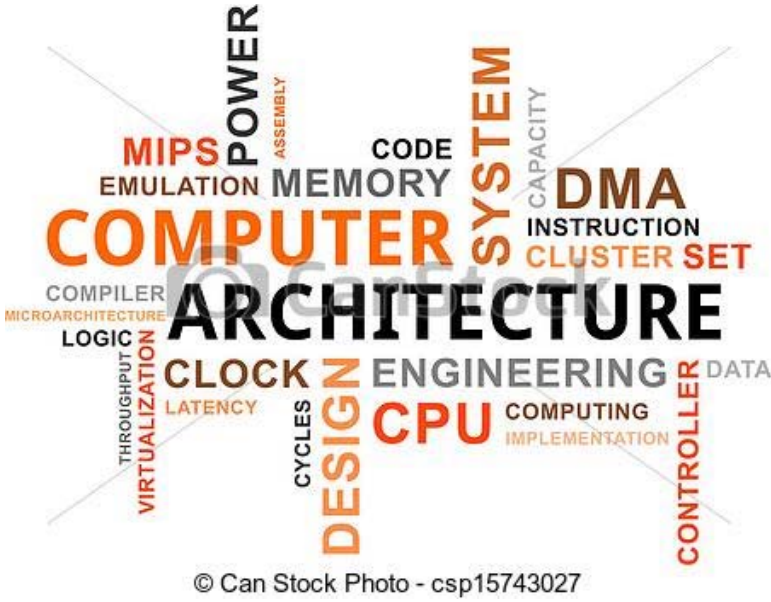
SCHOOL OF INFORMATICS



Αρχιτεκτονική Υπολογιστών

Διάλεξη 8 – Υποσύστημα Μνήμης (Μέρος Α)

Γεώργιος Κεραμίδας, Επίκουρος Καθηγητής

3ο Εξάμηνο, Τμήμα Πληροφορικής

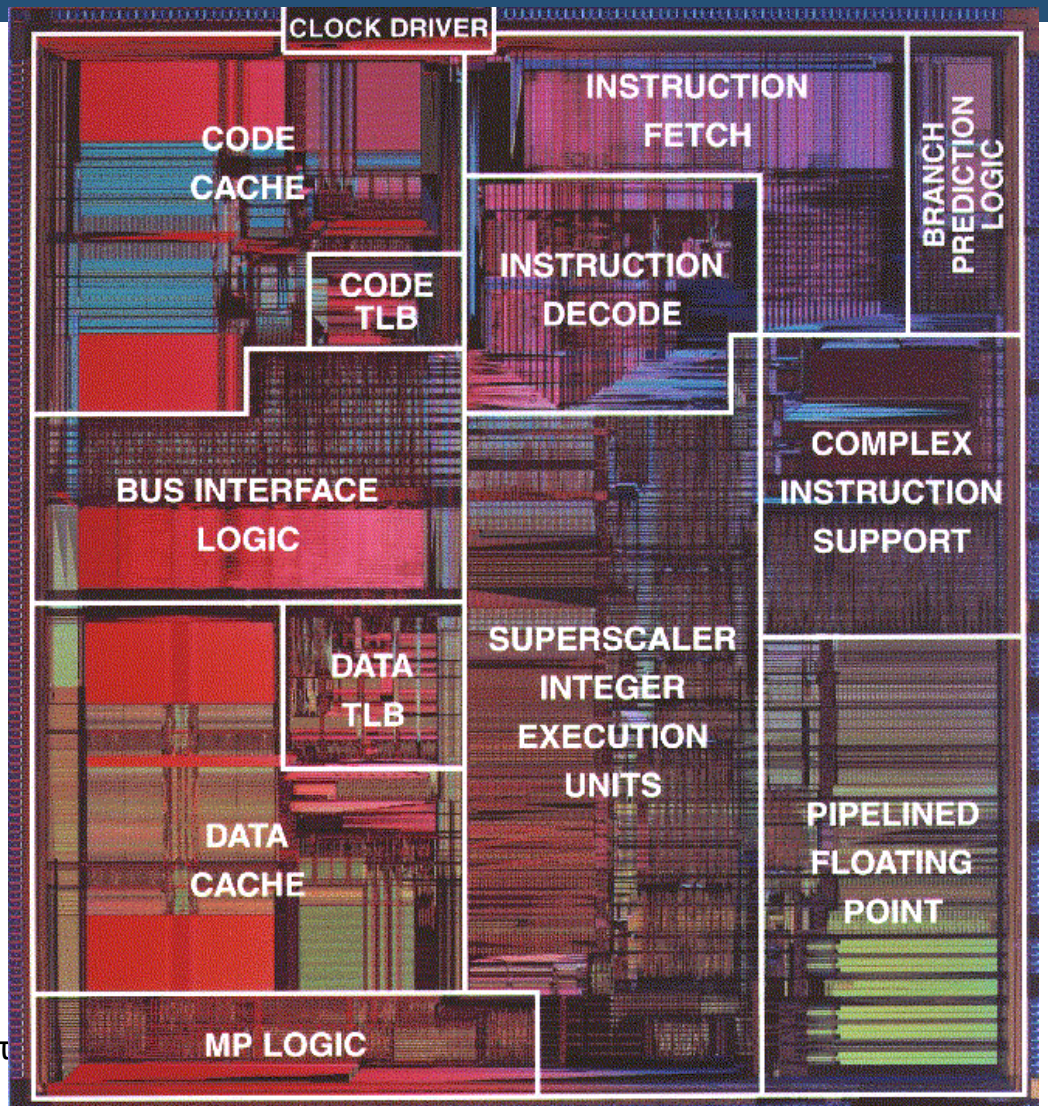
ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

Αντιστοίχιση με ύλη Βιβλίου

- **Το συγκεκριμένο σετ διαφανειών καλύπτει τα εξής κεφάλαια/ενότητες:**
  - **Κεφάλαιο 5: 5.1, 5.2 και 5.3**

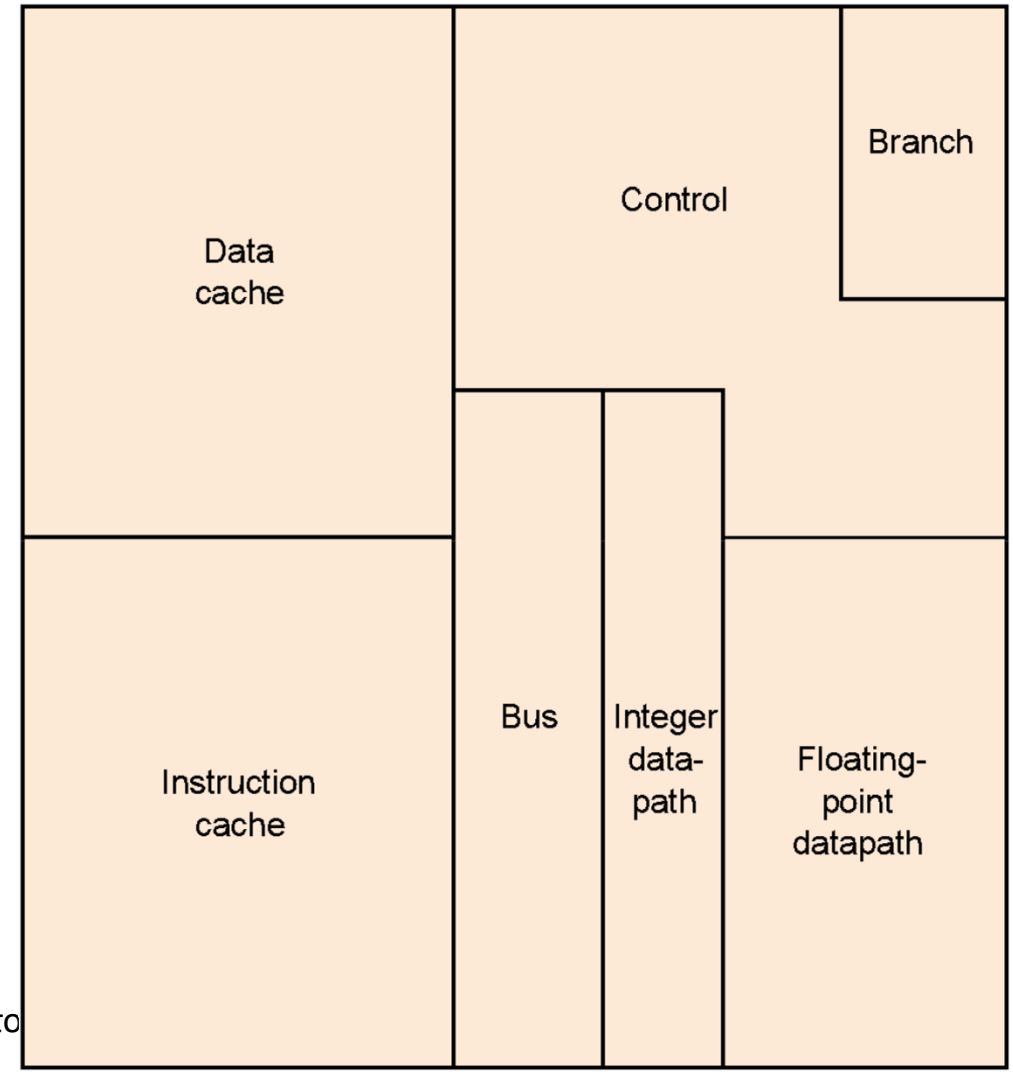
Διάταξη ενός Τυπικού Μικροεπεξεργαστή

- **The Intel Pentium Classic**

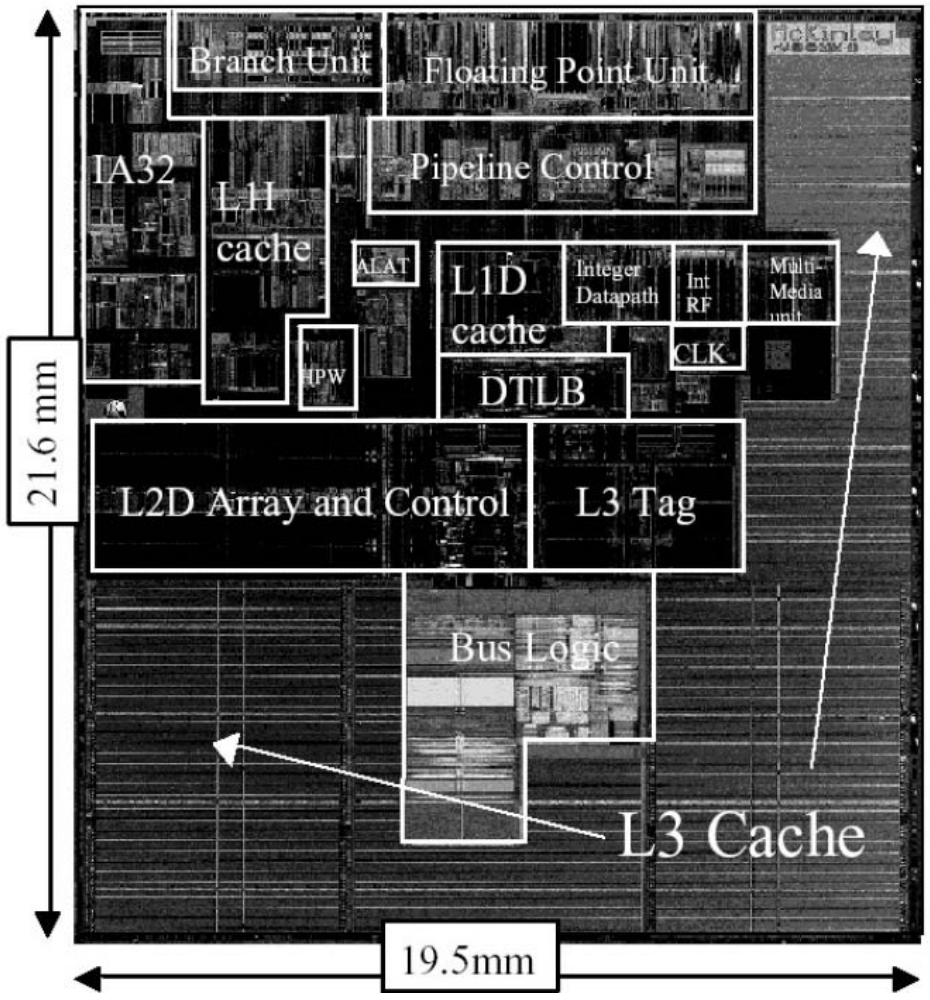


Διάταξη ενός Τυπικού Μικροεπεξεργαστή

- **The Intel Pentium Classic**



Itanium-2 On Chip Caches



Level 1, 16KB, 4-way s.a., 64B line, quad-port (2 load+2 store), single cycle latency

Level 2, 256KB, 4-way s.a, 128B line, quad-port (4 load or 4 store), five cycle latency

Level 3, 3MB, 12-way s.a., 128B line, single 32B port, twelve cycle latency

Power 7 On-Chip Caches [IBM 2009]

