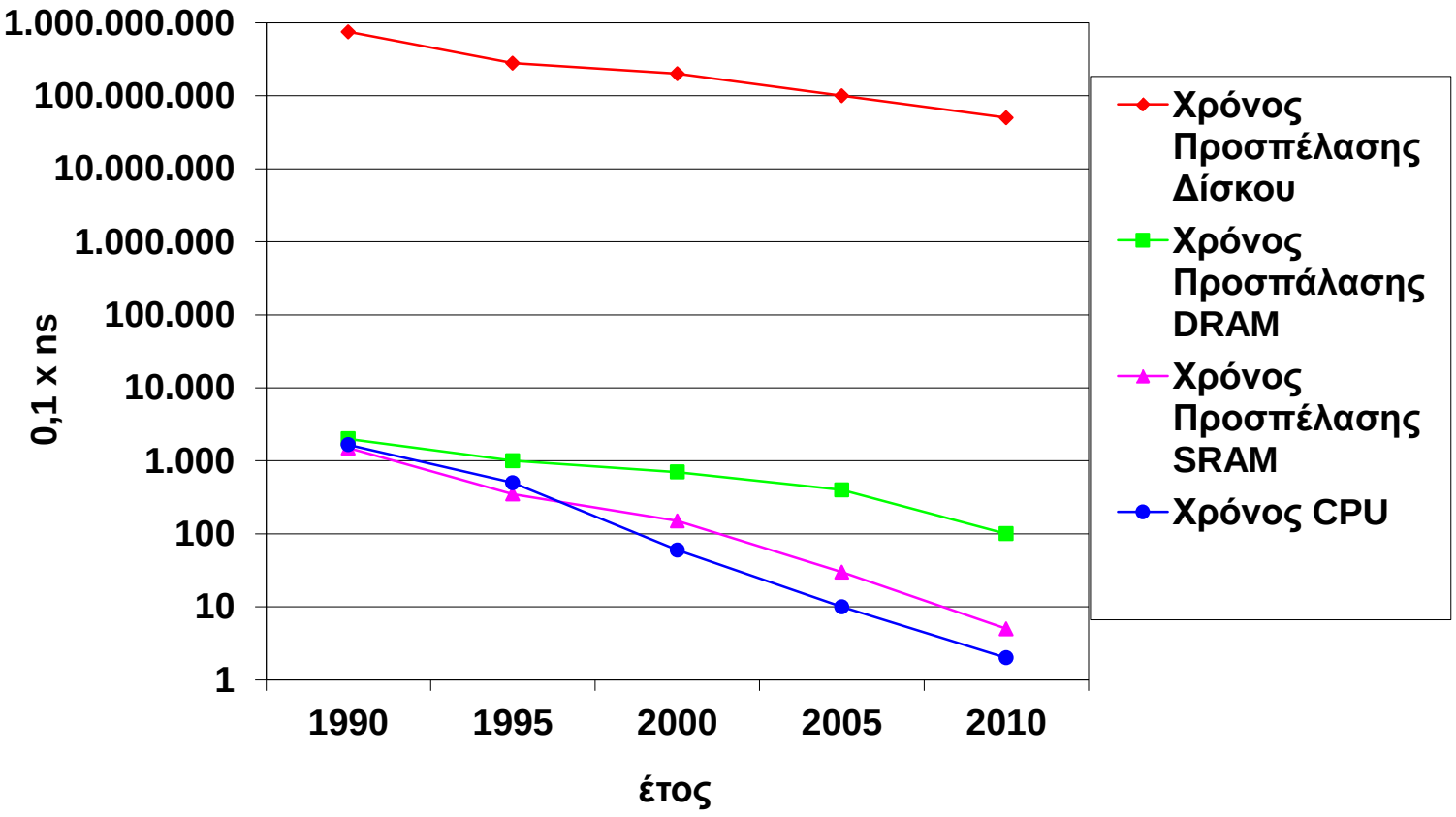
Advanced Data Indexing
(Προηγμένη ευρετηρίαση δεδομένων)

Ιεραρχίες Μνήμης – Δευτερεύουσα Μνήμη

Το Κενό Μεταξύ CPU και Μνήμης



Τοπικότητα

Υπάρχουν εντολές σε προγράμματα που κάνουν κακό χειρισμό της τοπικότητας;

- Η Αρχή της Τοπικότητας:
  - Τα προγράμματα τείνουν να χρησιμοποιούν δεδομένα και εντολές που βρίσκονται «κοντά» χωρικά ή χρονικά σε αυτά που χρησιμοποιήσαμε πρόσφατα.
  - Χρονική Τοπικότητα: Στοιχεία που προσπελάσθηκαν πρόσφατα είναι πιθανό να προσπελασθούν πάλι στο κοντινό μέλλον.
  - Χωρική τοπικότητα: Τα στοιχεία που βρίσκονται σε κοντινές διευθύνσεις μνήμης τείνουν να προσπελαύνονται κοντά σε σχέση με το χρόνο.

Παράδειγμα Τοπικότητας:

- Δεδομένα
  - Προσπέλαση στοιχείων πίνακα στη σειρά: Χωρική
  - Αναφορά στο sum σε κάθε επανάληψη: Χρονική
- Εντολές
  - Προσπέλαση εντολών στη σειρά: Χωρική
  - Προσπέλαση εντολών σε επανάληψη: Χρονική

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

Παράδειγμα Τοπικότητας

- Ισχυρισμός: Αν κοιτώντας ένα κομμάτι κώδικα μπορείτε να πάρετε μία άποψη σχετικά με την τοπικότητα του τότε είστε καλοί προγραμματιστές!
- Ερώτηση: Έχει η παρακάτω συνάρτηση καλή τοπικότητα;

```
int sumarrayrows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum
}
```

Παράδειγμα Τοπικότητας

- Ερώτηση: Έχει η παρακάτω συνάρτηση καλή τοπικότητα;

```
int sumarraycols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];
    return sum
}
```

Τοπικότητα

- Εξαρτάται φυσικά από το πώς αποθηκεύεται ο πίνακας στην κύρια μνήμη (κατά γραμμές ή κατά στήλες, ή αν συμπιέζεται [Sparse Matrices], κλπ.)

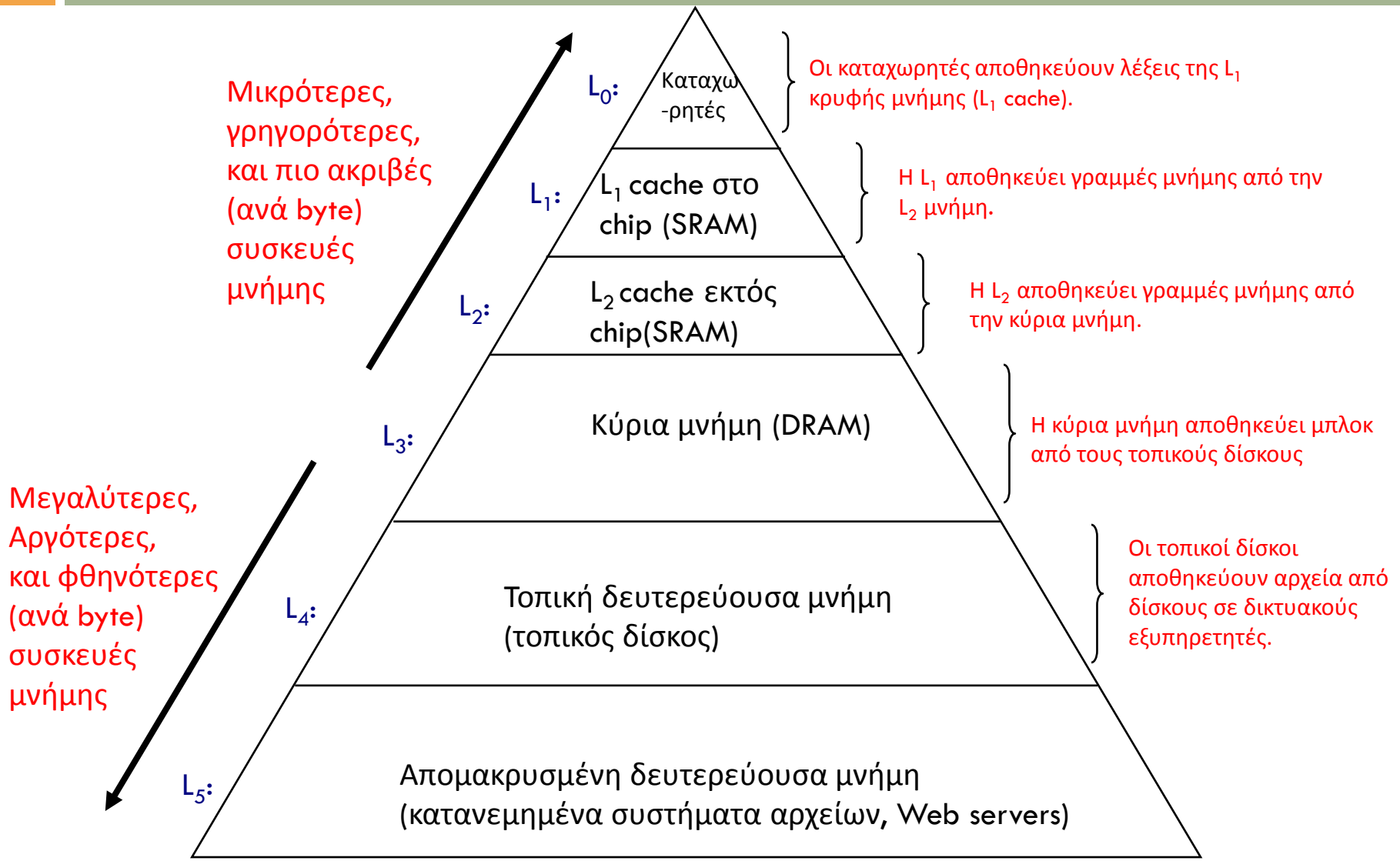
0	1	2	.	.	.	.	.	N-1
1								
2								
.								
.								
.								
M-1								

In row-major storage, a multidimensional array in linear memory is organized such that rows are stored one after the other. It is the approach used by the C programming language, among others.
Also used in the scientific programming languages Fortran and Julia, the matrix-oriented languages MATLAB, etc.