32KB L1 I$/core

32KB L1 D$/core

3-cycle latency

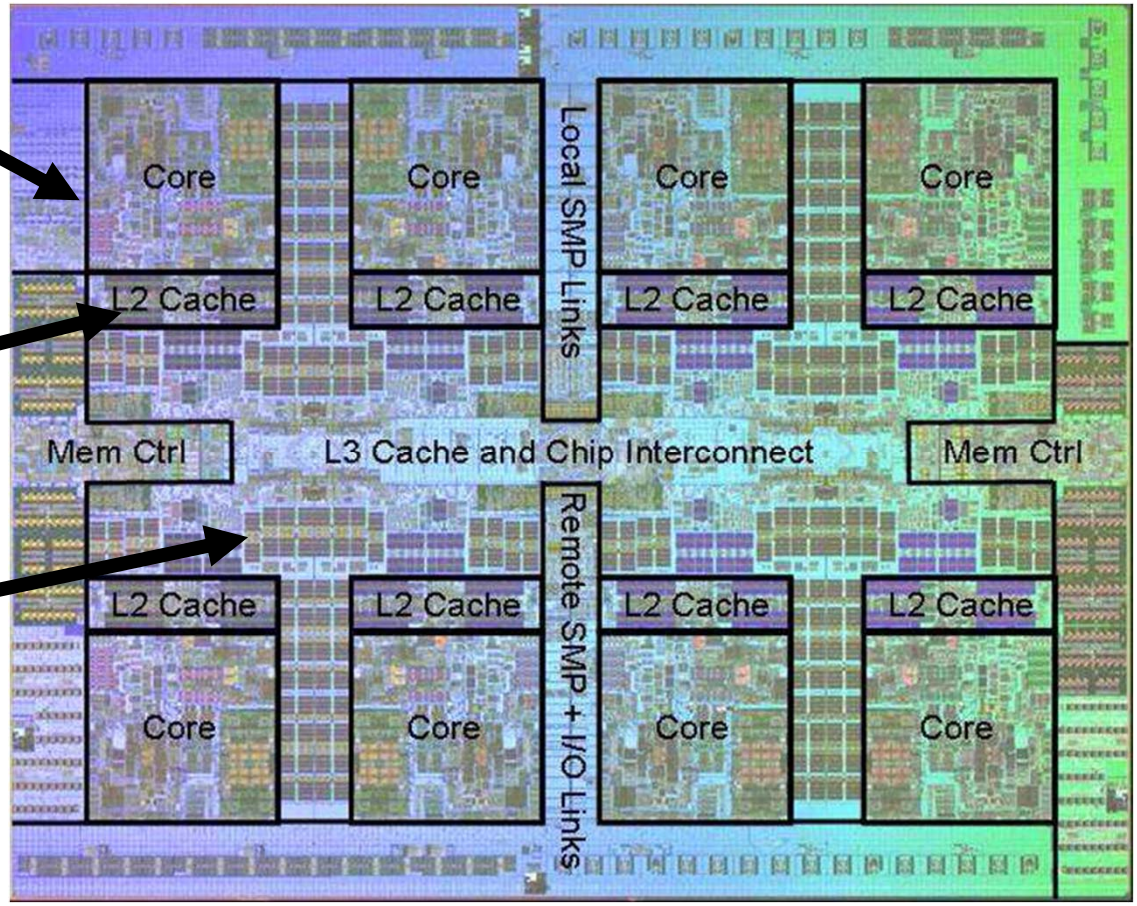
256KB Unified L2$/core

8-cycle latency

32MB Unified Shared L3$

Embedded DRAM

25-cycle latency to local slice

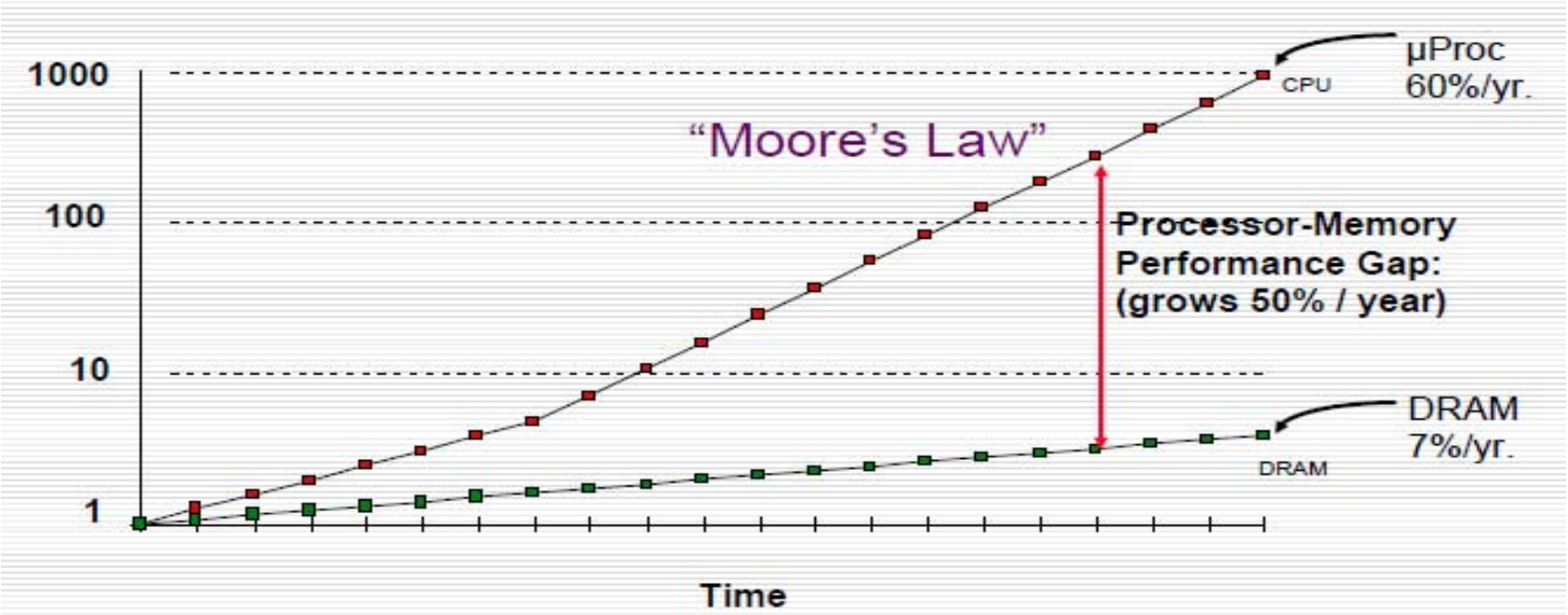


Τεχνολογία μνήμης

- Στατική RAM (Static RAM – SRAM)
  - 0.5ns – 2.5ns, $2000 – $5000 ανά GB
- Δυναμική RAM (Dynamic RAM – DRAM)
  - 50ns – 70ns, $20 – $75 ανά GB
- Μαγνητικός δίσκος
  - 5ms – 20ms, $0.20 – $2 ανά GB
- Ιδανική μνήμη
  - Χρόνος προσπέλασης της SRAM
  - Χωρητικότητα και κόστος/GB του δίσκου

Το Χάσμα Μνήμης (Memory Wall / Gap)

"A Case for Intelligent RAM: IRAM," by David Patterson et. al., IEEE Micro, 1997

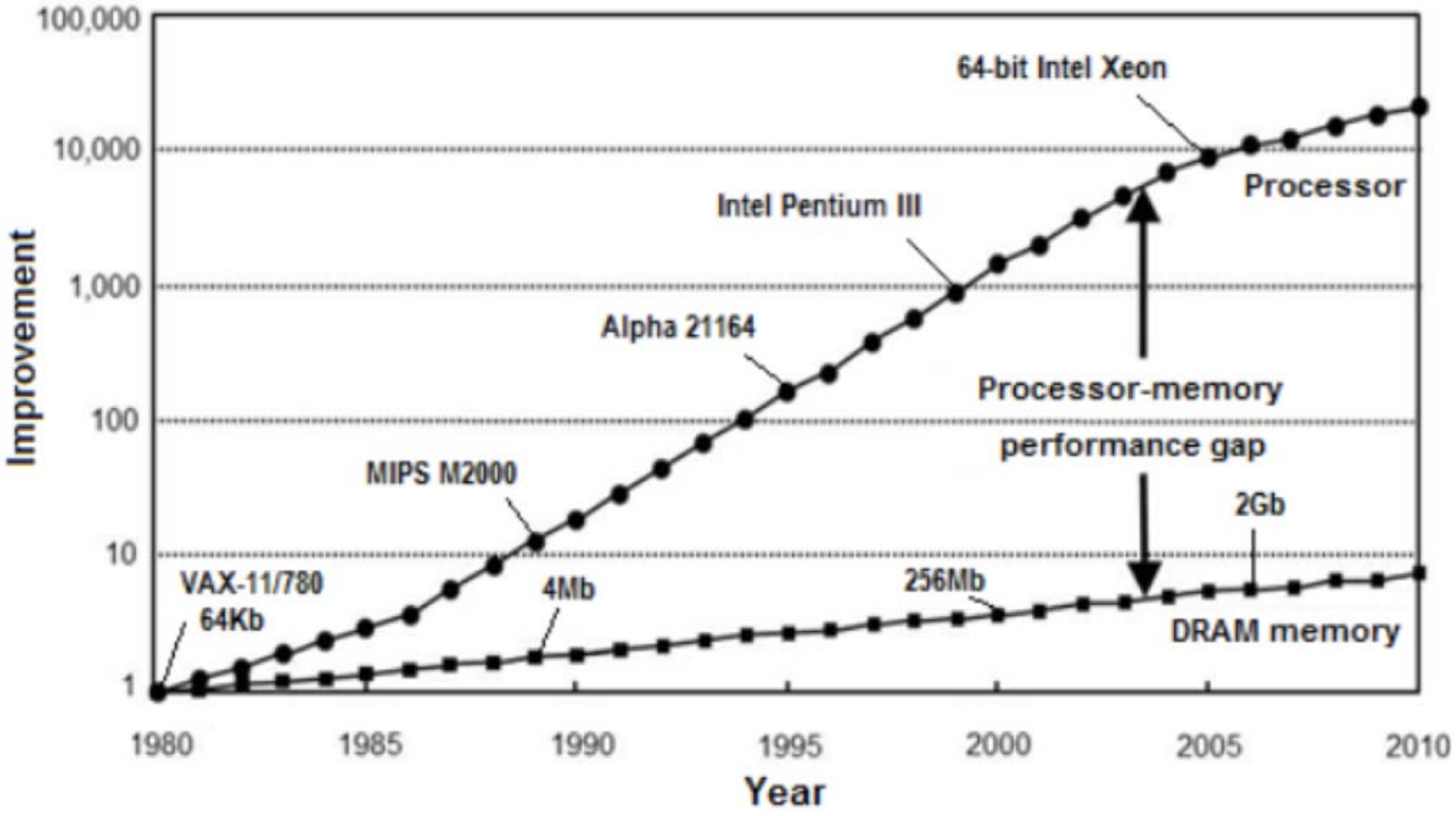


CPU 2x/2 χρόνια

DRAM 2x/10 χρόνια

Το χάσμα αυξάνεται 50% το χρόνο …

Το Χάσμα Μνήμης (Memory Wall / Gap)

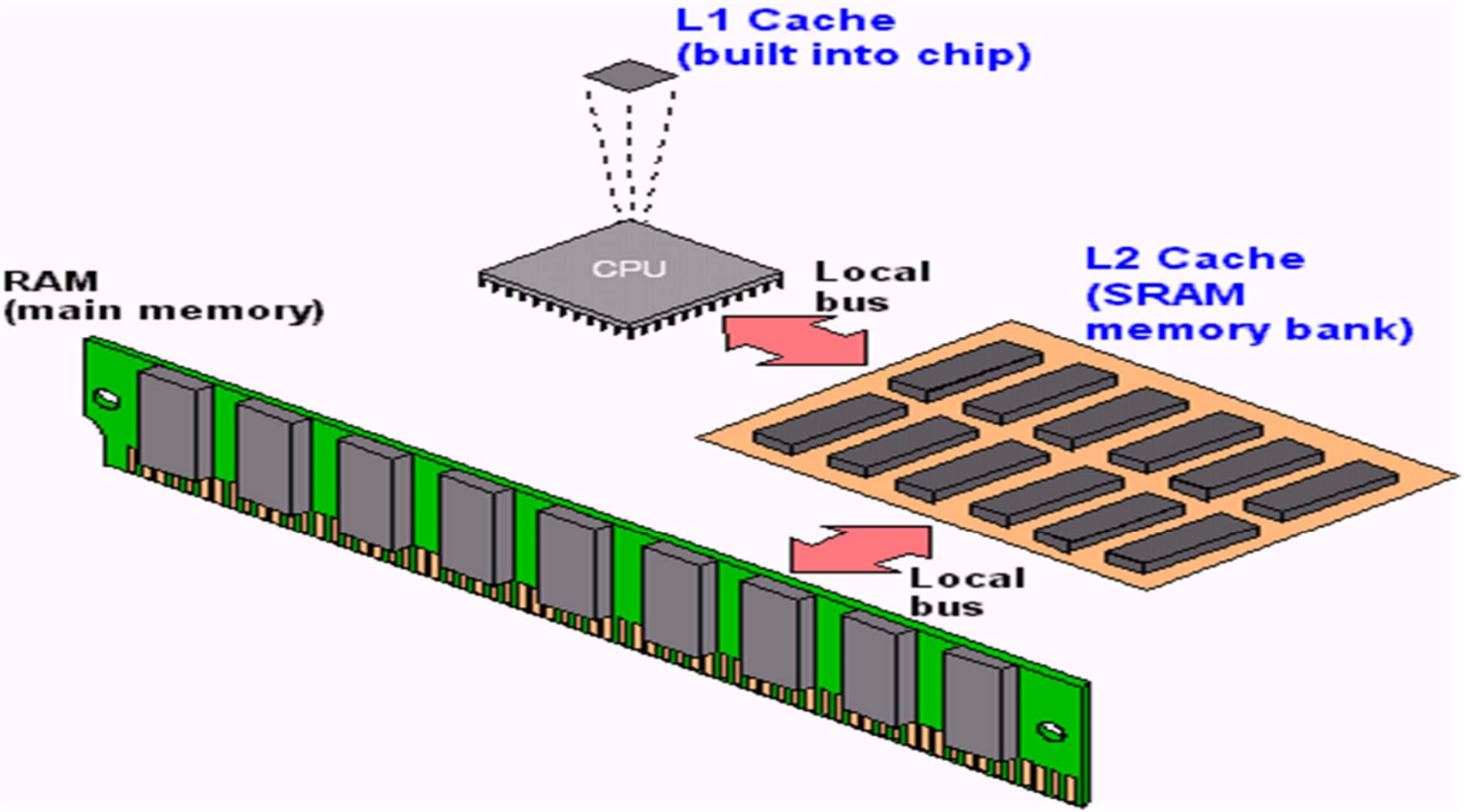


Χαρακτηριστικά Συστημάτων Μνήμης

- Τοποθεσία
- Χωρητικότητα
- Μονάδα μεταφοράς
- Μέθοδος πρόσβασης
- Απόδοση
- Φυσικοί τύποι
- Φυσικά χαρακτηριστικά
- Οργάνωση

Τοποθεσία

- CPU
- Εσωτερική
- Εξωτερική



Μέθοδοι Πρόσβασης (1)

- Διαδοχικός
  - Ξεκινάει από την αρχή και εκτελεί ανάγνωση με τη σειρά
  - Ο χρόνος πρόσβασης εξαρτάται από την τοποθεσία των δεδομένων και τις προηγούμενες θέσεις τους
- Π.χ. Ταινίες
  - Άμεσος
- Ξεχωριστά μπλοκ έχουν μοναδική διεύθυνση
- Η πρόσβαση γίνεται με άλμα στην κατάλληλη περιοχή και στην συνέχεια διαδοχική αναζήτηση
  - Ο χρόνος πρόσβασης εξαρτάται από την τοποθεσία των δεδομένων και τις προηγούμενες θέσεις τους
  - Π.χ. Δίσκοι

Μέθοδοι Πρόσβασης (2)

- **RAM: Random Access Memory ή Μνήμες Τυχαίας Προσπέλασης**
- **Ερώτηση: Τι είναι «τυχαίο» σε αυτές τις μνήμες ????**

- **Τυχαία**
  - **Για μεμονωμένες διευθύνσεις εντοπίζονται οι θέσεις ακριβώς**
  - **Ο χρόνος πρόσβασης είναι ανεξάρτητος από την τοποθεσία των δεδομένων και τις προηγούμενες προσπελάσεις**
  - **π.χ. RAM**

Φυσικά χαρακτηριστικά

- **Φθορά**
- **Πτητικότητα (ROM, RAM)**
- **Ικανότητα διαγραφής (ROM, RAM)**
- **Κατανάλωση ισχύος**

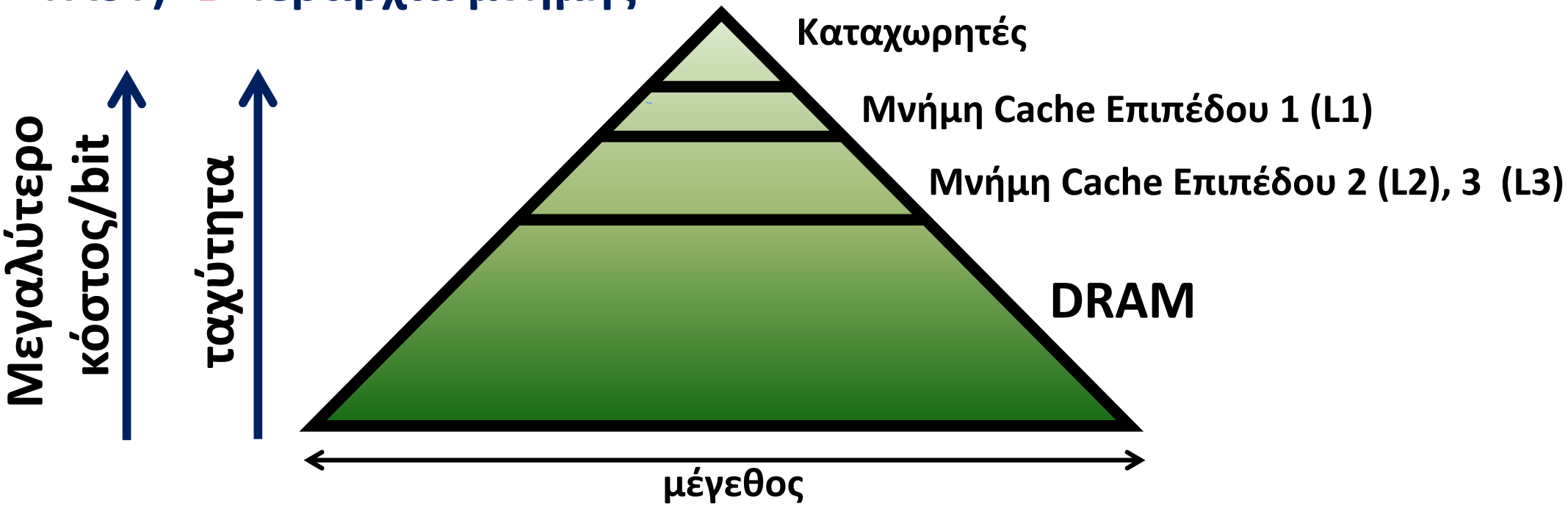
- **Ερώτηση: Γιατί έχουμε μνήμες ROM εφόσον υπάρχουν οι RAM?**

Ας ξεκαθαρίσουμε …

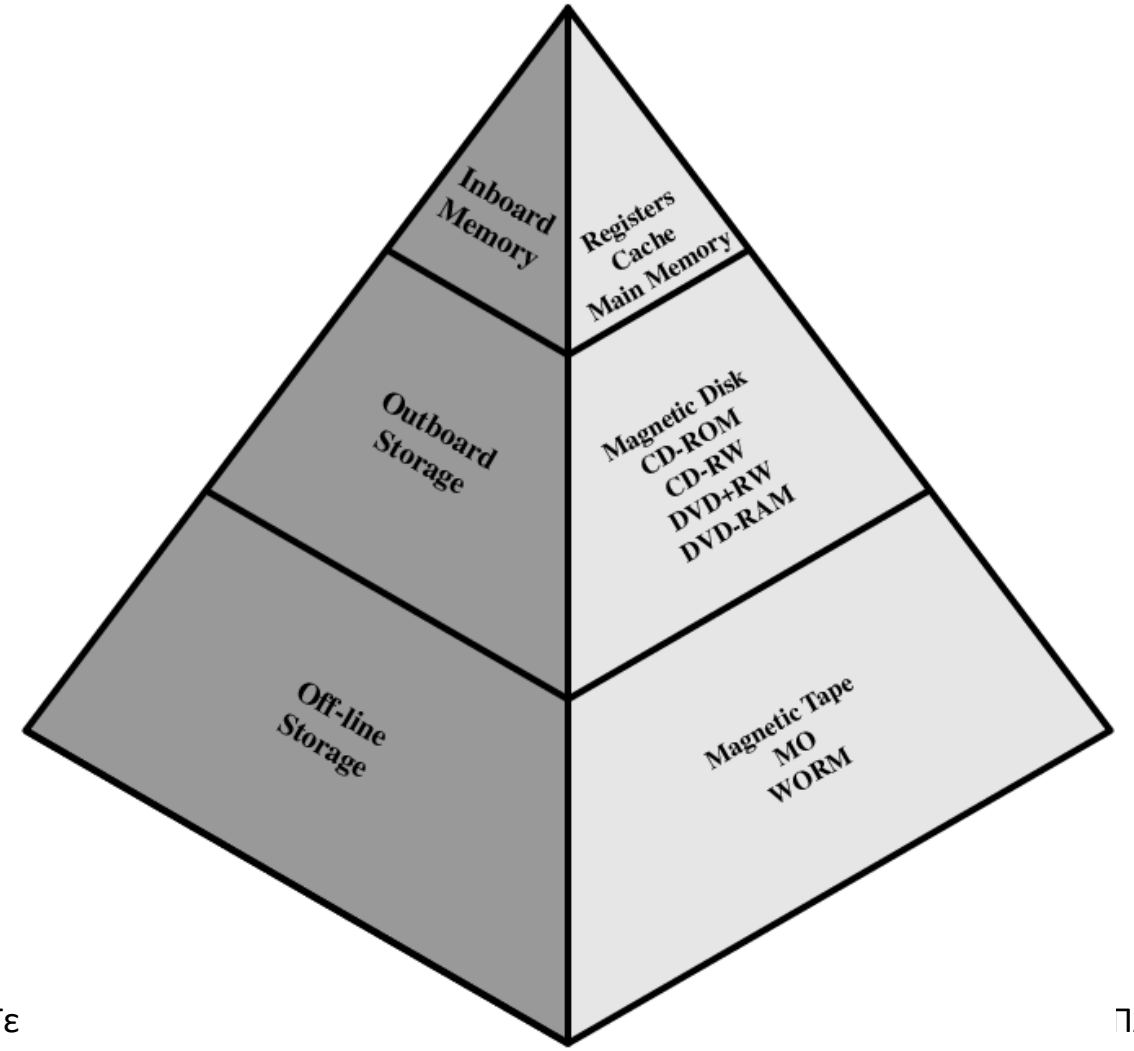
- Μνήμες RAM: **Vast or Fast**, επιπλέον **Fast** σημαίνει μεγαλύτερο κόστος ανά bit
- Μνήμη τυχαίας προσπέλασης (RAM): Οποιοδήποτε (τυχαία) διεύθυνση και να διαλέξουμε για read/write, έχουμε τον ίδιο χρόνο πρόσβασης (access time)
- ROM is still RAM: οι ROM (σε αντίθεση με τις RAM) είναι μη-πτητικές (non-volatile), δηλ. δεν χάνουν τα δεδομένα τους όταν δεν υπάρχει τροφοδοσία
- DRAMs μπορούν να είναι on-chip: οι SRAM είναι γρήγορες (μικρό access time), οι DRAMs έχουν υψηλή πυκνότητα ολοκλήρωσης (high density)

Χαρακτηριστικά (υπο-)Συστήματος Μνήμης

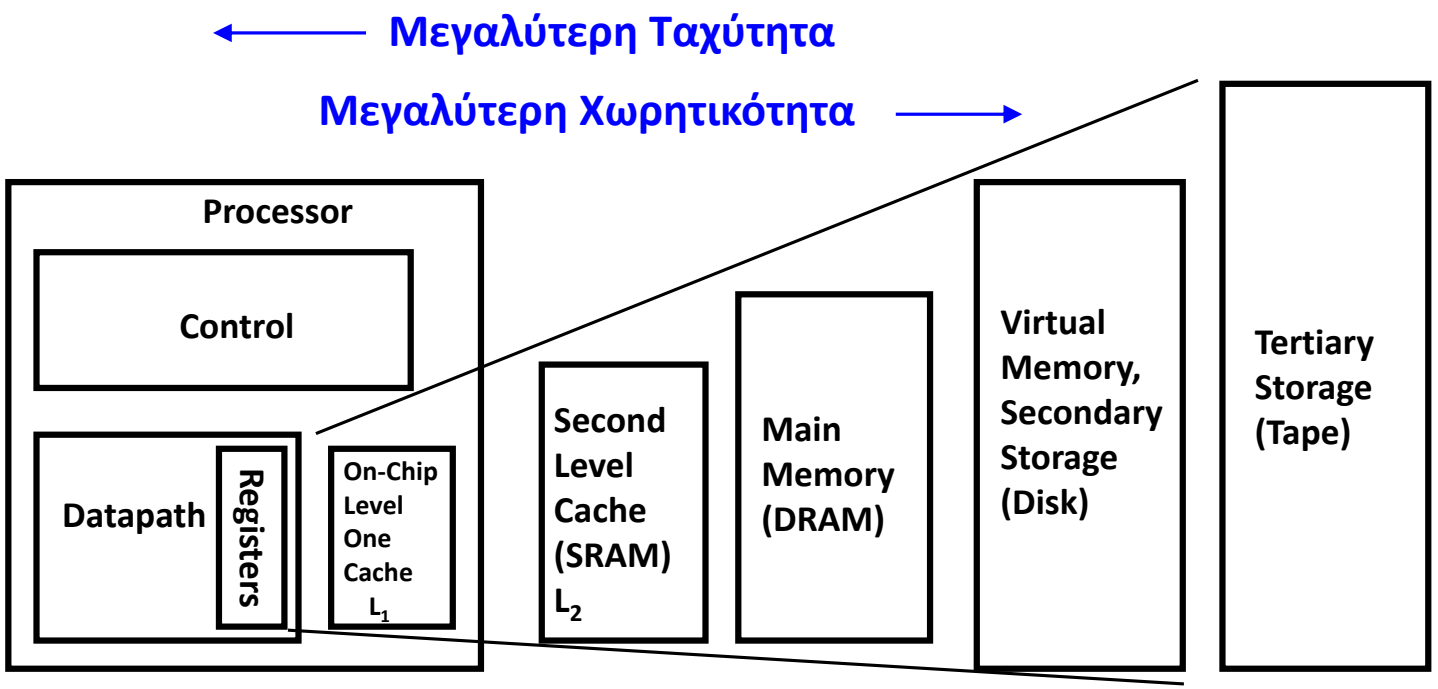
- Ουσιαστικό χαρακτηριστικό της μνήμης: Η μνήμη μπορεί να είναι ΤΕΡΑΣΤΙΑ 'Η ΓΡΗΓΟΡΗ αλλά ΟΧΙ και τα δύο μαζί **(VAST or FAST) → Ιεραρχία μνήμης**



Ιεραρχία Μνήμης - Διάγραμμα



Ιεραρχίας Μνήμης & Χρόνος Πρόσβασης



Ταχύτητα :	1-5ns	10ns	30ns	50ns	10ms	10sec
Μέγεθος :	<1KB	<256KB	<8MB	<4GB	>1GB	TB
Bandwidth :	150GB/s	50GB/s	25GB/s	4GB/s	10MB/s	

Σύγκριση Μεγεθών της Κρυφής Μνήμης

Επεξεργαστής	Είδος	Έτος κατασκευής	L1 κρυφή μνήμη	L2 κρυφή μνήμη	L3 κρυφή μνήμη
IBM 360/85	Mainframe	1968	16 to 32 KB	—	—
PDP-11/70	Minicomputer	1975	1 KB	—	—
VAX 11/780	Minicomputer	1978	16 KB	—	—
IBM 3033	Mainframe	1978	64 KB	—	—
IBM 3090	Mainframe	1985	128 to 256 KB	—	—
Intel 80486	PC	1989	8 KB	—	—
Pentium	PC	1993	8 KB/8 KB	256 to 512 KB	—
PowerPC 601	PC	1993	32 KB	—	—
PowerPC 620	PC	1996	32 KB/32 KB	—	—
PowerPC G4	PC/server	1999	32 KB/32 KB	256 KB to 1 MB	2 MB
IBM S/390 G4	Mainframe	1997	32 KB	256 KB	2 MB
IBM S/390 G6	Mainframe	1999	256 KB	8 MB	—
Pentium 4	PC/server	2000	8 KB/8 KB	256 KB	—
IBM SP	High-end server/ supercomputer	2000	64 KB/32 KB	8 MB	—
CRAY MTAb	Supercomputer	2000	8 KB	2 MB	—
Itanium	PC/server	2001	16 KB/16 KB	96 KB	4 MB
SGI Origin 2001	High-end server	2001	32 KB/32 KB	4 MB	—
Itanium 2	PC/server	2002	32 KB	256 KB	6 MB
IBM POWER5	High-end server	2003	64 KB	1.9 MB	36 MB
CRAY XD-1	Supercomputer	2004	64 KB/64 KB	1MB	—

Μόνο L1 – πρώτη φορά πάνω από 100kB

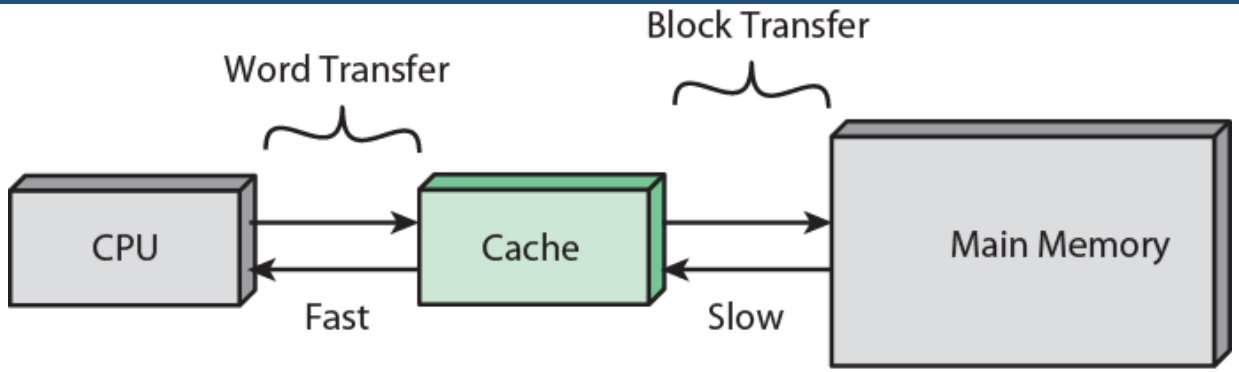
πρώτη φορά L2

πρώτη φορά L3

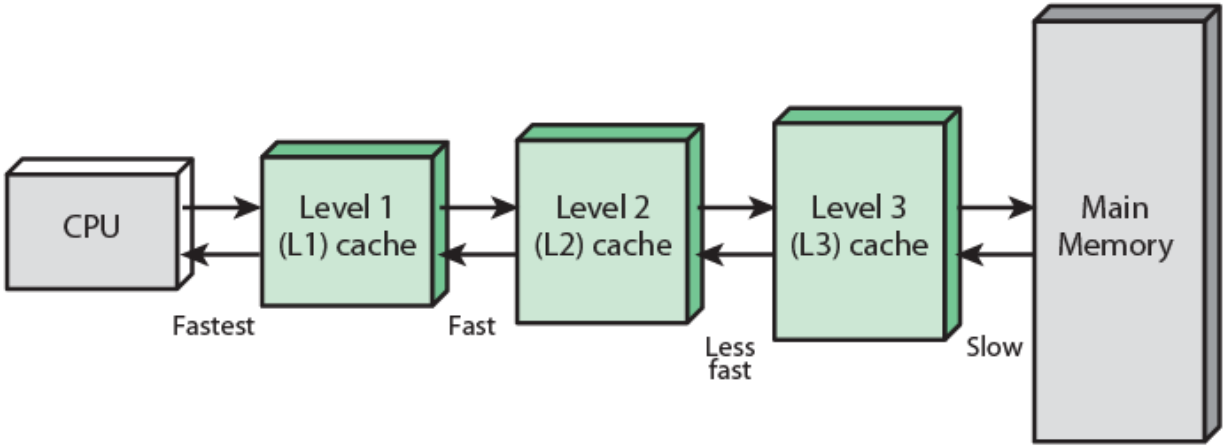
Supercomputer

Supercomputer

Κρυφή Μνήμη & Κύρια Μνήμη



(a) Single cache



(b) Three-level cache organization

Γιατί οι Caches είναι χρήσιμες?