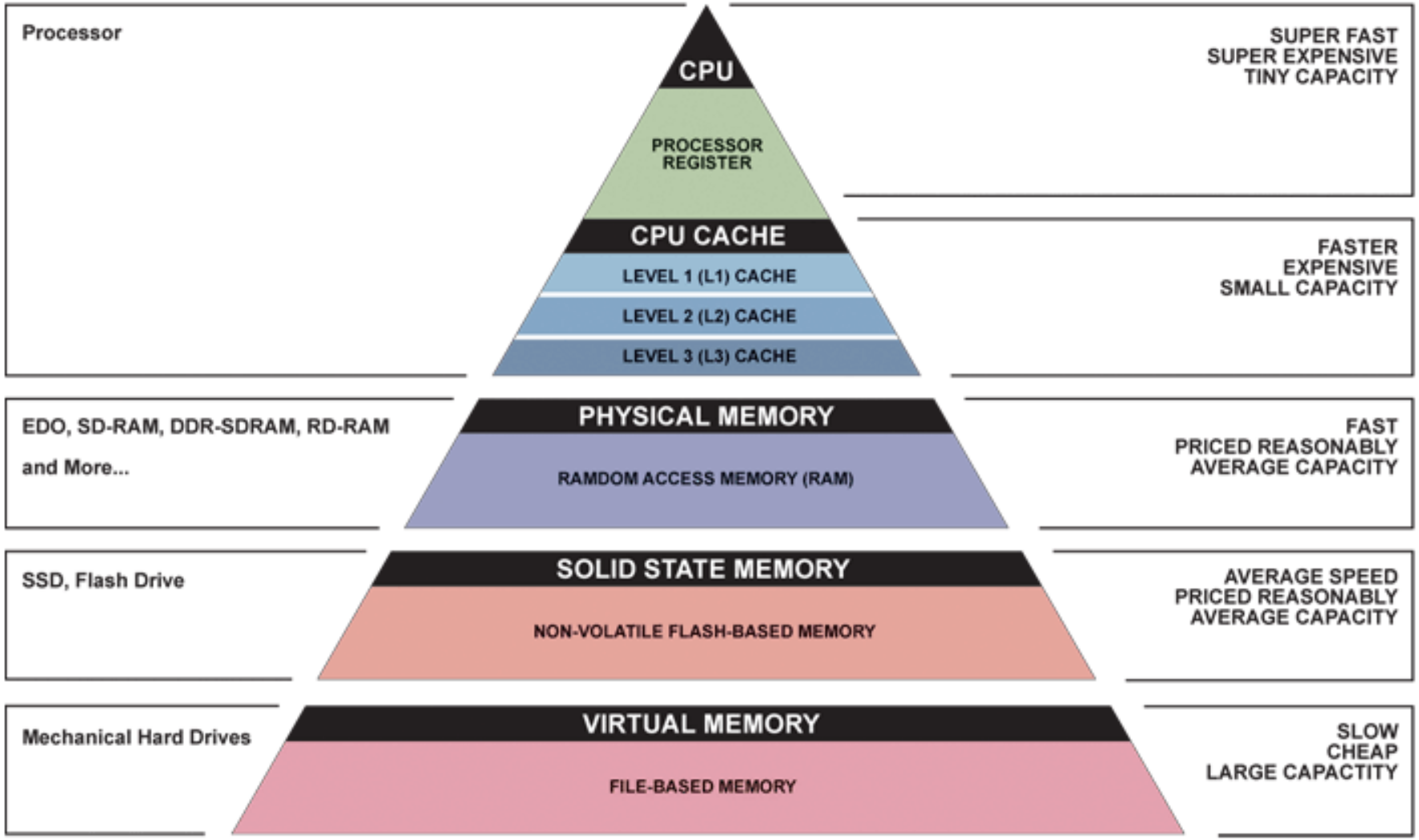
Ιεραρχία Μνήμης

- Μερικές θεμελιώδεις και διηνεκείς ιδιότητες του υλικού και του λογισμικού:
  - Οι μνήμες υψηλής ταχύτητας κοστίζουν περισσότερο ανά byte και έχουν μικρότερη χωρητικότητα.
  - Το κενό απόδοσης μεταξύ CPU και κύριας μνήμης μεγαλώνει ολοένα και περισσότερο.
  - Τα καλώς γραμμένα προγράμματα τείνουν να έχουν καλή τοπικότητα.
- Αυτές οι θεμελιώδεις ιδιότητες συμπληρώνουν η μία την άλλη.
- Υπονοούν ότι η καλύτερη οργάνωση του συστήματος μνήμης είναι ιεραρχική.

Ιεραρχία Μνήμης (2-Level Cache)

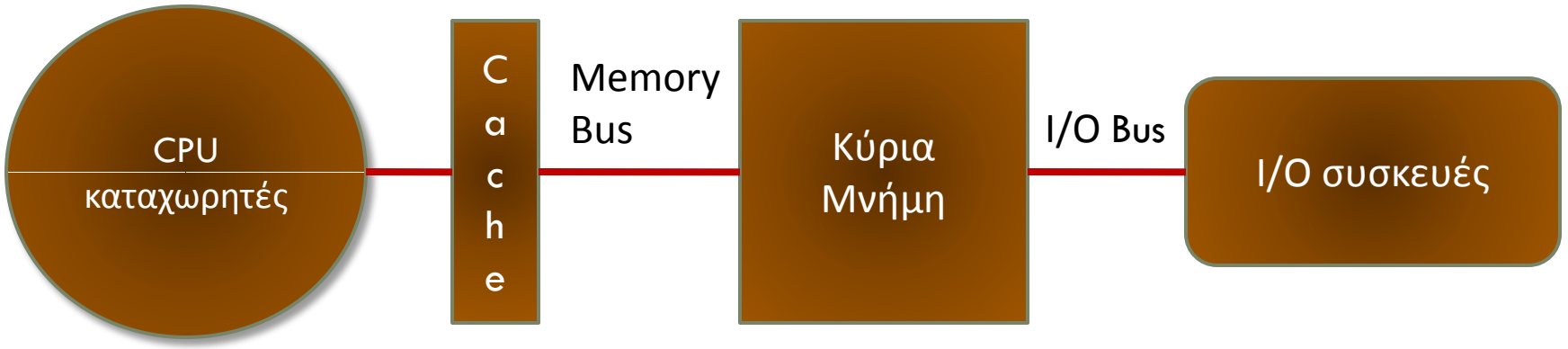


Ιεραρχία Μνήμης (3-Level Cache)