- **Απλοποιημένο παράδειγμα:**
  - **Address space: ~100MB (SPEC2K, ref. inputs), cache: 16KB**
  - **16KB/100MB = 0.015% cached data → ~10x αύξηση στην απόδοση**
- **Κάνουν την κύρια μνήμη να "φαίνεται" πιο γρήγορη**

Γιατί είναι ωφέλιμη η Ιεραρχία Μνήμης;

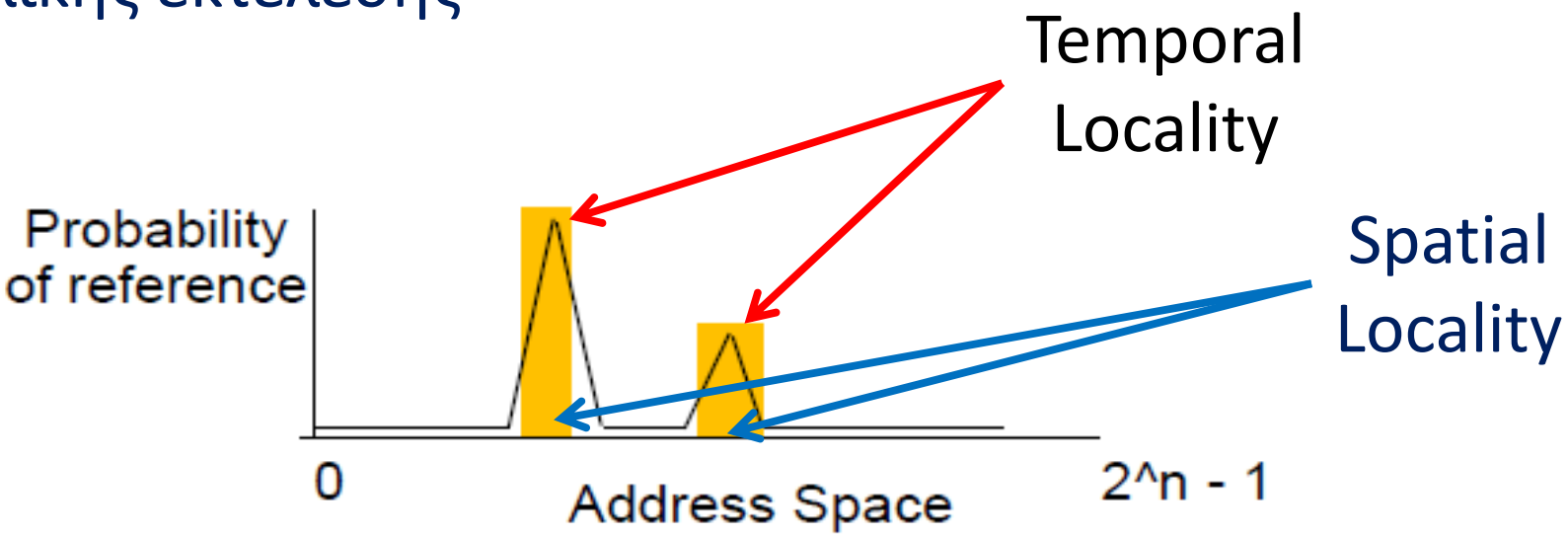
- **Κατά κανόνα τα προγράμματα προσπελαύνουν ένα μικρό μόνο μέρος του συνόλου των διευθύνσεων (εντολές/δεδομένα) κατά την εκτέλεση ενός συγκεκριμένου τμήματός τους ανά πάσα χρονική στιγμή**
- **Δύο είδη τοπικότητας δεδομένων:**
  - **Χρονική τοπικότητα (Temporal Locality): Στοιχεία που έχουν πρόσφατα προσπελαστεί τείνουν να προσπελαύνονται ξανά στο άμεσο μέλλον π.χ., εντολές σε ένα βρόχο, μεταβλητές επαγωγής (induction variables)**
  - **Χωρική τοπικότητα (Spatial locality): Γειτονικά στοιχεία όσων έχουν ήδη προσπελαστεί, έχουν αυξημένη πιθανότητα να προσπελαστούν στο άμεσο μέλλον π.χ., προσπέλαση εντολών στη σειρά, δεδομένα πινάκων**

Γιατί είναι ωφέλιμη η Ιεραρχία Μνήμης;

- **Η ύπαρξη τοπικότητας στις αναφορές ενός προγράμματος, καθιστά εφικτή τη δυνατότητα να ικανοποιείται η αίτηση για δεδομένα από επίπεδα μνήμης που βρίσκονται ιεραρχικά ανώτερα**
- **Ιεραρχία μνήμης**
  - **Αποθήκευσε τα πάντα στο δίσκο**
  - **Αντίγραψε τα πρόσφατα προσπελασθέντα (και τα κοντινά τους) αντικείμενα από το δίσκο σε μια μικρότερη μνήμη DRAM**
    - **Κύρια μνήμη**
  - **Αντέγραψε τα πιο πρόσφατα προσπελασθένταί (και τα κοντινά τους) αντικείμενα από τη DRAM σε μια μικρότερη μνήμη SRAM**
    - **Κρυφή μνήμη (cache) προσαρτημένη στη CPU**

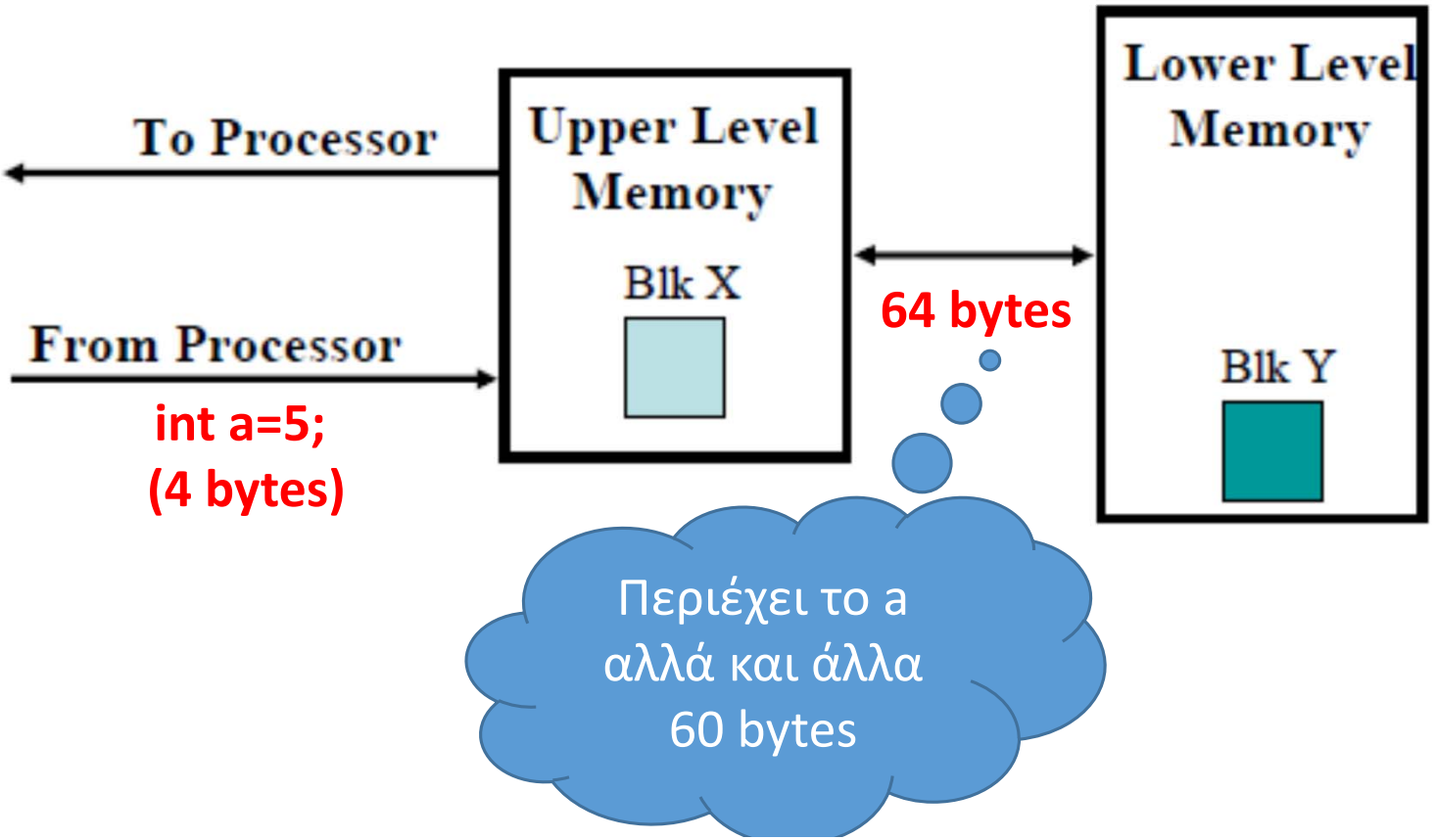
Locality - Οπτικά

- Η θεωρία του Locality: Οι εφαρμογές τείνουν να κάνουν access ένα μικρό μέρος του address space της εφαρμογής μια δεδομένη χρονική στιγμή και για ένα μεγάλο μέρος της συνολικής εκτέλεσης

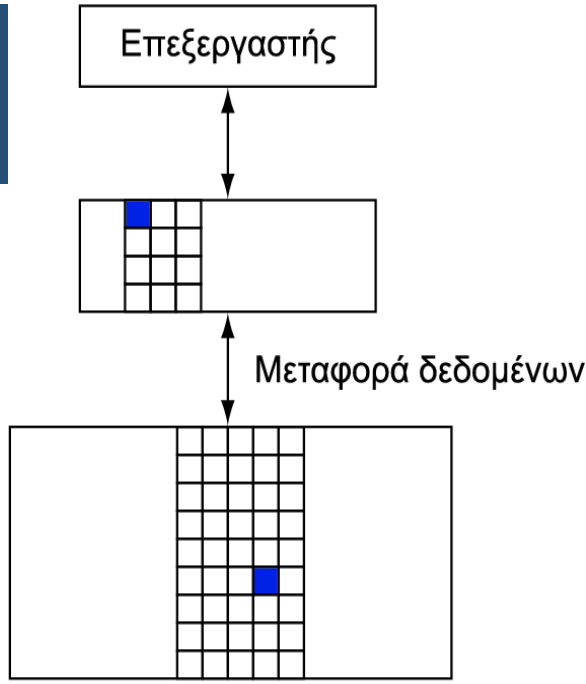


Cache Block

- Cache block – cache line
  - Η μικρότερη μονάδα μεταφοράς δεδομένων μεταξύ των επιπέδων μνήμης



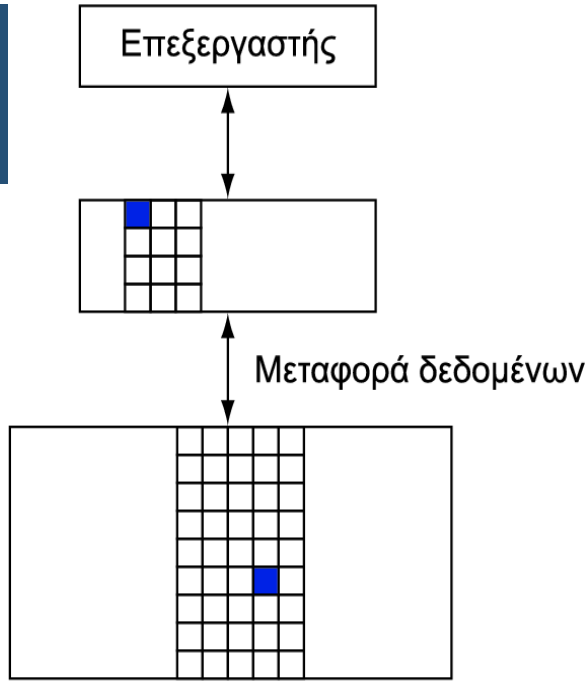
Βασικές Έννοιες



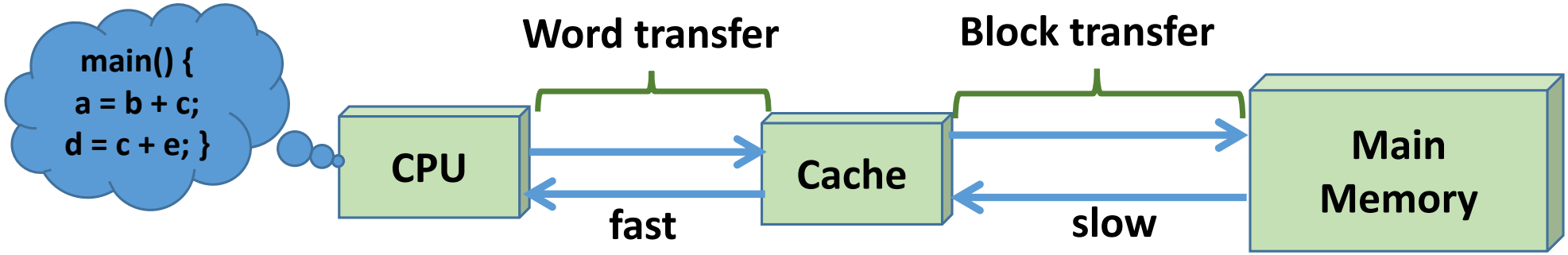
- Σε κάθε προσπέλαση κοιτάμε αν τα περιεχόμενα της μνήμης στη ζητούμενη διεύθυνση βρίσκονται στην cache
- Ναι: επιτυχία ή ευστοχία ή cache HIT
  - Hit rate: ποσοστό προσπελάσεων που πετυχαίνουν
- Όχι: αποτυχία ή αστοχία ή cache MISS
  - Θέση στη cache κενή ? Φέρε τα δεδομένα από τη μνήμη ή το χαμηλότερο επιπεδο και στείλτα στη CPU
  - Θέση στη cache πιασμένη από άλλα δεδομένα? cache REPLACEMENT:
  - Miss rate: ποσοστό που αποτυχαίνει (100-hit rate)

Βασικές Έννοιες (2)

- **hit:** το block βρίσκεται σε κάποια θέση του εξεταζόμενου επιπέδου μνήμης
  - hit rate: hits/συνολικές προσπελάσεις μνήμης
  - hit time: χρόνος προσπέλασης των δεδομένων
- **miss:** το block δεν υπάρχει στο εξεταζόμενο επίπεδο μνήμης
  - miss rate : 1 – (hit rate)
  - miss penalty: (χρόνος μεταφοράς των δεδομένων ενός block στο συγκεκριμένο επίπεδο μνήμης) + (χρόνος απόκτησης των δεδομένων από την CPU)
    - access time: χρόνος απόκτησης της 1ης λέξης
    - transfer time: χρόνος απόκτησης των υπόλοιπων λέξεων



Μέσος Χρόνος Πρόσβασης στη Μνήμη



Βασική εξίσωση (χωρίς cache) – Χρόνος πρόσβασης συστήματος μνήμης:

Average Memory Access Time = Memory_Access_Time + Transfer_Time

Με cache:

Average Memory Access Time = Hit Time + Miss Rate * Miss Penalty

Hit ratio * Access_Time

Άσκηση (θεωρητική)

- **Μέσος χρόνος προσπέλασης της μνήμης ή μέσος χρόνος προσπέλασης του ιεραρχικού συστήματος μνήμης**

$$t_a = p \times t_h + (1 - p)t_m.$$

Άσκηση (θεωρητική)

- Ένα σύστημα μνήμης έχει ένα επίπεδο cache συνδεδεμένο με την κύρια μνήμη. Ο χρόνος πρόσβασης σε μία διεύθυνση που υπάρχει στην cache είναι th ενώ ο αντίστοιχος χρόνος όταν αυτή δεν είναι στην cache είναι tm
- Να βρεθεί μία σχέση για το E, όπου E η απόδοση της μνήμης ορισμένη ως th/tα όπου ta είναι ο μέσος χρόνος πρόσβασης για όλα τα hits & misses σε ένα δεδομένο χρονικό διάστημα.
- **Λύση:**
- Αν p η πιθανότητα των hits, τότε (1-p) η πιθανότητα για cache miss. Επομένως:

$$t_a = p \times t_h + (1-p)t_m.$$

$$E = \frac{t_h}{t_a} = \frac{t_h}{p \times t_h + (1-p)t_m} = \frac{1}{p + (1-p)\frac{t_m}{t_h}}$$

1η Άσκηση – Εκφώνηση

- **Α) Να υπολογίσετε το μέσο χρόνο προσπέλασης του ιεραρχικού συστήματος μνήμης με τα χαρακτηριστικά του πίνακα**

Μνήμη	Χρόνος Προσπέλασης (ns)	Λόγος Επιτυχίας
L1 cache	2	0.30
L2 cache	8	0.75
L3 cache	40	0.90
Main Memory	100	1

- **Β) Έστω ότι για λόγους ελάττωσης του κόστους πρέπει να αφαιρεθεί ένα από τα επίπεδα μνήμης. Ποιο πρέπει να αφαιρεθεί ώστε να επηρεαστεί λιγότερο ο μέσος χρόνος προσπέλασης του ιεραρχικού συστήματος μνήμης.**

1η Άσκηση – Λύση (Α)

- **Α) Να υπολογίσετε το μέσο χρόνο προσπέλασης του ιεραρχικού συστήματος μνήμης με τα χαρακτηριστικά του πίνακα**
- **Θεωρούμε ότι η κύρια μνήμης (ΜΜ) έχει πάντα τα δεδομένα (τα έχει φορτώσει το λειτουργικό σύστημα).**

Ο μέσος χρόνος προσπέλασης της μνήμης (Τ) είναι:

$$T = \Sigma_{i=1,4} \ (E_i - E_{i-1}) \times T_i \text{ , όπου}$$

- **E_i ο λόγος επιτυχίας του επιπέδου i**
- **T_i ο χρόνος πρόσβασης του επιπέδου i**

Μνήμη	Χρόνος Προσπ. (ns)	Λόγος Επιτυχίας
L1	2	0.30
L2	8	0.75
L3	40	0.90
MM	100	1

T = 0.3 x 2 + (0.75 – 0.30) x 8 + (0.90-0.75) x 40 + (1-0.90) x 100 = 20.2 nsec

Συμπέρασμα:

- **Ο μέσος χρόνος πρόσβασης στην μνήμη "κοστίζει" 20.2nsec**
- **Αν δεν είχαμε ιεραρχία μνήμης, θα κόστιζε 100nsec**

1η Άσκηση – Λύση (Β)

- **Β) Έστω ότι για λόγους ελάττωσης του κόστους πρέπει να αφαιρεθεί ένα από τα επίπεδα μνήμης. Ποιο πρέπει να αφαιρεθεί ώστε να επηρεαστεί λιγότερο ο μέσος χρόνος προσπέλασης του ιεραρχικού συστήματος μνήμης.**
- **Θα εξετάσουμε κάθε περίπτωση ξεχωριστά.**

Μνήμη	Χρόνος Προσπ. (ns)	Λόγος Επιτυχίας
L1	2	0.30
L2	8	0.75
L3	40	0.90
MM	100	1

Χωρίς L1:

T1 = 0.75 x 8 + (0.90-0.75) x 40 + (1-0.90) x 100 = 22 nsec

Χωρίς L2:

T2 = 0.3 x 2 + (0.90-0.30) x 40 + (1-0.90) x 100 = 34.6 nsec

Χωρίς L3:

T3 = 0.3 x 2 + (0.75 – 0.30) x 8 + (1-0.75) x 100 = 29.2 nsec

Συμπέρασμα:

- **T1<T3<T2 → καλύτερα να αφαιρεθεί η L1 cache**

Σε ένα άλλο πρόγραμμα μπορείς να είχαμε διαφορετικά αποτελέσματα

2η Άσκηση – Εκφώνηση

- **Ένας σχεδιαστής πρέπει να επιλέξει μεταξύ των ακόλουθων επιλογών:**
  - **1) Υλοποίηση κρυφής μνήμης ενός επιπέδου με μέγεθος 16KB, λόγο επιτυχίας h_1=0.85 και χρόνο προσπέλασης t_1=9ns**
  - **2) Υλοποίηση κρυφής μνήμης ενός επιπέδου με μέγεθος 32KB, λόγο επιτυχίας h_2=0.95 και χρόνο προσπέλασης t_2=12ns**
- **Ποια επιλογή πετυχαίνει τον ελάχιστο χρόνο προσπέλασης του συστήματος μνήμης? Θεωρήστε ότι η CPU προσπελαύνει πληροφορία μόνο από την κρυφή μνήμη**

2η Άσκηση – Λύση

- Έστω t_{block}, ο χρόνος μεταφοράς ενός μπλοκ πληροφορίας από την κύρια μνήμη στην cache.

Ο μέσοι χρόνοι προσπέλασης (T_1 και T_2) είναι:

$T_1 = h_1 \times t_1 + (1 - h_1) \times (t_1 + t_{block})$

$T_2 = h_2 \times t_2 + (1 - h_2) \times (t_2 + t_{block})$

Η πρώτη λύση θα είναι καλύτερη αν:

$T2 > T1 \rightarrow \ldots\ldots\ldots \rightarrow t_{block} < 30$ ns

Επιλογή 1:
size=16KB,
h_1=0.85, t_1=9ns

Επιλογή 2:
size=32KB,
h_2=0.95, t_2=12ns

Συμπέρασμα 1: Η πρώτη λύση είναι καλύτερη όταν η τιμή του t_{block} είναι μικρότερη των 30ns

2η Άσκηση – Λύση

- Έστω t_{block}, ο χρόνος μεταφοράς ενός μπλοκ πληροφορίας από την κύρια μνήμη στην cache.

Ο μέσοι χρόνοι προσπέλασης (T_1 και T_2) είναι:

$T_1 = h_1 \times t_1 + (1 - h_1) \times (t_1 + t_{block})$

$T_2 = h_2 \times t_2 + (1 - h_2) \times (t_2 + t_{block})$

Η πρώτη λύση θα είναι καλύτερη αν:

$T2 > T1 \rightarrow \ldots\ldots\ldots \rightarrow t_{block} < 30$ ns

Επιλογή 1:
size=16KB,
$h_1=0.85$, $t_1=9$ns

Επιλογή 2:
size=32KB,
$h_2=0.95$, $t_2=12$ns

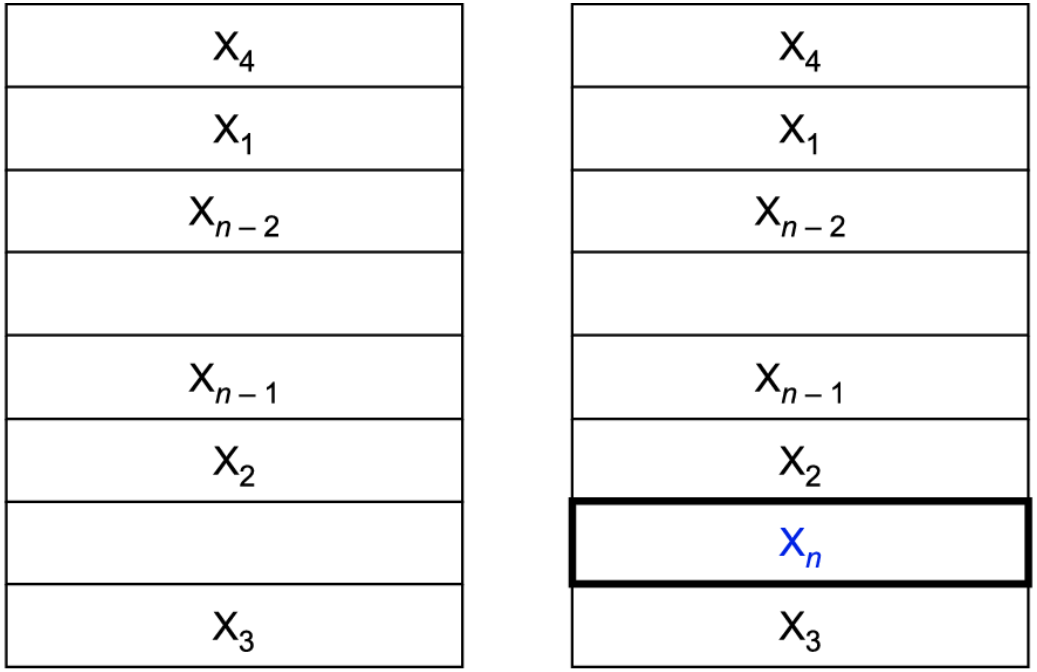
Συμπέρασμα 2: Μεγαλύτερη cache δεν είναι πάντα η καλύτερη επιλογή. Αυτό ισχύει κυρίως για τις caches πρώτου επιπέδου

Επανάληψη… Τι είναι κρυφή μνήμη?

- **Ενδιάμεση μνήμη ανάμεσα στον ταχύτατο επεξεργαστή και την πιο αργή κύρια μνήμη – λειτουργεί στην ταχύτητα του επεξεργαστή (το πρώτο επίπεδο cache)**
- **Σκοπός της να μειώνει το χρόνο αδράνειας του επεξεργαστή, όταν περιμένει τη μεταφορά δεδομένων από την κύρια μνήμη**
- **Αυξάνει την απόδοση των επεξεργαστών δηλ. μειώνει τον χρόνο εκτέλεσης των εφαρμογών που εκτελούνται**
- **Βελτιώνει τη σχέση κόστους και απόδοσης**

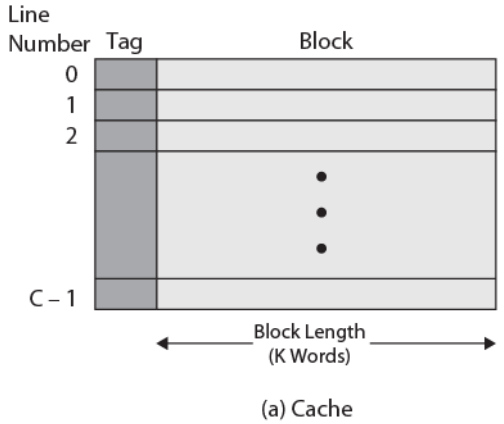
Πρόσβαση κρυφής μνήμης

- **Πώς γνωρίζουμε αν τα δεδομένα είναι παρόντα;**
- **Πού κοιτάζουμε;**
- **Δεδομένες προσπελάσεις X_1, …, X_{n-1}, X_n**



α. Πριν από την αναφορά στο X_n β. Μετά από την αναφορά στο X_n

Πρόσβαση κρυφής μνήμης



- **Παράδειγμα: 2^{14} γραμμές → 16384 γραμμές**
- **Θα πρέπει να βρούμε τρόπο να έχουμε μικρούς χρόνους πρόσβασης στις κρυφές μνήμες ? Π.χ. 2-3 κύκλους σε κρυφές μνήμες πρώτου επιπέδου**
- **Δεν μπορούμε να ψάχνουμε μία-μία τις θέσεις στην κρυφή μνήμη πχ. Θα χρειαζόμασταν ~16384 κύκλους (χοντρικά για το διπλανό παράδειγμα) → Η κρυφή μνήμη θα ήταν άχρηστη**
- **Χρειαζόμαστε κάτι πιο έξυπνο!!!**

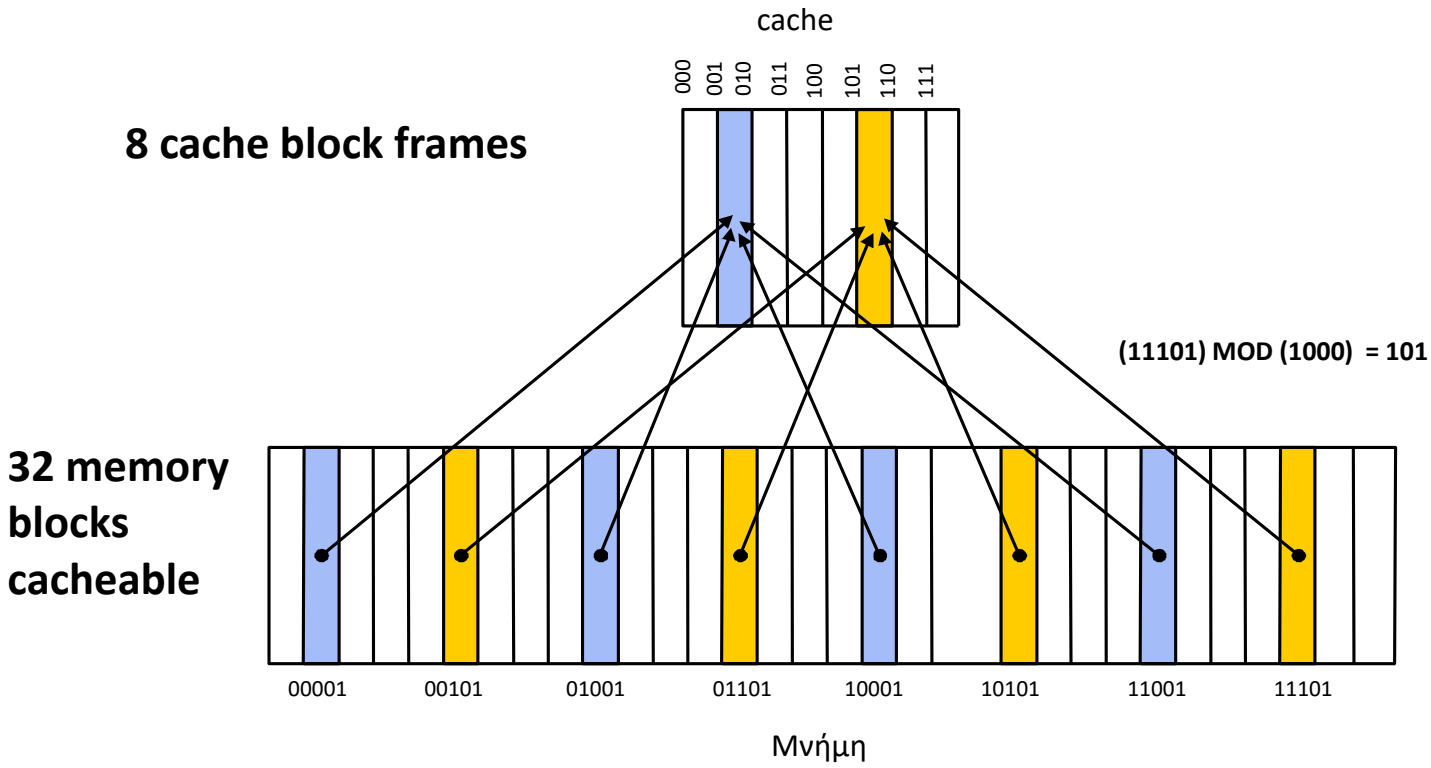
- Έστω Κρυφή Μνήμη μεγέθους 64kBytes
  - 16384 (2^{14}) γραμμές
  - Block μεγέθους 4 bytes
  - 14-bit διευθύνσεις γραμμών

Κρυφή μνήμη άμεσης απεικόνισης

- Η θέση καθορίζεται από τη διεύθυνση
- Άμεση απεικόνιση ή χαρτογράφηση (direct mapping): μόνο μία επιλογή
  - Κάθε block μπορεί να αποθηκευθεί μόνο σε μία θέση στην cache
- Πλήθος μπλοκς είναι δύναμη του 2
- Χρήση των χαμηλής ταξής bit της διεύθυνσης

(Διεύθυνση μπλοκ) modulo (#Μπλοκ κρυφής μνήμης)

Στο παράδειγμά μας: (διεύθυνση block) MOD (8)