▲ Simplified Computer Memory Hierarchy
Illustration: Ryan J. Leng

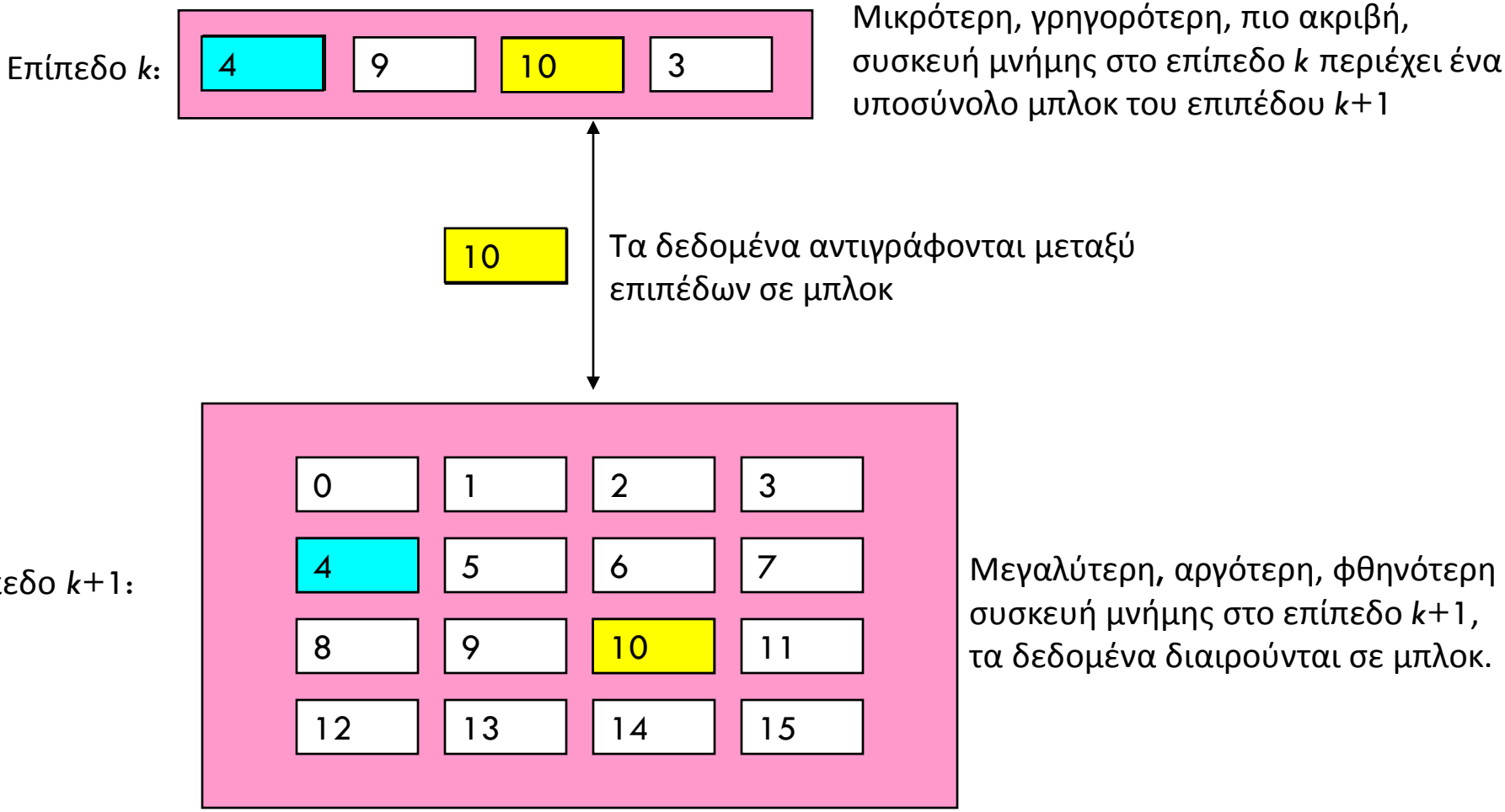
Ιεραρχία Μνήμης σε Τυπικούς Η/Υ



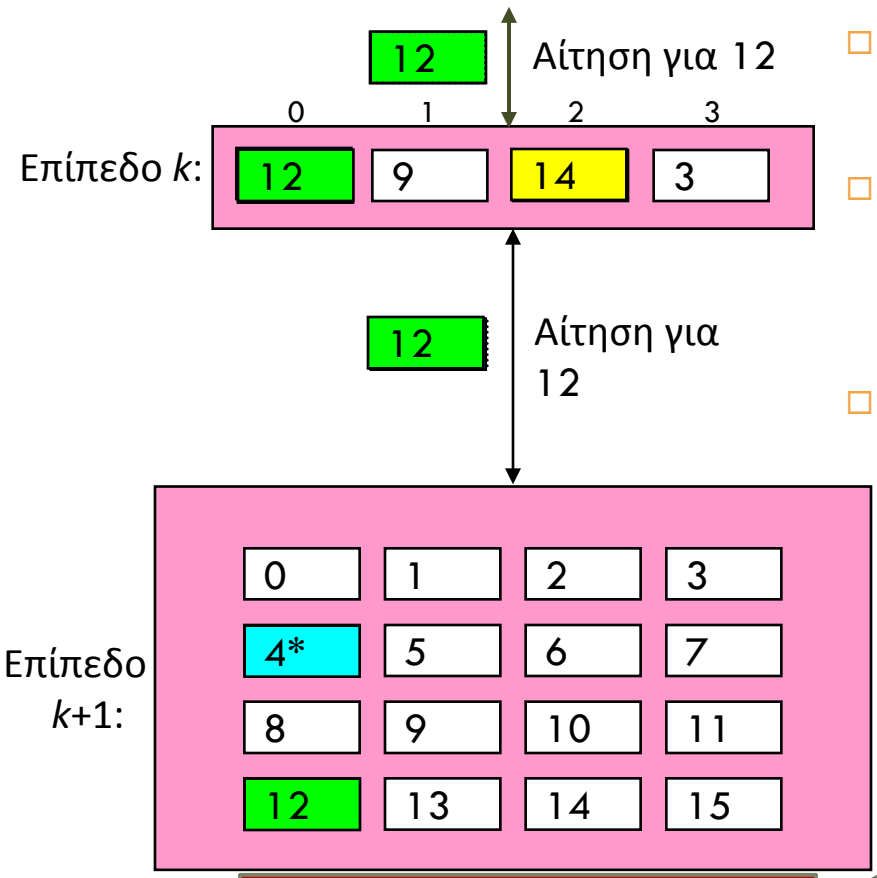
Μέγεθος:	1-8KB	512KB-8MB	1GB-16GB	100GB-1TB
Ταχύτητα:	0.25ns	1ns	30ns	8ms

Ιεραρχία Μνήμης

Το μπλοκ μεταξύ Κύριας Μνήμης και Cache λέγεται γραμμή Cache (cache line)



Μερικές Γενικές Έννοιες



- Το πρόγραμμα χρειάζεται το αντικείμενο *d*, που αποθηκεύεται σε κάποιο μπλοκ *b*.
- Cache Επιτυχία
  - Το πρόγραμμα βρίσκει το *b* στην μνήμη επιπέδου *k*. (π.χ. μπλοκ 14)
- Cache Αποτυχία
  - Το *b* δεν είναι στο επίπεδο *k*, και άρα πρέπει να το φέρουμε από το *k*+1. (π.χ. μπλοκ 12)
  - Αν η μνήμη επιπέδου *k* είναι γεμάτη, τότε κάποιο μπλοκ πρέπει να αντικατασταθεί. Ποιο είναι το «θύμα»;
    - Πολιτική Τοποθέτησης: που θα πάει το νέο μπλοκ; (π.χ. με *b* mod 4 θύμα το 4 στη θέση 0)
    - Πολιτική Αντικατάστασης: ποιο μπλοκ πρέπει να αντικατασταθεί; (π.χ. LRU)

1. Πλήρως συσχετίσιμη
2. Μερικώς συσχετίσιμη
3. Ευθέως συσχετίσιμη

Πολιτική Τοποθέτησης

- Πολιτική τοποθέτησης:
  - Πλήρως συσχετίσιμη:
    - Το νέο μπλοκ μπορεί να πάει οπουδήποτε στην cache
  - Μερικώς συσχετίσιμη:
    - Το νέο μπλοκ μπορεί να πάει μόνο σε συγκεκριμένες θέσεις στην cache, N-way associative (σε N θέσεις).
  - Ευθέως συσχετίσιμη:
    - Το νέο μπλοκ μπορεί να πάει μόνο σε μία συγκεκριμένη θέση στην cache.

Πολιτική Τοποθέτησης (παράδειγμα)

32-Block Χώρος Μνήμης:

Επίπεδο *k*+1:

1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Ευθέως συσχ.:
Το 12 μπορεί να πάει στο μπλοκ 4 (12 mod 8)

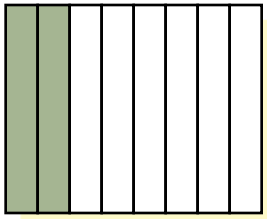
Μερικώς συσχ.:
Το 12 πάει οπουδήποτε στο σύνολο 0 (12 mod 4)

Πλήρως συσχ. :
Το 12 μπορεί να πάει οπουδήποτε

Επίπεδο *k*:

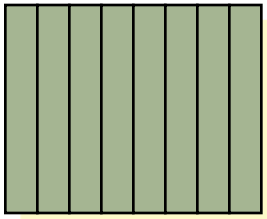
0 1 2 3 4 5 6 7

0 1 2 3 4 5 6 7



0 1 2 3

0 1 2 3 4 5 6 7



Μερικές Γενικές Έννοιες

- Τύποι αποτυχιών:
  - Αναγκαστική αποτυχία:
    - Συμβαίνει επειδή η μνήμη είναι άδεια.
  - Αποτυχία Σύγκρουσης
    - Συμβαίνει όταν έχουμε περιορισμένη συσχέτιση.
      - Π.χ. το μπλοκ *i* στο επίπεδο *k*+1 πρέπει να τοποθετηθεί στο μπλοκ (*i* mod 4) στο επίπεδο *k*. Αναφορά στα μπλοκ 0, 8, 4, 8, 0, 4, ... θα έδινε αποτυχία κάθε φορά.
  - Αποτυχία Χωρητικότητας
    - Συμβαίνει όταν το σύνολο των ενεργών μπλοκ (σύνολο εργασίας – working set) είναι μεγαλύτερο από την μνήμη επιπέδου *k*.

Πολιτική Αντικατάστασης

- Μερικοί Αλγόριθμοι αντικατάστασης:
  - LRU (Least Recently Used):
    - Αντικατάσταση του παλαιότερα χρησιμοποιημένου στοιχείου.
  - MRU (Most Recently Used):
    - Αντικατάσταση του πιο πρόσφατα χρησιμοποιημένου στοιχείου.
  - LFU (Least Frequently Used):
    - Αντικατάσταση του πιο σπάνια χρησιμοποιημένου στοιχείου.
  - RR (Random Replacement):
    - Τυχαία Αντικατάσταση.
- http://en.wikipedia.org/wiki/Cache_algorithms

Χαρακτηριστικά ενός Η/Υ Dell 650

www.dell.com

www.intel.com

Ταχύτητα CPU	2.4 – 3.2 GHz
L3 cache μέγεθος	0.5 – 2 MB
Μνήμη	1/4 – 4 GB
Δίσκος	36 GB– 146 GB
	7.200 – 15.000 RPM
CD/DVD	8 – 48x
L2 cache μέγεθος	256 – 512 KB
L2 cache line μέγεθος	128 Bytes
L1 cache line μέγεθος	64 Bytes
L1 cache μέγεθος	16 KB