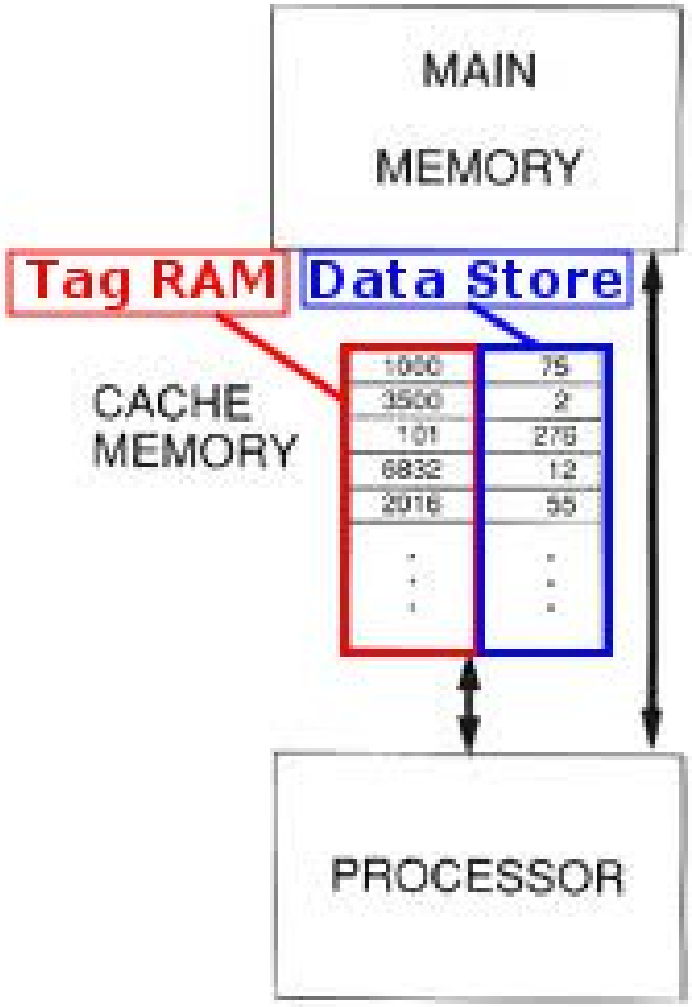


Ετικέτες και έγκυρα bit

- **Πώς γνωρίζουμε ποιο συγκεκριμένο μπλοκ αποθηκεύεται σε μια θέση της κρυφής μνήμης;**
  - **Αποθήκευση της δ/νσης του μπλοκ μαζί με τα δεδομένα**
  - **Στη πραγματικότητα, χρειάζονται μόνο τα bits υψηλής τάξης της διεύθυνσης**
  - **Ονομάζονται ετικέτα (tag)**
- **Και αν δεν υπάρχουν δεδομένα σε μια θέση;**
  - **Έγκυρο (valid) bit: 1 = παρόντα, 0 = όχι παρόντα**
  - **Αρχικά 0**

Εσωτερική οργάνωση κρυφής μνήμης

- **Η κρυφή μνήμη αποτελείται από δύο βασικά μέρη (ξεχωριστά κελιά αποθήκευσης για καθένα από αυτά…):**
  - **Data -> το μέρος που αποθηκεύονται οι πληροφορίες**
  - **Tag -> αποθηκεύονται κομμάτια διευθύνσεων της κύριας μνήμης (ανάλογα με την τεχνική χαρτογράφησης…)**
- **Τα valid bits δεν φαίνονται στο σχήμα**



Παράδειγμα κρυφής μνήμης

- **8 μπλοκ, 1 λέξη/μπλοκ, άμεσης απεικόνισης**
- **Αρχική κατάσταση**

Αριθμοδείκτης	V	Ετικέτα	Δεδομένα
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Παράδειγμα κρυφής μνήμης

Δ/νση λέξης	Δυαδική δ/νση	Ευστοχία/αστοχία	Μπλοκ κρυφής μνήμης
22	10 110	Αστοχία	110

Αριθμοδείκτης	V	Ετικέτα	Δεδομένα
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	**Y**	**10**	**Mem[10110]**
111	N		

Παράδειγμα κρυφής μνήμης

Δ/νση λέξης	Δυαδική δ/νση	Ευστοχία/αστοχία	Μπλοκ κρυφής μνήμης
26	11 010	Αστοχία	010

Αριθμοδείκτης	V	Ετικέτα	Δεδομένα
000	N		
001	N		
010	**Y**	**11**	**Mem[11010]**
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Παράδειγμα κρυφής μνήμης

Δ/νση λέξης	Δυαδική δ/νση	Ευστοχία/αστοχία	Μπλοκ κρυφής μνήμης
22	10 110	Ευστοχία	110
26	11 010	Ευστοχία	010

Αριθμοδείκτης	V	Ετικέτα	Δεδομένα
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Παράδειγμα κρυφής μνήμης

Δ/νση λέξης	Δυαδική δ/νση	Ευστοχία/αστοχία	Μπλοκ κρυφής μνήμης
16	10 000	Αστοχία	000
3	00 011	Αστοχία	011
16	10 000	Ευστοχία	000

Αριθμοδείκτης	V	Ετικέτα	Δεδομένα
000	Y	10	Mem[10000]
001	N		
010	Y	11	Mem[11010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Παράδειγμα κρυφής μνήμης

Δ/νση λέξης	Δυαδική δ/νση	Ευστοχία/αστοχία	Μπλοκ κρυφής μνήμης
18	10 010	Αστοχία	010

- **Αντικατάσταση του παλαιού block 010 (26_{10}) με το νέο 18_{10}**

Αριθμοδείκτης	V	Ετικέτα	Δεδομένα
000	Y	10	Mem[10000]
001	N		
010	**Y**	**10**	**Mem[10010]**
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

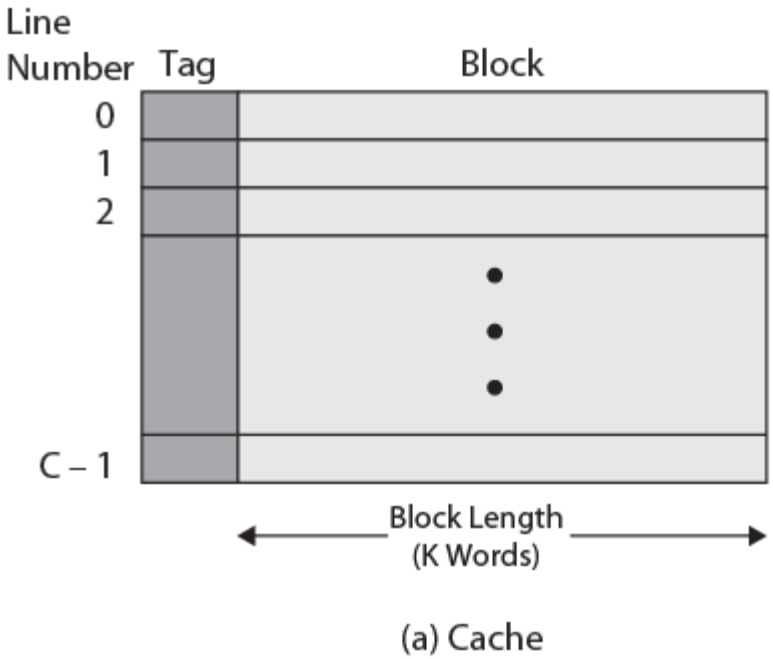
Άμεση χαρτογράφηση (Direct mapping)

- Κάθε μπλοκ της κύριας μνήμης χαρτογραφεί (αντιστοιχίζεται σε) μόνο μία γραμμή κρυφής μνήμης
  - Με άλλα λόγια εάν ένα μπλοκ βρίσκεται μέσα στην κρυφή μνήμη, θα πρέπει να είναι σε ένα συγκεκριμένο μέρος
- Η διεύθυνση αποτελείται από δύο μέρη:
  - 1) Τα λιγότερο σημαντικά w bits (offset) προσδιορίζουν μια μοναδική λέξη (ΠΡΟΣΟΧΗ: στο προηγούμενο παράδειγμα w=0!!!)
  - 2) Τα πιο σημαντικά s bits (MSBs) καθορίζουν ένα μπλοκ μνήμης
- Τα s bits χωρίζονται σε ένα πεδίο r bits (index) για επιλογή της γραμμής της cache και μια ετικέτα (tag) από s-r bits (περισσότερο σημαντικά)

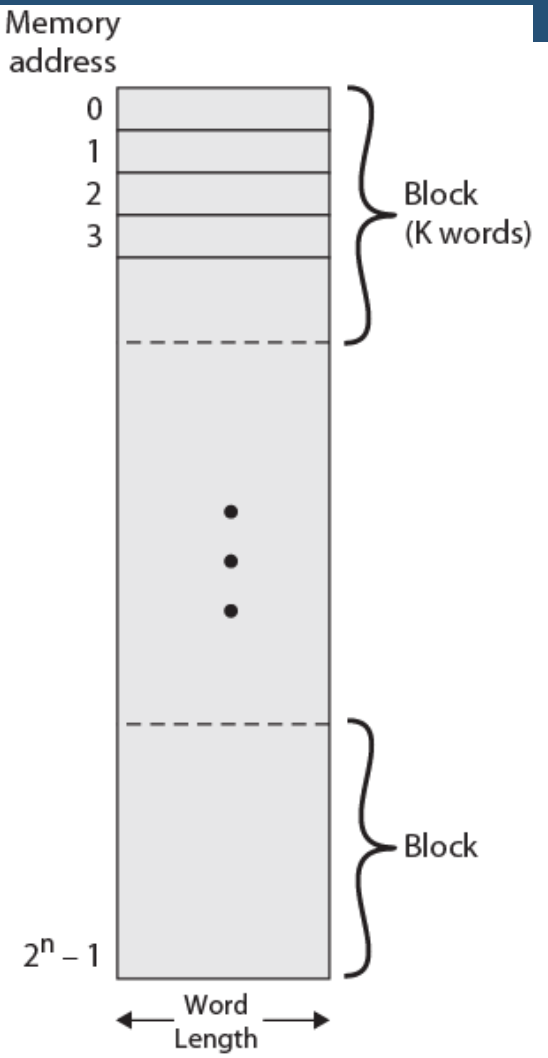
Άμεση χαρτογράφηση (Direct mapping)

- Η διεύθυνση αποτελείται από δύο μέρη:
  - 1) Τα λιγότερο σημαντικά w bits (offset) προσδιορίζουν μια μοναδική λέξη **(ΠΡΟΣΟΧΗ: στο προηγούμενο παράδειγμα w=0!!!)**
- **Δηλαδή θεωρήσαμε (μέχρι στιγμής) ότι το cache block size (κάθε γραμμή της κρυφής μνήμης) χωράει ένα byte**
- **Γιατί είναι κακό αυτό ?**

Δομή Κρυφής/Κύριας Μνήμης



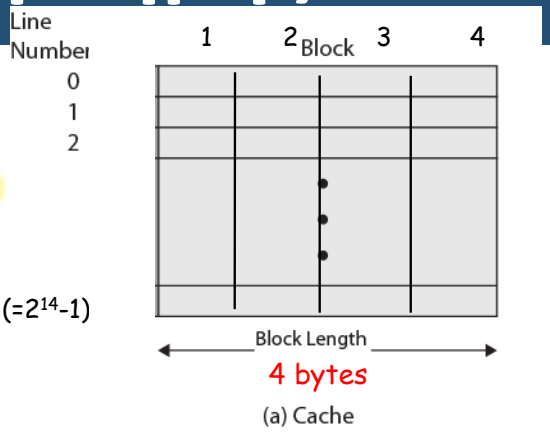
(a) Cache



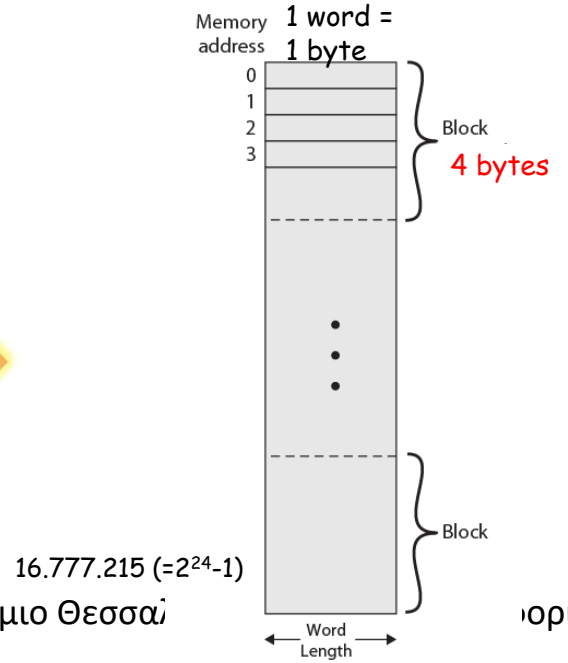
(b) Main memory

Γιατί χαρτογράφηση κρυφής μνήμης;

- Έστω Κρυφή Μνήμη μεγέθους 64kBytes
  - 16384 (2^{14}) γραμμές
  - Block μεγέθους 4 bytes
  - 14-bit διευθύνσεις γραμμών

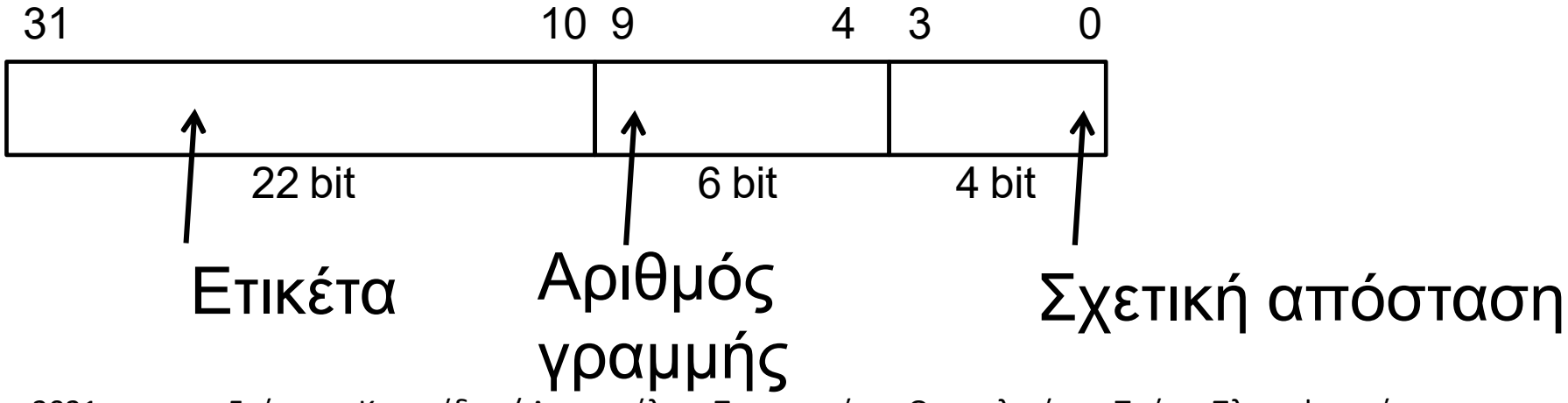


- Έστω Κύρια Μνήμη μεγέθους 16MBytes
  - 16777216 (2^{24}) γραμμές
  - 1 byte ανά γραμμή
  - Block μεγέθους 4 bytes
  - 24-bit διευθύνσεις γραμμών