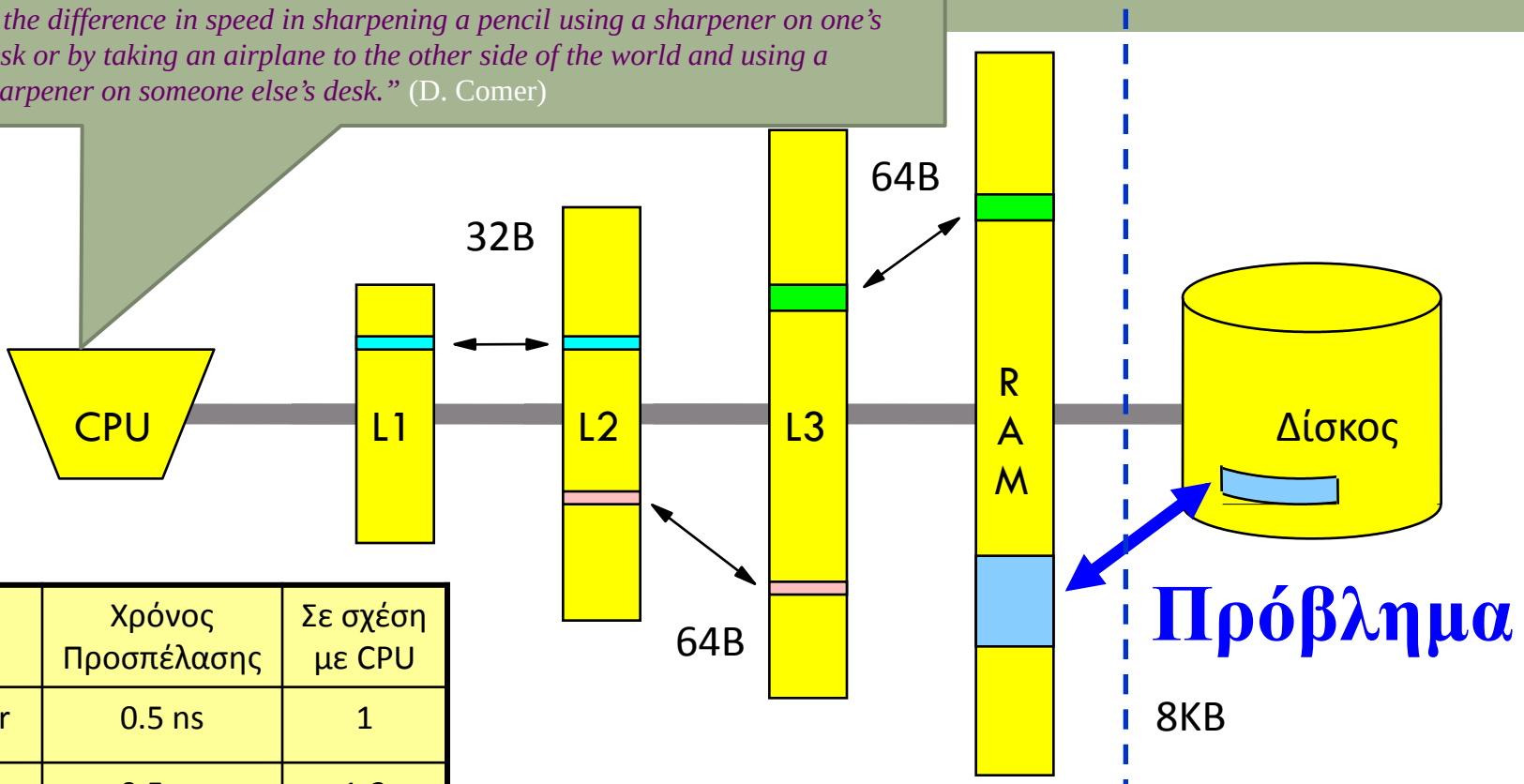
Dell's homepage July 2004

Οι Αλγοριθμικές Προκλήσεις

- Το υλικό δεν είναι ομοιόμορφο – πολλές παράμετροι
  - Πλήθος επιπέδων μνήμης
  - Μέγεθος cache
  - Cache γραμμή/μέγεθος μπλοκ δίσκου
  - Συσχέτιση Cache
  - Cache πολιτική αντικατάστασης
  - Ταχύτητα CPU/BUS/μνήμης…
- Τα προγράμματα θα πρέπει ιδεατά να τρέχουν για διαφορετικές τιμές παραμέτρων
  - Γνωρίζοντας τις παραμέτρους κατά το χρόνο εκτέλεσης ή
  - Γνωρίζοντας εξαρχής μερικές βασικές παραμέτρους ή
  - Αγνοώντας την μνήμη ιεραρχίας ← **Πράξη**
- Τα προγράμματα εκτελούνται σε απρόβλεπτες συνθήκες
  - Γενικές βιβλιοθήκες λογισμικού
  - Κώδικας που κατεβαίνει από το διαδίκτυο, Java applets
  - Δυναμικά περιβάλλοντα, π.χ. πολλαπλές διεργασίες

Κάποια Βασικά για τις Ιεραρχικές Μνήμες

"The difference in speed between modern CPU and disk technologies is analogous to the difference in speed in sharpening a pencil using a sharpener on one's desk or by taking an airplane to the other side of the world and using a sharpener on someone else's desk." (D. Comer)



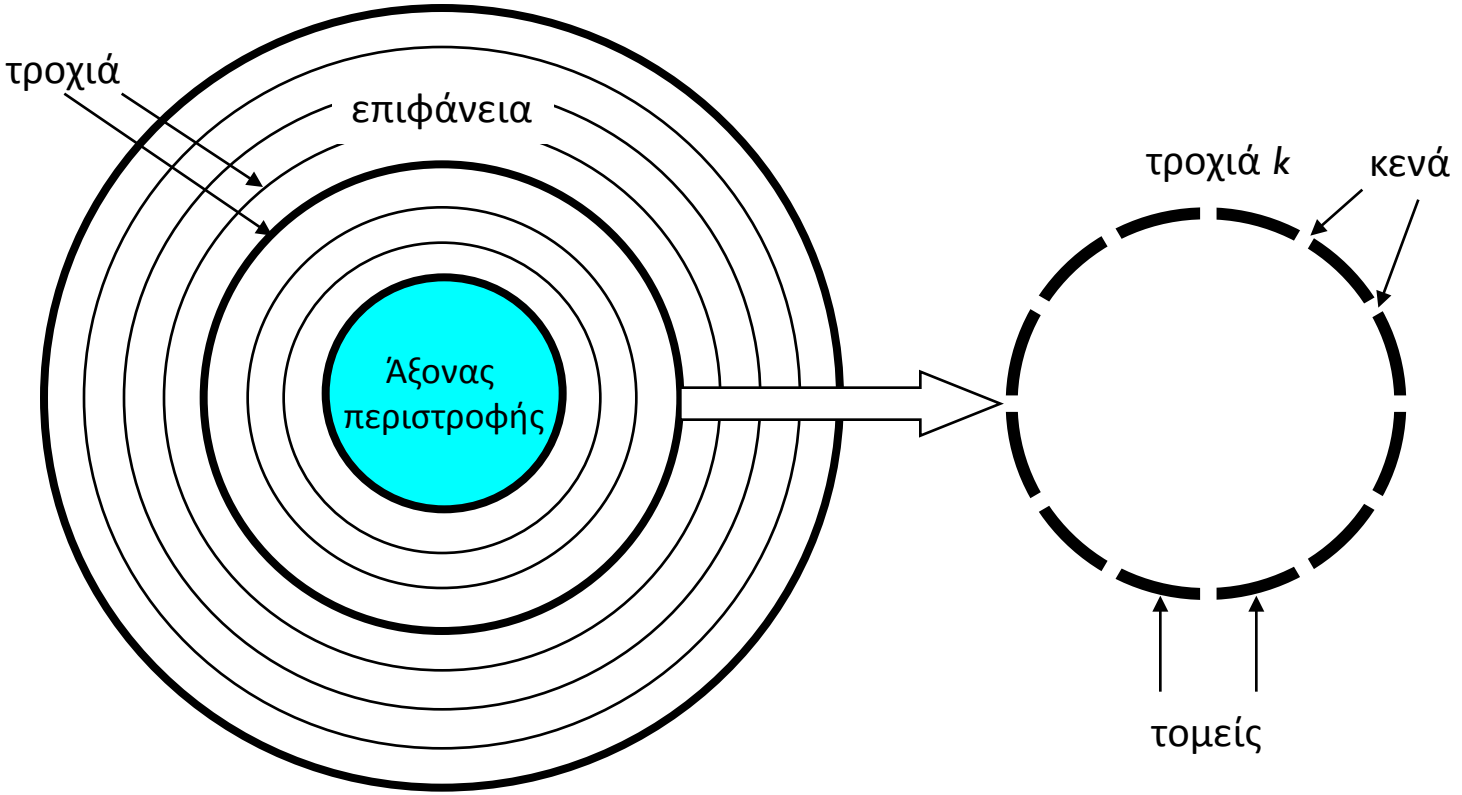
	Χρόνος Προσπέλασης	Σε σχέση με CPU
Register	0.5 ns	1
L1 cache	0.5 ns	1-2
L2 cache	3 ns	2-7
DRAM	50 ns	80-200
Δίσκος	10 ms	10^7

Αυξανόμενος χρόνος προσπέλασης και μέγεθος

Αυξημένο

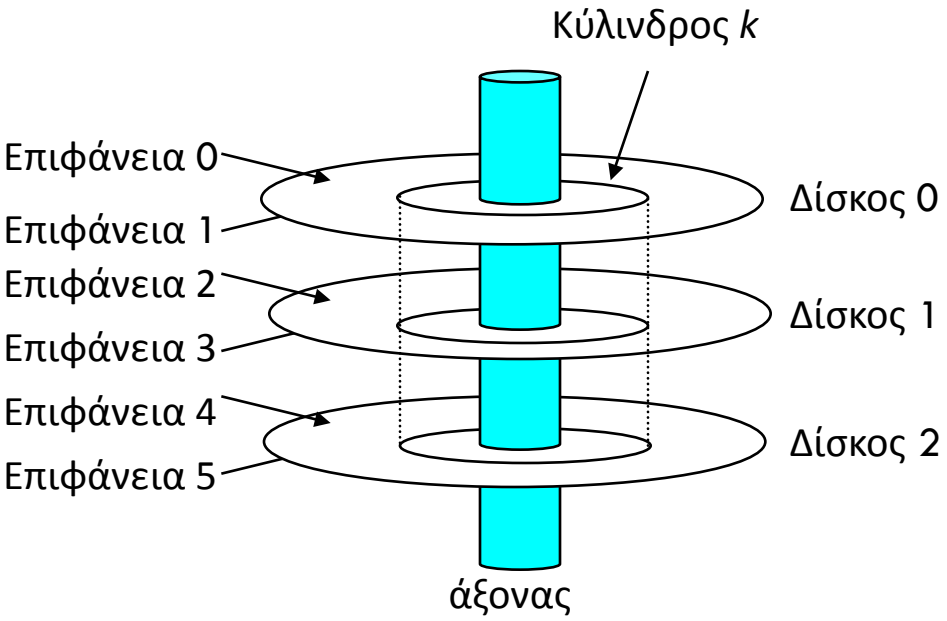
Γεωμετρία Δίσκου (με κινητά μέρη)

- Οι σκληροί δίσκοι αποτελούνται από έναν δίσκο, με δύο επιφάνειες.
- Κάθε επιφάνεια αποτελείται από ομόκεντρους κύκλους, που ονομάζονται τροχιές (tracks) ή αυλάκια.
- Κάθε τροχιά αποτελείται από τομείς (sectors) που χωρίζονται με κενά.



Γεωμετρία Δίσκου (Πολλαπλοί Δίσκοι)

- Οι ευθυγραμμισμένες τροχιές δημιουργούν κυλίνδρους.



Χωρητικότητα Σκληρού Δίσκου

- Χωρητικότητα: μέγιστο πλήθος bytes που μπορούν να αποθηκευτούν.
  - Σε Gigabytes (GB $\approx 10^9$ bytes) ή Terabytes (TB $\approx 10^{12}$ bytes).
- Η χωρητικότητα καθορίζεται από τις εξής τεχνολογικές παραμέτρους:
  - Πυκνότητα αποθήκευσης (bits/in): πλήθος bits που αποθηκεύονται σε 1 ίντσα μίας τροχιάς.
  - Πυκνότητα τροχιών (tracks/in): πλήθος τροχιών που υπάρχουν σε 1 ίντσα πάνω στην ακτίνα του δίσκου.
  - Πυκνότητα Επιφάνειας (bits/in^2): γινόμενο των δύο παραπάνω.
- Οι μοντέρνοι δίσκοι χωρίζουν τις τροχιές σε ξένα υποσύνολα που ονομάζονται ζώνες εγγραφής.
  - Κάθε τροχιά σε μία ζώνη έχει το ίδιο πλήθος τομέων, που καθορίζεται από την διάμετρο της πιο εσωτερικής τροχιάς.
  - Κάθε ζώνη έχει διαφορετικό πλήθος τομέων ανά τροχιά.

Παράδειγμα

- Χωρητικότητα = (# bytes/τομέα) x (μέσο # τομέων/τροχιά) x
  (# τροχιές/επιφάνεια) x (# επιφάνεια/δίσκο) x
  (# δίσκοι/σκληρό δίσκο)

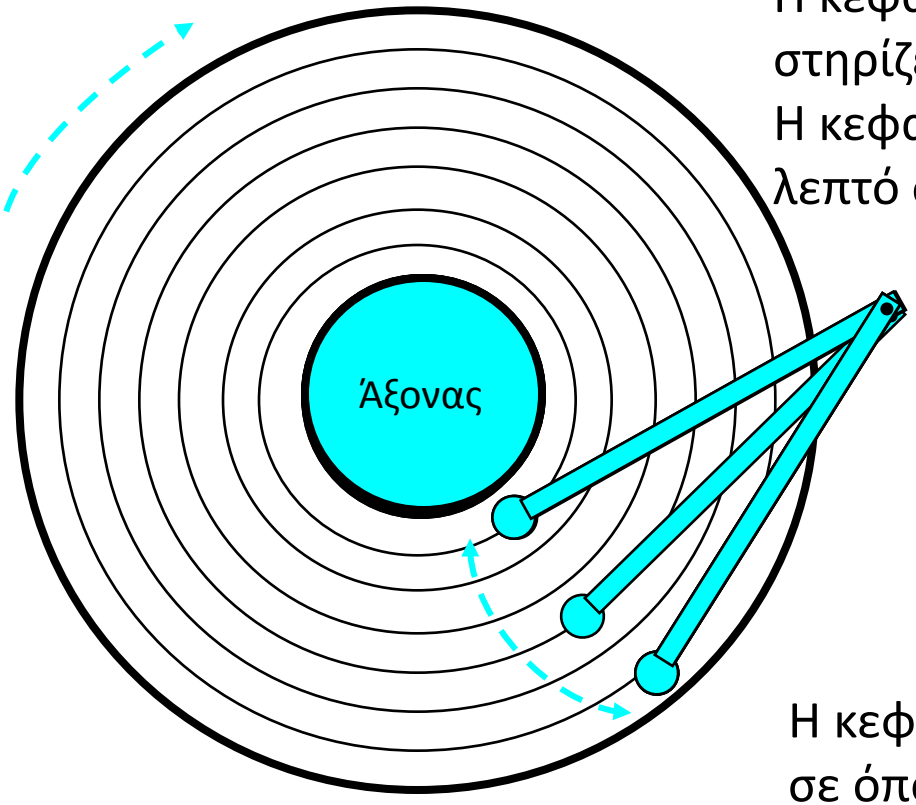
Παράδειγμα:

- 512 bytes/τομέα
- 300 τομείς/τροχιά (κατά μέσο όρο)
- 20,000 τροχιές/επιφάνεια
- 2 επιφάνειες/δίσκο
- 5 δίσκοι/σκληρό δίσκο

- Χωρητικότητα = 512 x 300 x 20000 x 2 x 5 bytes
  = 30,720,000,000 bytes
  ≈ 28.61 GB

Λειτουργία Δίσκου

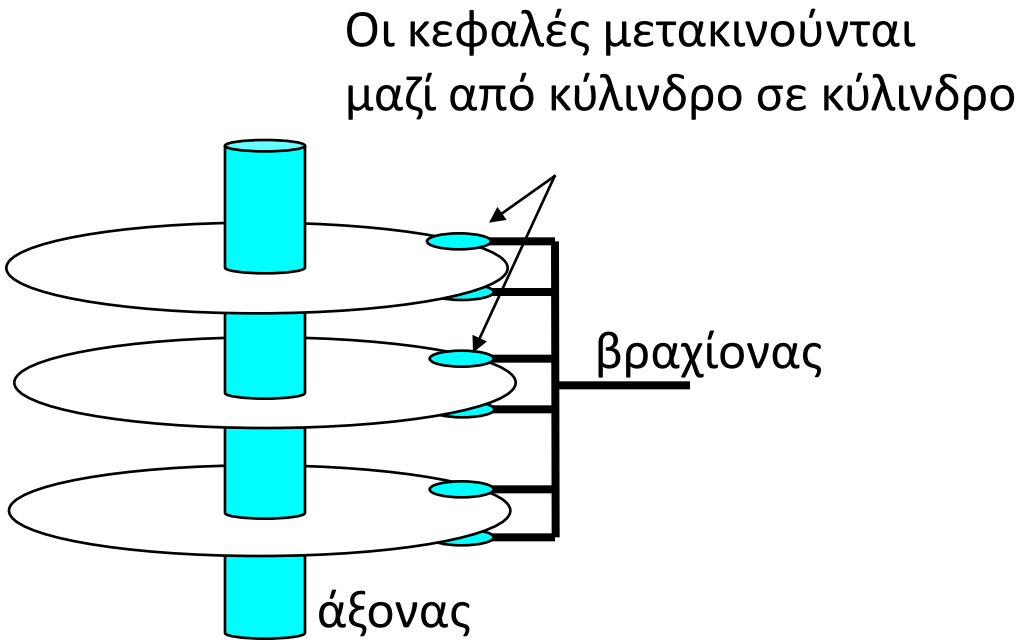
Η επιφάνεια του δίσκου κινείται με σταθερή ταχύτητα.



Η κεφαλή εγγραφής/ανάγνωσης στηρίζεται σε έναν βραχίονα. Η κεφαλή κινείται πάνω σε ένα λεπτό στρώμα αέρα.

Η κεφαλή μπορεί να πάει σε όποια θέση θέλει πάνω στο δίσκο.

Λειτουργία Δίσκου (Πολλαπλοί Δίσκοι)



Χρόνος Προσπέλασης Δίσκου

- Μέσος χρόνος προσπέλασης ενός τομέα προσεγγίζεται από:
  - $T_{προσπέλασης} = T_{μέσης\text{-}αναζήτησης} + T_{μέσης\text{-}περιστροφής} + T_{μέσης\text{-}μεταφοράς}$
- Χρόνος Αναζήτησης ($T_{μέσης\text{-}αναζήτησης}$)
  - Χρόνος για τοποθέτηση κεφαλών πάνω από τον κατάλληλο κύλινδρο που περιέχει τον τομέα που θέλουμε.
  - Τυπική τιμή $T_{μέσης\text{-}αναζήτησης}$ = 9 ms
- Καθυστέρηση Περιστροφής ($T_{μέσης\text{-}περιστροφής}$)
  - Ο χρόνος αναμονής ώστε το πρώτο bit του τομέα να περάσει κάτω από την κεφαλή.
  - $T_{μέσης\text{-}περιστροφής}$ = 1/2 x 1/RPMs x 60 sec
- Ταχύτητα Μεταφοράς ($T_{μέσης\text{-}μεταφοράς}$)
  - Χρόνος για να διαβαστούν τα bits από τον τομέα.
  - $T_{μέσης\text{-}μεταφοράς}$ = 1/RPM x 1/(μέσο πλήθος sectors/track) x 60 secs