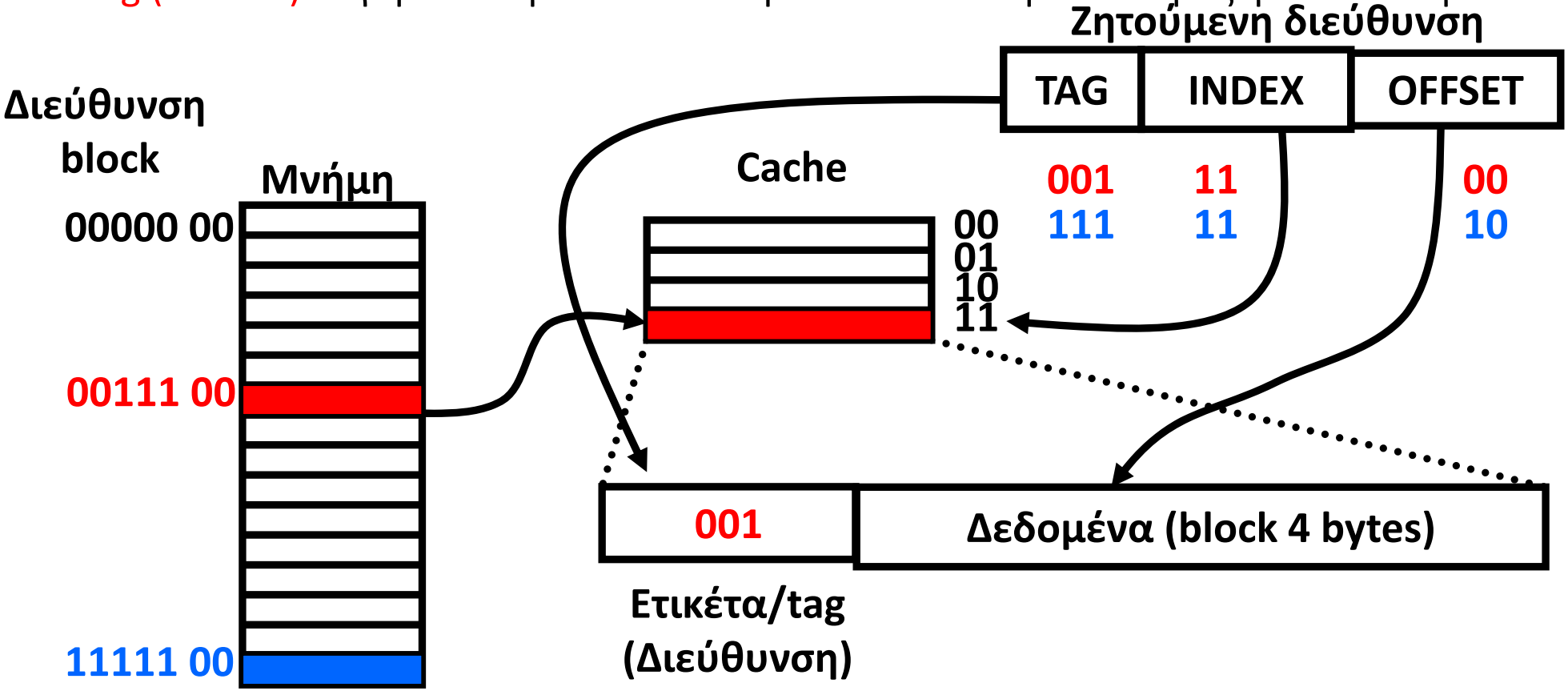
Παράδειγμα: μεγαλύτερο μέγεθος μπλοκ

- **64 μπλοκ, 16 byte/μπλοκ**
- **Σε ποιο αριθμό μπλοκ απεικονίζεται η διεύθυνση 1200;**
- **Διεύθυνση μπλοκ = ⌊1200/16⌋ = 75**
- **Αριθμός μπλοκ = 75 modulo 64 = 11**

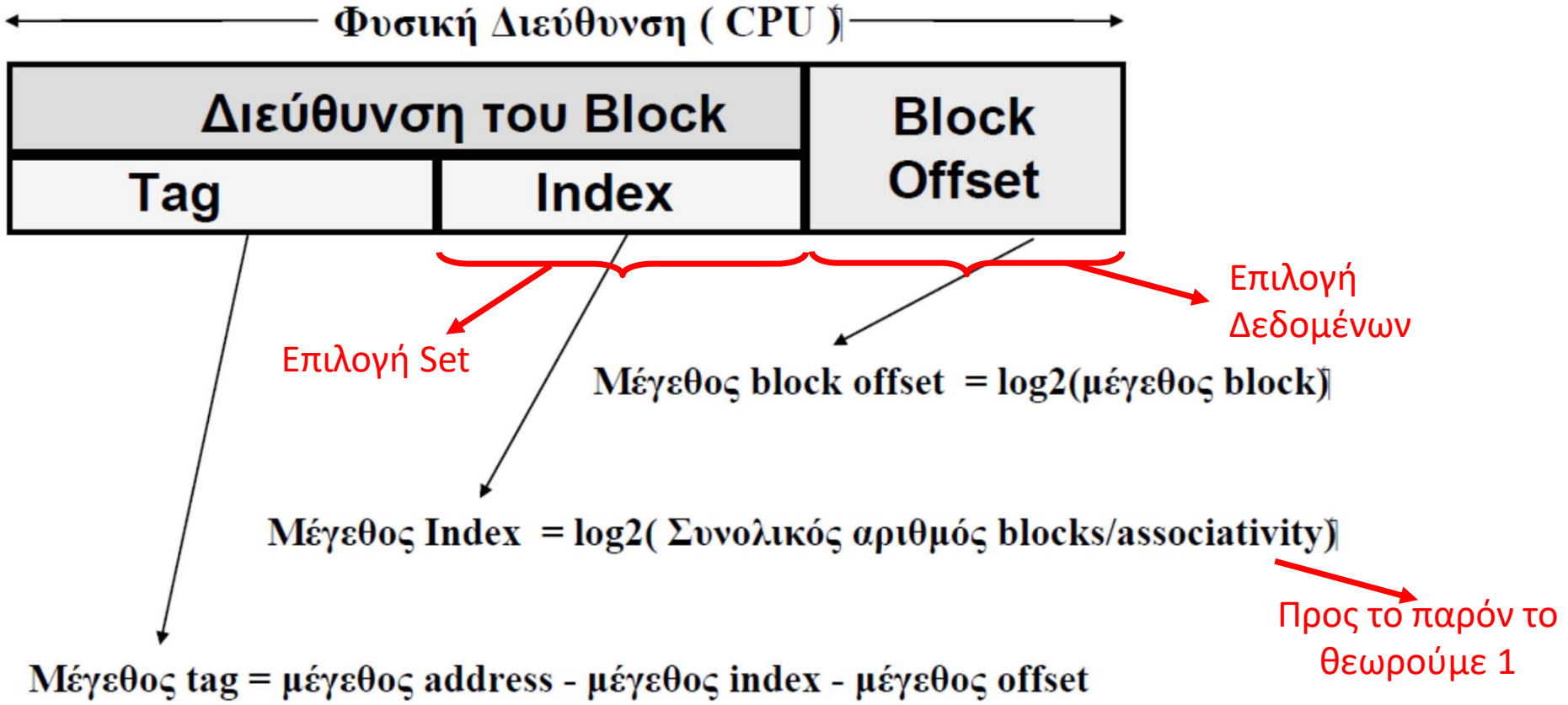


Προσδιορισμός Block

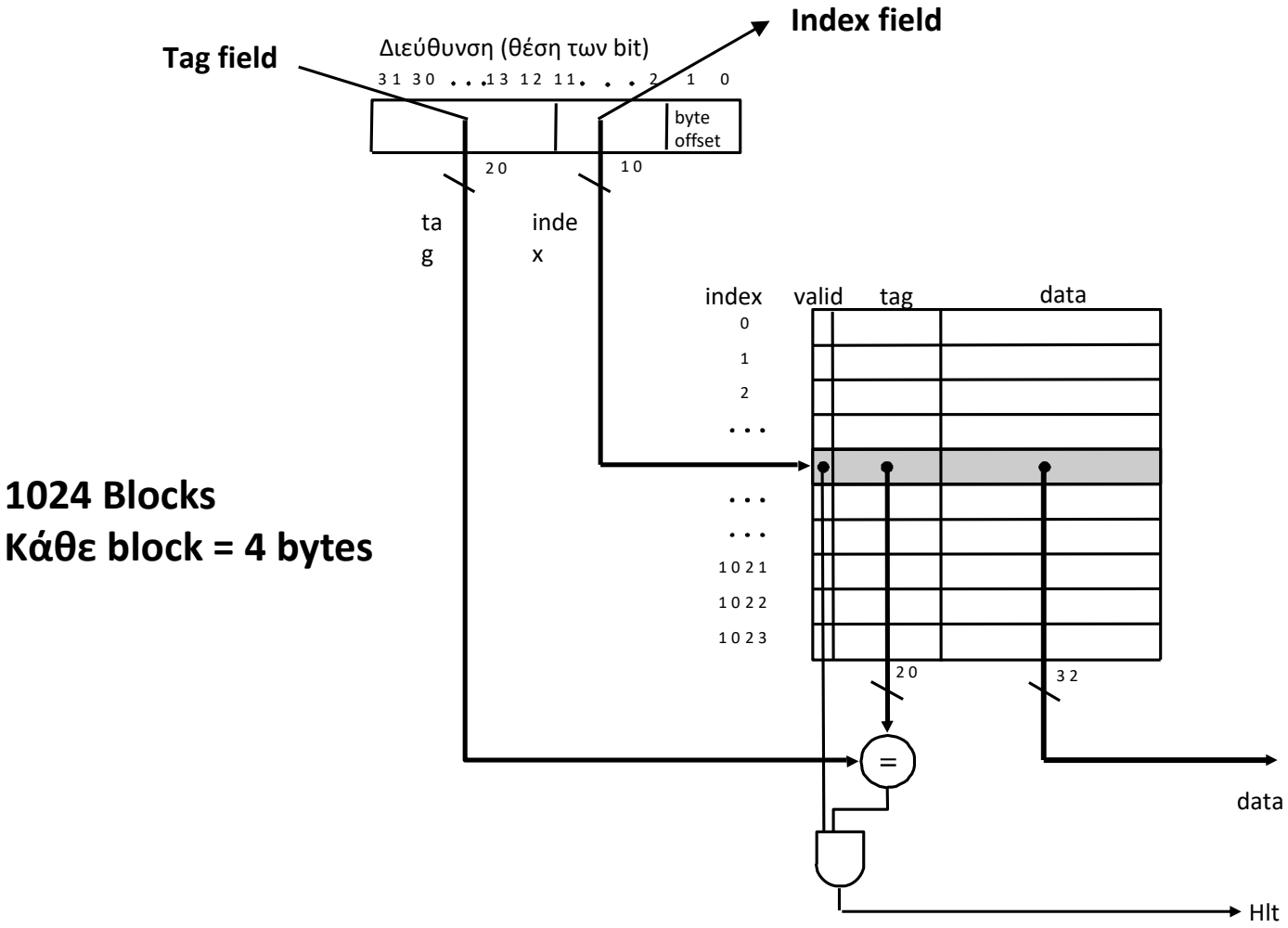
- Ζητούμενη διεύθυνση χωρίζεται σε 3 μέρη: offset (επιλογή λέξης), index (επιλογή γραμμής), tag (ετικέτα)
- Το index μας δίνει τη θέση στη cache που θα ψάξουμε
- Το tag (ετικέτα) συγκρίνεται με αυτό που φυλάσσεται στην cache μαζί με τα δεδομένα



Χωρισμός Διεύθυνσης



Παράδειγμα : Direct Mapped Cache, 4B block



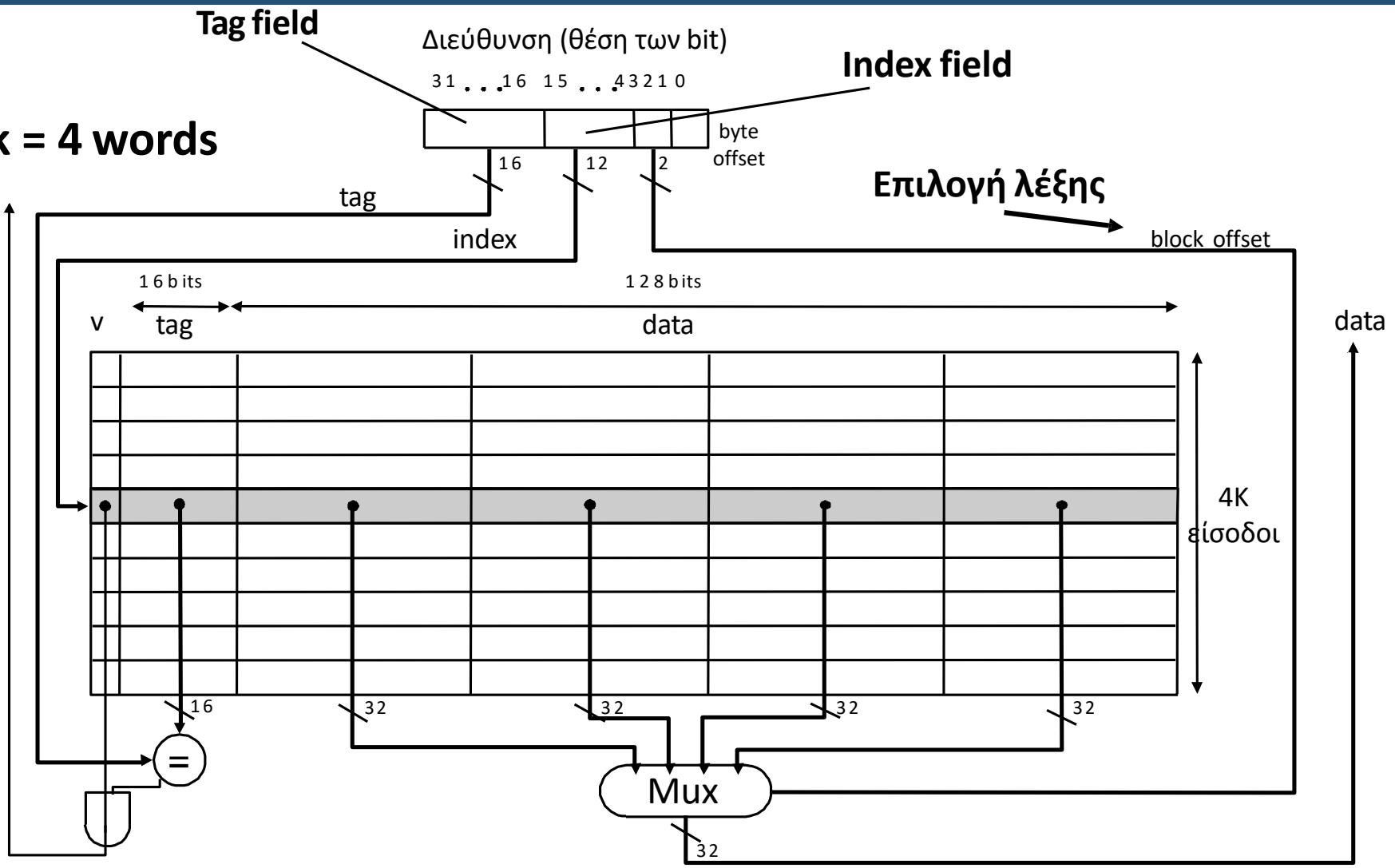
1024 Blocks
Κάθε block = 4 bytes

Παράδειγμα : Direct Mapped Cache, 16B block

4K blocks

Κάθε block = 4 words

Καλύτερη αξιοποίηση του spatial locality



Παράδειγμα Άμεσης Χαρτογράφησης

- **Έστω ΜΜ με 4096 words με ένα word ανά γραμμή. Έστω cache με χωρητικότητα 64 words και μέγεθος block 4 words. Πόσες γραμμές έχει η cache???**
  - **Απάντηση: 16**
- **Ποιο είναι το Σύνολο των bits διεύθυνσης???**
  - **Απάντηση: 12**