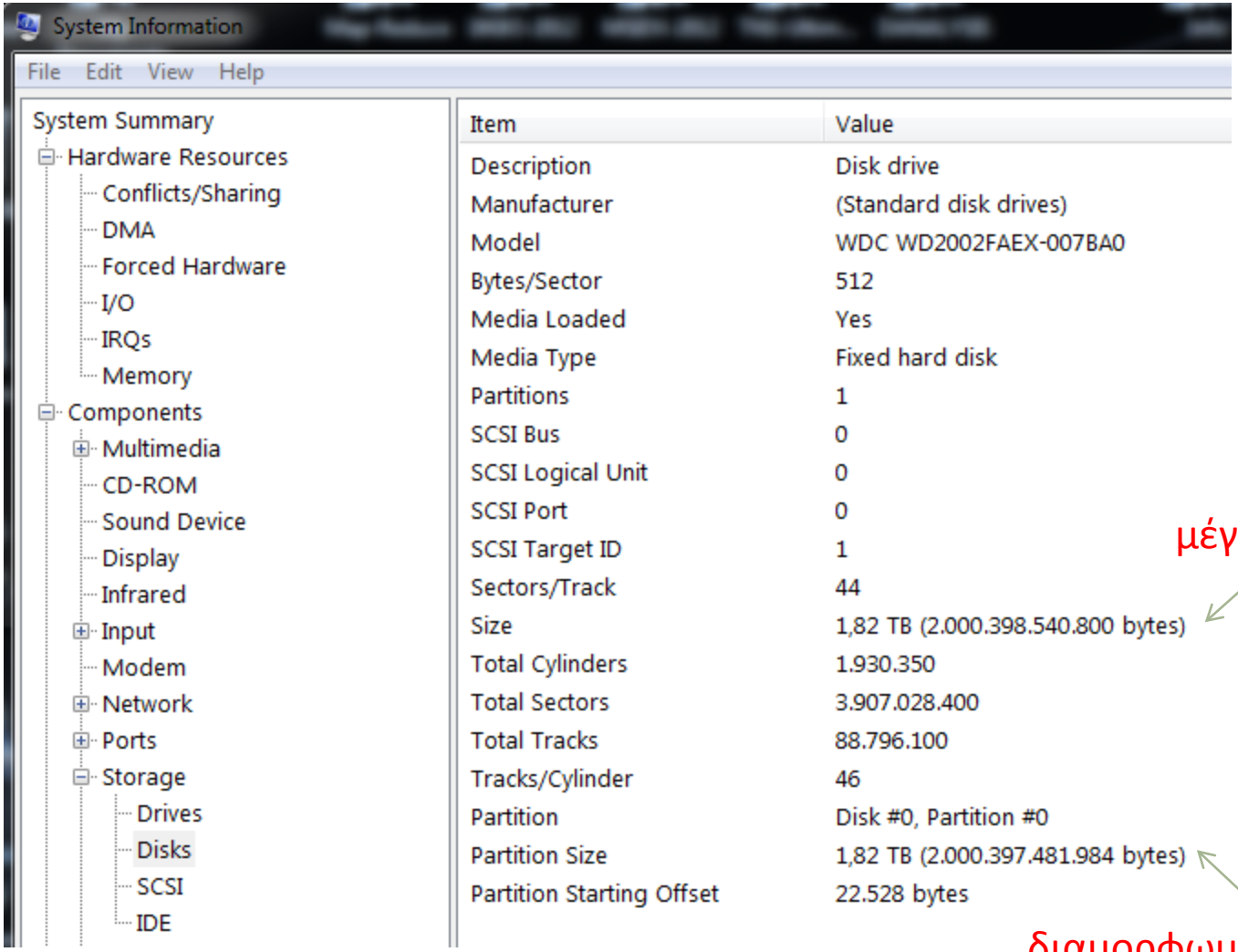
Παράδειγμα

- Μας δίνονται:
  - Ταχύτητα Περιστροφής = 10,000 RPM
  - Μέσος χρόνος αναζήτησης = 9 ms
  - Μέσο πλήθος τομέων/τροχιά = 400
- Υπολογίζουμε:
  - $T_{\text{μέσης-περιστροφής}}$= 1/2 x (60 secs/10000 RPM) x 1000 ms/sec = 3 ms
  - $T_{\text{μέσης-μεταφοράς}}$= 60/10000 RPM x 1/400 sec/τροχιά x 1000 ms/sec = 0.015 ms
  - $T_{\text{προσπέλασης}}$= 9 ms + 3 ms + 0.015 ms
- Σημαντικά σημεία:
  - Ο χρόνος προσπέλασης κυριαρχείται από το χρόνο αναζήτησης και τον χρόνο περιστροφής.
  - Το πρώτο bit του τομέα είναι το πιο ακριβό, τα άλλα είναι σχετικά δωρεάν.

Λογικά Μπλοκ Δίσκου

- Οι μοντέρνοι δίσκοι δίνουν μία αφαιρετική άποψη αυτής της πολύπλοκης γεωμετρίας:
  - Το πλήθος των τομέων μοντελοποιείται σαν μία ακολουθία από ***λογικά μπλοκ*** μεγέθους *b*.
- Αντιστοίχηση μεταξύ λογικών μπλοκ και πραγματικών τομέων
  - Αυτή η αντιστοίχηση πραγματοποιείται στον **ελεγκτή δίσκου**.
  - Μετατρέπει αιτήσεις σε μορφή λογικών μπλοκ σε τριάδες της μορφής (επιφάνεια, τροχιά, τομέας).
- Επιτρέπει στον ελεγκτή να βάλει στην άκρη μερικούς κυλίνδρους για κάθε ζώνη
  - Έτσι προκύπτει η διαφορά μεταξύ «διαμορφωμένης χωρητικότητας» και «μέγιστης χωρητικότητας».

Παράδειγμα



μέγιστη χωρητικότητα

διαμορφωμένη χωρητικότητα

Δίσκοι Στερεάς Κατάστασης (SSD)

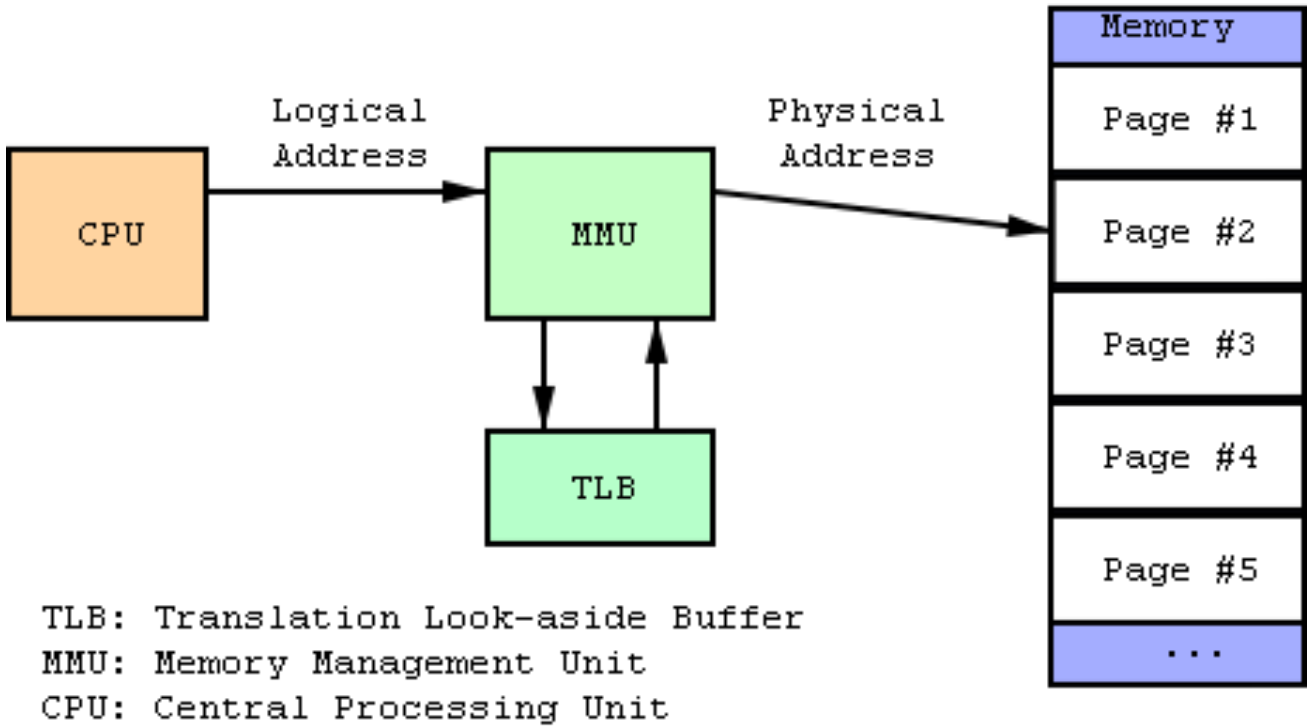
- Δεν έχουν μηχανικά μέρη.
- Χαμηλότεροι χρόνοι προσπέλασης (0.035-0.1 ms)
- Αθόρυβοι
- Πιο ακριβοί (νέα τεχνολογία)
  - NAND-based Flash μνήμη: περιορισμοί ως προς εγγραφή
  - RAM-based: απαιτούν μπαταρία

Αποδοτικότητα SSD

SSD benchmark, showing about 230 MB/s reading speed (blue), 210 MB/s writing speed (red) and about 0.1 ms seek time (green), all independent from the accessed disk location.

Μία σύγκριση στο: http://www.storagesearch.com/bitmicro-art3.html

Ιδεατή Μνήμη – Virtual Memory



Χρήσιμες Πληροφορίες

- Σχετικά με μηχανικούς Σκληρούς Δίσκους: http://en.wikipedia.org/wiki/Hard_disk_drive
- Σχετικά με Δίσκους Στερεάς Κατάστασης (SSD): http://en.wikipedia.org/wiki/Solid-state_drive
- Σχετικά με τις Ιεραρχίες Μνήμης: http://en.wikipedia.org/wiki/Memory_hierarchy
- Σχετικά με την επίδραση των Cache: http://igoro.com/archive/gallery-of-processor-cache-effects/

Παράδειγμα για την Ιεραρχία Μνήμης

Πόσο πιο γρήγορα θα τρέξει το Loop-1 σε σχέση με το Loop-2;

```
int[] arr = new int[64 * 1024 * 1024];

// Loop 1
for (int i = 0; i < arr.Length; i++) arr[i] *= 3;

// Loop 2
for (int i = 0; i < arr.Length; i += 16) arr[i] *= 3;
```

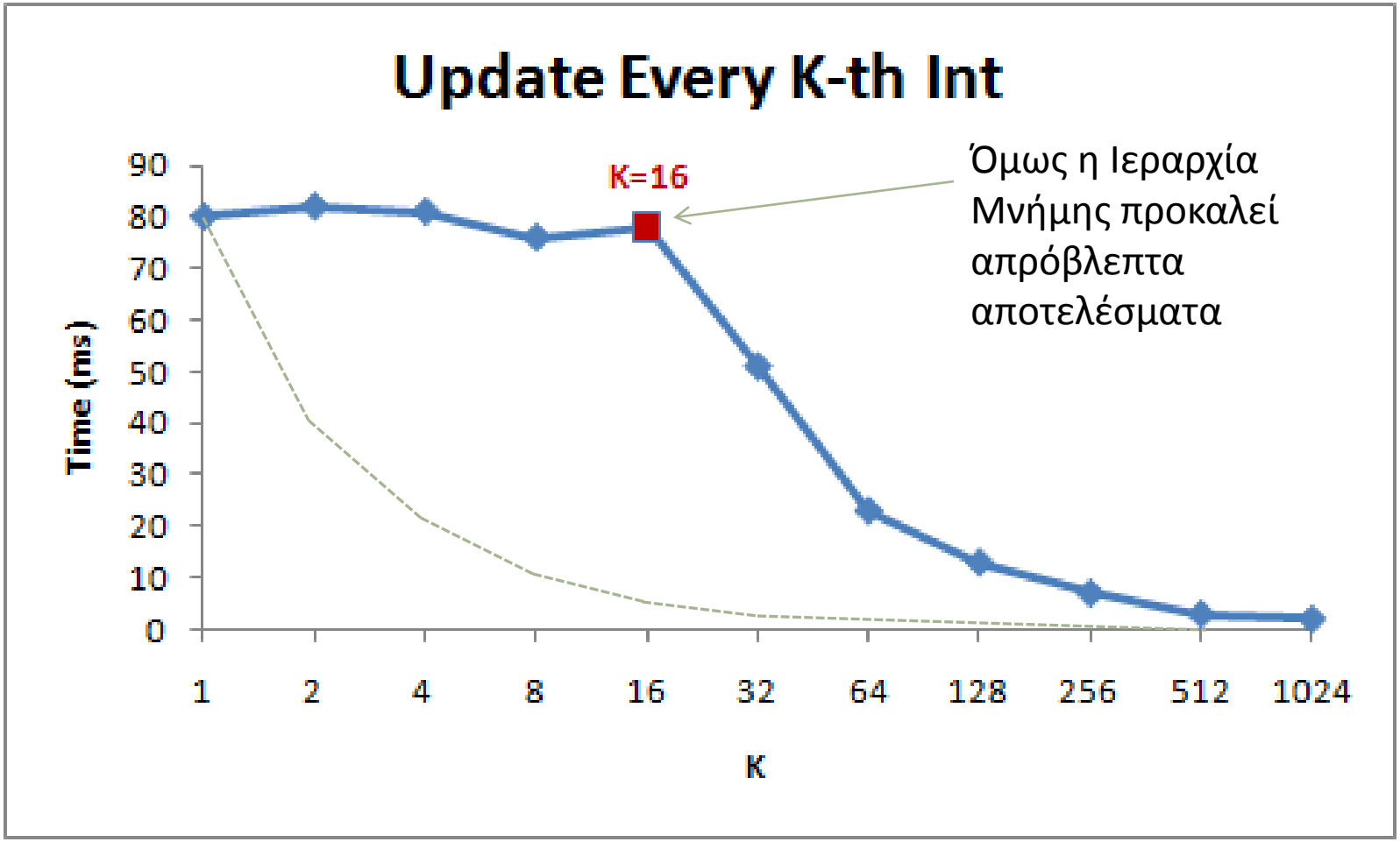
Παράδειγμα για την Ιεραρχία Μνήμης

- Το Loop-1 κάνει προσπέλαση σε κάθε στοιχείο του πίνακα, ενώ το Loop-2 σε κάθε 16ο στοιχείο.

- Εφόσον λοιπόν το Loop-2 κάνει μόνο το 6.25% της δουλειάς που κάνει το Loop-1 θα έπρεπε να κάνει και το 6.25% του αντίστοιχου χρόνου.

Παράδειγμα για την Ιεραρχία Μνήμης

for (int i = 0; i < arr.Length; i += K) arr[i] *= 3;



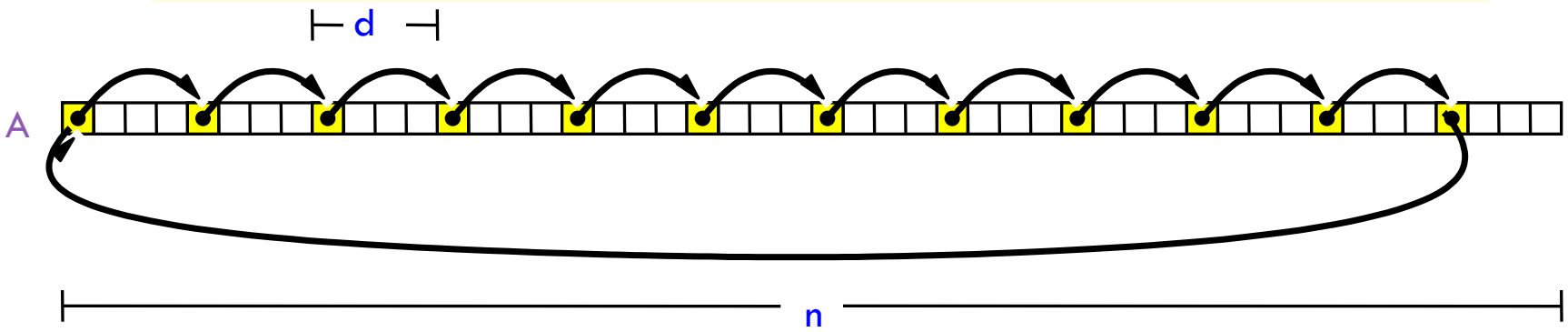
Παράδειγμα για την Ιεραρχία Μνήμης

- Παρατηρούμε ότι όταν το βήμα μεταβάλλεται από 1 μέχρι 16 ο χρόνος εκτέλεσης σχεδόν δεν αλλάζει.
- Μετά από 16 μειώνεται σχεδόν στο μισό κάθε φορά.
- Η αιτία είναι ότι η CPU δεν επεξεργάζεται την μνήμη byte προς byte, αλλά σε τμήματα των 64 bytes (συνήθως) που λέγονται cache lines.
- Οι 16 ακέραιοι (int) είναι 64 bytes (μία cache line).
- Άρα με βήμα ως 16 γίνεται προσπέλαση στο ίδιο πλήθος cache lines, ενώ με βήμα 32 στο μισό πλήθος, με βήμα 64 στο ¼, κλπ.

Ένα απλό Πρόγραμμα

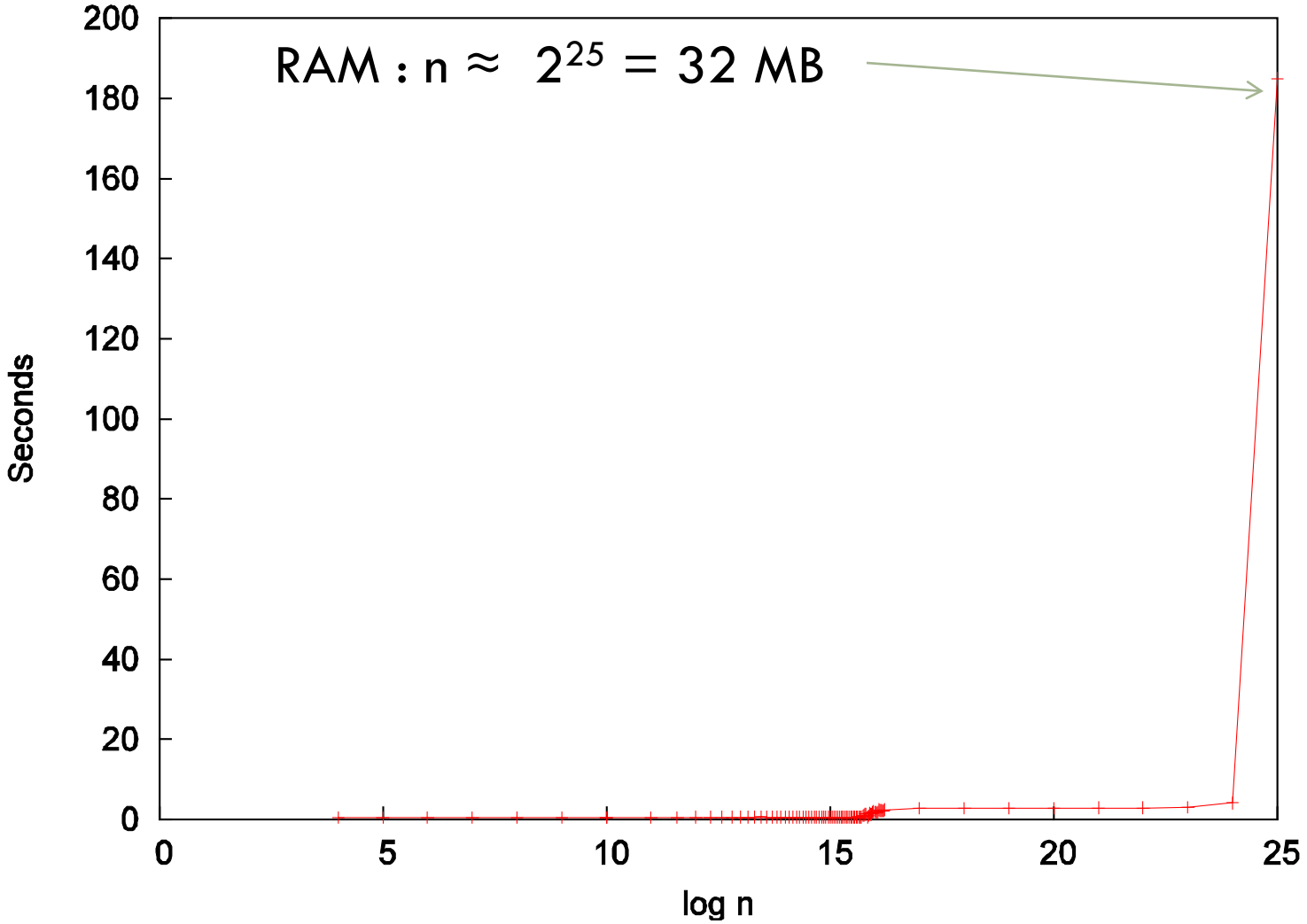
Διασχίζουμε πίνακα ακεραίων μεγέθους n κυκλικά προσπελάζοντας τις θέσεις του διαδοχικά ανά d (pointer chasing)

```
for (i=0; i+d<n; i+=d) A[i]=i+d;
A[i]=0;
for (i=0, j=0; j<8*1024*1024; j++) i = A[i];
```



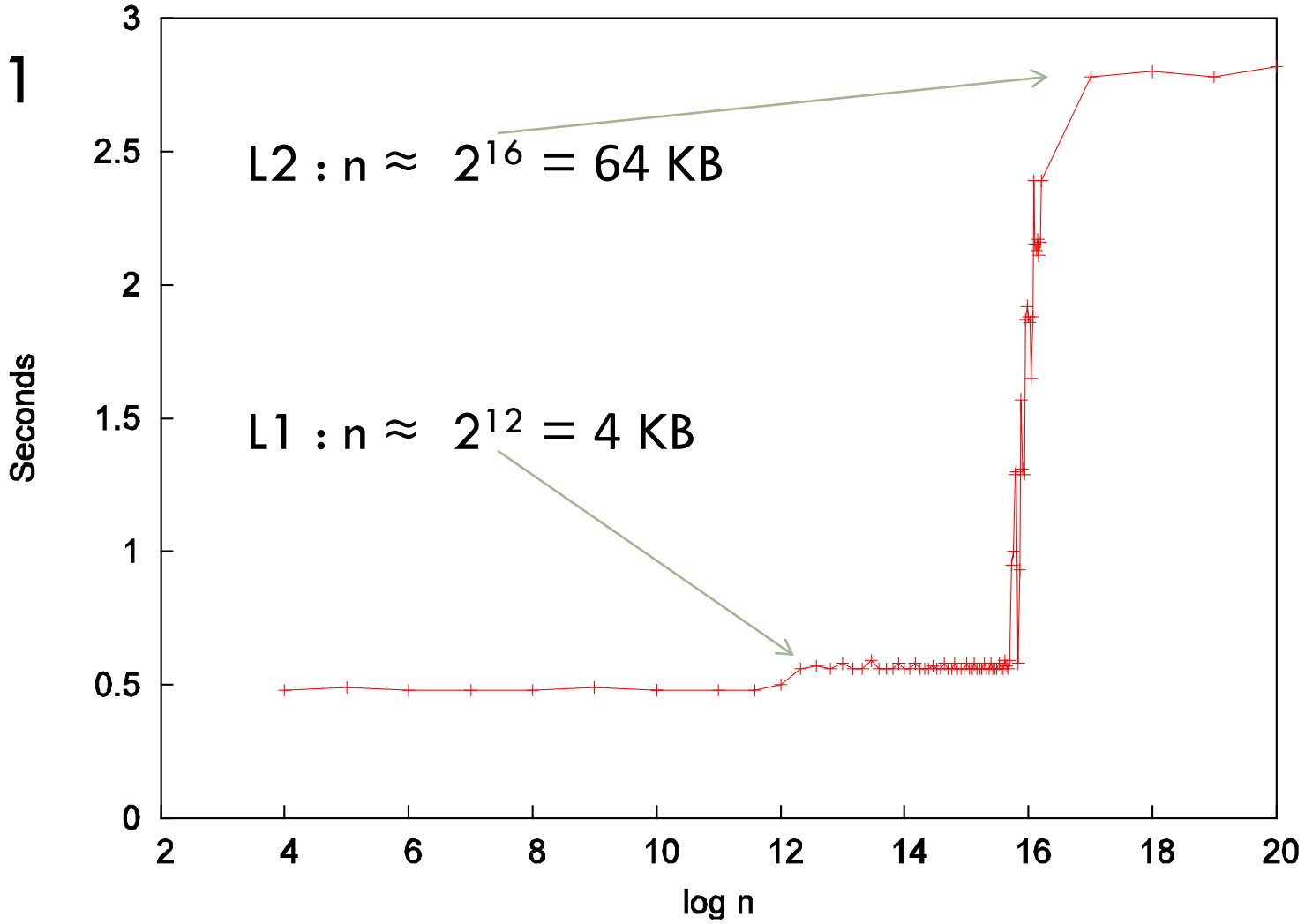
Παράδειγμα εκτίμησης μεγέθους RAM

d=1



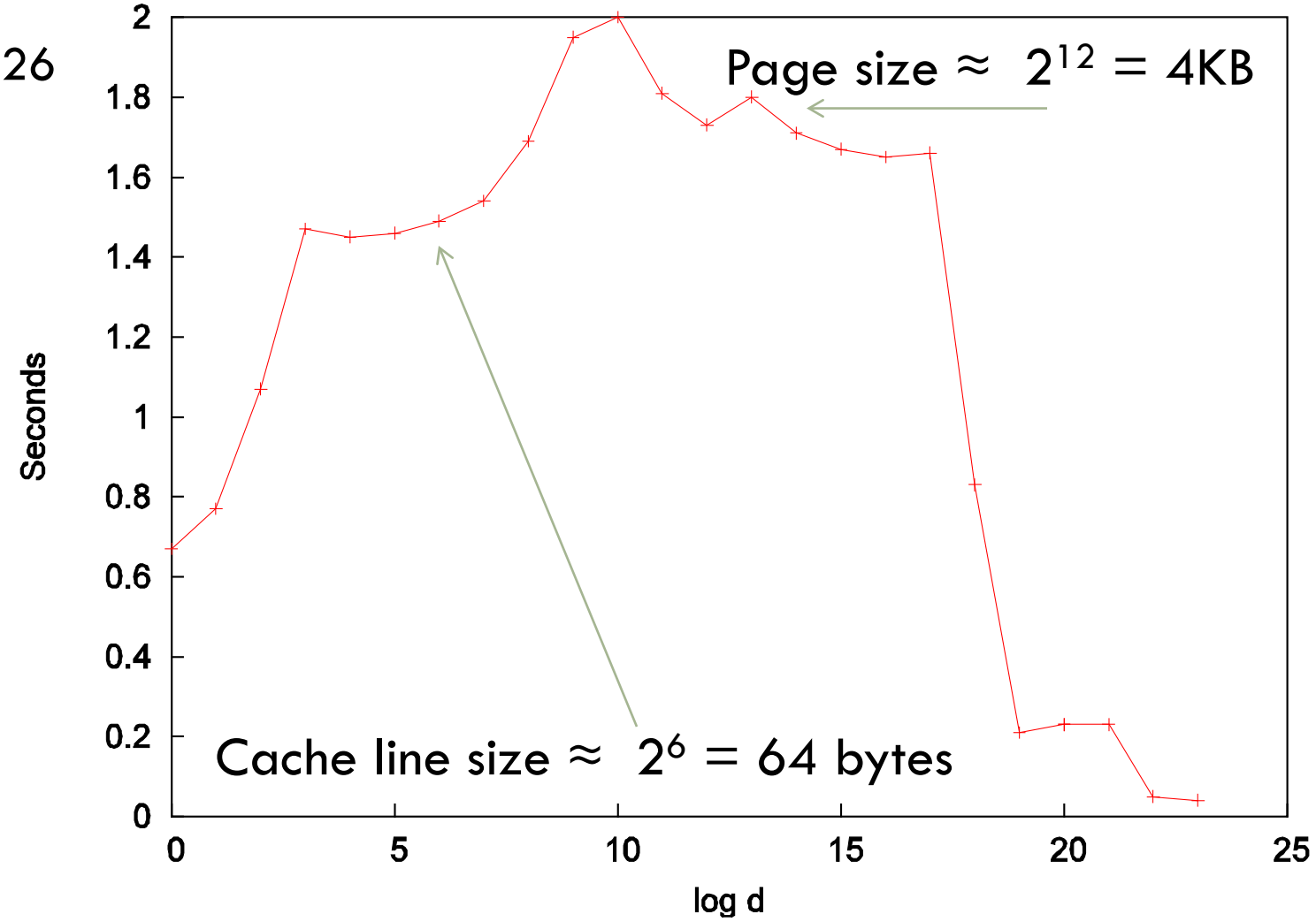
Παράδειγμα εκτίμησης μεγέθους L1,L2-Cache

d=1



Παράδειγμα εκτίμησης μεγέθους cache-line

$n=2^{26}$



Γενική Μεθοδολογία

- **Μέγεθος cache:** Όσον αφορά το μέγεθος της cache απλά σαρώνουμε πίνακες σταδιακά μεγαλύτερων μεγεθών (το ίδιο πλήθος προσπελάσεων) και όταν δούμε μία αλλαγή (μικρή) στους χρόνους τότε σημαίνει ότι περάσαμε από τη cache στη κύρια μνήμη. Προσοχή η μεταβολή είναι μικρή και πάντα να έχουμε υπόψη λογικές τιμές cache καθώς και άλλα φαινόμενα που επηρεάζουν τη διαδικασία.
- **L1 and L2 sizes:** Εδώ απλά φτιάχνουμε διαδοχικά μεγέθη πίνακα όπου το κάθε κελί δεικτοδοτεί το επόμενό του (με το τελευταίο να δεικτοδοτεί το πρώτο) και σαρώνουμε τον πίνακα υπολογίζοντας μέσο χρόνο προσπέλασης. Προσοχή και πάλι είναι δυσδιάκριτο που ακριβώς μεταβάλλεται η cache.
- **Line Size:** Εδώ κάνουμε ότι και ακριβώς παραπάνω μόνο που τώρα δεν δεικτοδοτεί το αμέσως επόμενο αλλά *d* θέσεις μακριά με στόχο να εντοπίσουμε τη πρώτη φορά που θα γίνονται πολλά cache misses. Το *d* είναι μικρό (~64bytes)
- **Cache speed:** Φτιάχνουμε διαδοχικά μεγαλύτερους πίνακες και τους προσπελαύνουμε πολλές φορές (το ίδιο όλους) προσπαθώντας να ελαχιστοποιήσουμε την επιρροή του cache line. Βέβαια, υποτίθεται ότι γνωρίζουμε τα σχετικά μεγέθη της cache.


ΤΕΛΟΣ