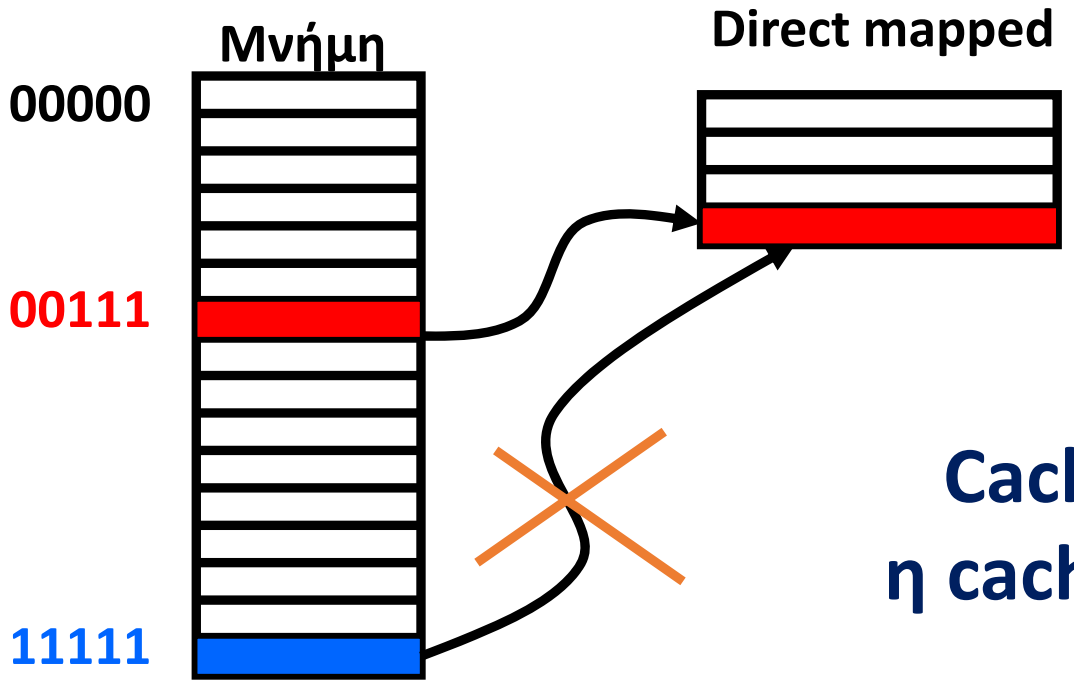
Ετικέτα	Γραμμή	Λέξη
S-R =6	R =4	W =2

Παράδειγμα Άμεσης Χαρτογράφησης

- Έστω στη γραμμή 2 της cache βρίσκεται το μπλοκ της κύριας μνήμης με διεύθυνση 34 (34 Mod 16 = 2) και το αντίστοιχο δυαδικό ψηφίο εγκυρότητας έχει τη λογική τιμή ένα. Η δυαδική παράσταση του 34, χρησιμοποιώντας 10 δυαδικά ψηφία, είναι 000010 0010. Ποια είναι η ετικέτα?
  - Απάντηση: 000010
- Έστω ότι ο επεξεργαστής ζητάει να προσπελάσει τη διεύθυνση 000010 0010 01. Έχουμε hit ή miss???
  - Απάντηση: hit
- Έστω ότι ο επεξεργαστής ζητάει να προσπελάσει τη διεύθυνση 000100 0010 01. Έχουμε hit ή miss???
  - Απάντηση: miss

Άμεση Χαρτογράφηση

- Direct mapped
  - 1-1 αντιστοιχία block μνήμης σε θέση στην cache



Cache miss ενώ η cache είναι άδεια

Άμεση Χαρτογράφηση: Πλεονεκτήματα & Μειονεκτήματα

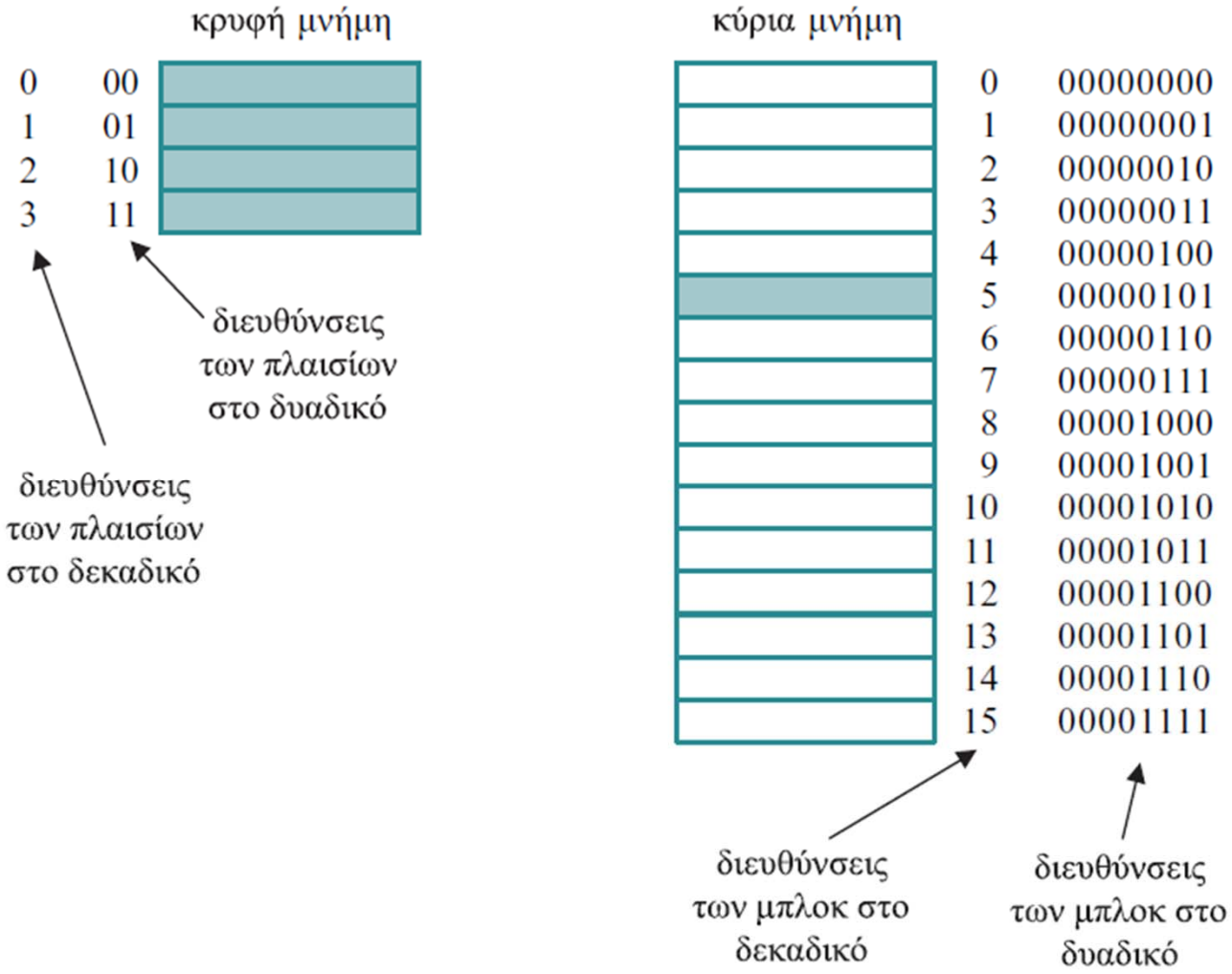
- **Είναι απλή στην κατασκευή της…**
- **…άρα είναι και φτηνή**
- **Μία σταθερή τοποθεσία για συγκεκριμένο μπλοκ… άρα είναι γρήγορη στη λειτουργία της**
  - **Εάν ένα πρόγραμμα προσπελάσει 2 μπλοκ τα οποία χαρτογραφούνται στην ίδια γραμμή επανειλημμένα, οι αστοχίες της κρυφής μνήμης (cache misses) είναι πολύ υψηλές**

Πλήρως Συσχετιστική Χαρτογράφηση

- Ένα μπλοκ κύριας μνήμης μπορεί να φορτωθεί σε οποιαδήποτε γραμμή της κρυφής μνήμης
- Η διεύθυνση μνήμης αποτελείται από 2 μέρη:
  - 1) τα bits της λέξης
  - 2) τα bits της ετικέτας που προσδιορίζουν μοναδικά τα μπλοκ μνήμης
  - Είναι ουσιαστικά σαν έχουμε μια γραμμή στην cache
- Κατά την αναζήτηση στην cache πρέπει να ελέγξουμε τις ετικέτες όλων των γραμμών…
- Η αναζήτηση της Κρυφής Μνήμης γίνεται ολοένα ακριβότερη!!!

Πλήρως Συσχετιστική Χαρτογράφηση

- **Κάθε γραμμή της κύριας μνήμης μπορεί να φορτωθεί σε οποιαδήποτε γραμμή της cache! (πχ η γραμμή με διεύθυνση 5)**



Παράδειγμα Πλήρως Συσχετιστικής Χαρτογράφησης

- **Έστω ΜΜ με 2048 words με ένα word ανά γραμμή. Έστω cache με χωρητικότητα 64 λέξεις και μέγεθος block 2 words. Πόσες γραμμές έχει η cache???**
  - **Απάντηση: 32**
- **Σύνολο bits διεύθυνσης???**
  - **Απάντηση: 11**

Ετικέτα	Γραμμή	Λέξη
S-R =10	R =0	W =1

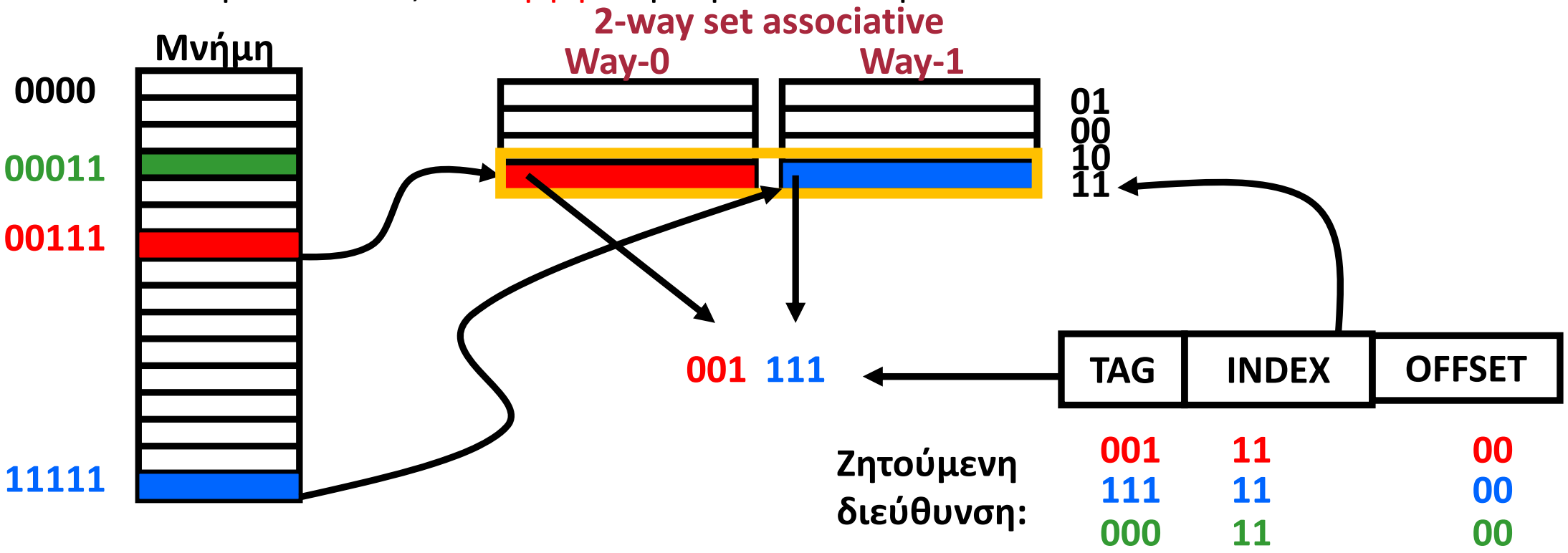
- **Το R είναι ίσο με το μηδέν γιατί στην ουσία δεν υπάρχει μιας και οποιαδήποτε γραμμή της ΜΜ μπορεί να αποθηκευτεί σε οποιαδήποτε γραμμή της cache!**

Πλήρως Συσχετιστική Χαρτογράφηση : Πλεονεκτήματα & Μειονεκτήματα

- **Είναι πολύπλοκη στην κατασκευή της…**
- **…άρα είναι και ακριβή**
- **Όλες οι γραμμές της cache πιθανόν να περιέχουν κάποιο συγκεκριμένο μπλοκ… άρα είναι πιο αργή στη λειτουργία της**
- **Επιτυγχάνει όμως το καλύτερο δυνατό hit ratio!**
- **Μήπως υπάρχει κάποια μέση λύση μεταξύ της άμεσης και της πλήρως συσχετιστικής χαρτογράφησης???**

Block Placement: Set Associative

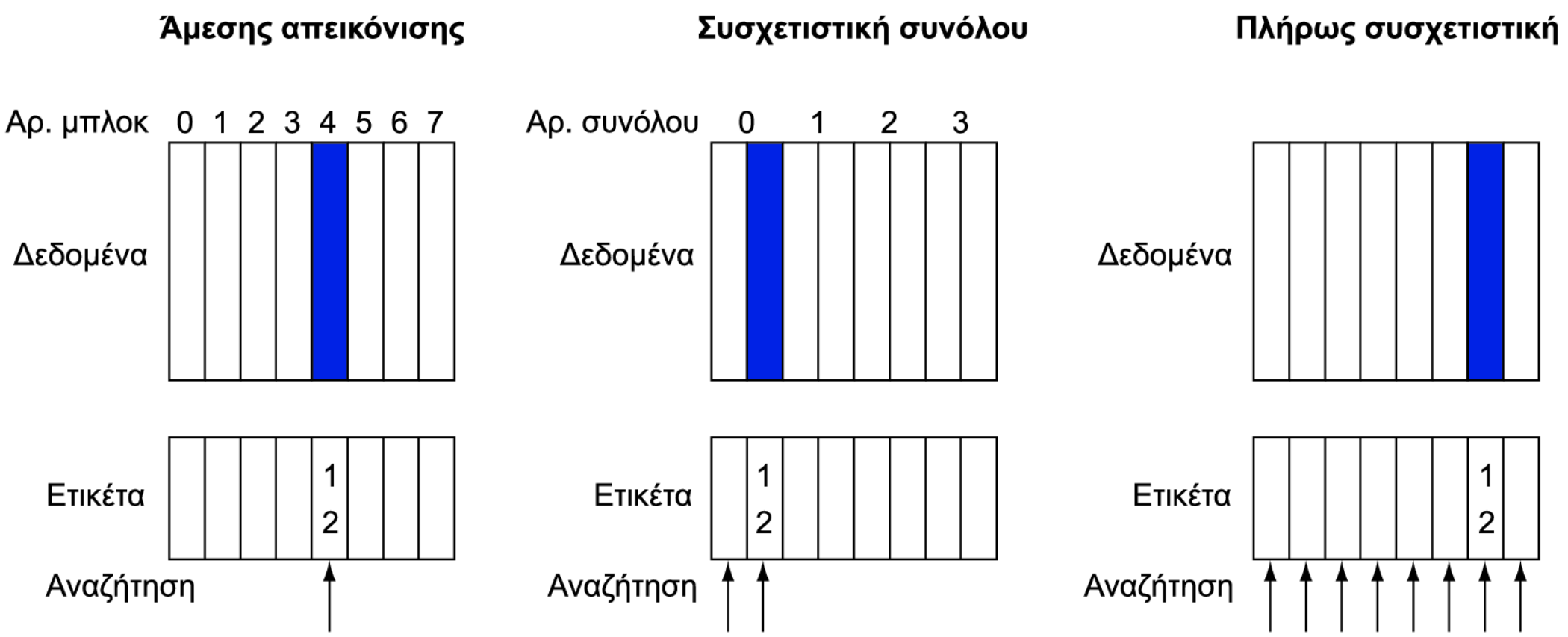
- Set associative (n-way):
  - 1-n αντιστοιχία block μνήμης σε θέσεις στην cache
  - Το index διαλέγει ένα ολόκληρο set
  - Πρέπει να συγκρίνουμε με όλα τα tags στο set για να βρούμε αν κάτι είναι στην cache
  - Καλύτερο Hit Rate, πιο αργή στην προσπέλαση



Συσχετιστική Χαρτογράφηση μέσω συνόλων (N-way associative mapping)

- Η κρυφή μνήμη χωρίζεται σε ένα αριθμό από σύνολα (sets), το καθένα με τη δική του διεύθυνση!!!
- Κάθε σύνολο περιλαμβάνει έναν αριθμό (N) από γραμμές (καλείται N δρόμων)
- Κάθε μπλοκ χαρτογραφείται σε ένα συγκεκριμένο σύνολο με άμεση χαρτογράφηση
- ...και στη συνέχεια με πλήρως συσχετιστική χαρτογράφηση σε μία γραμμή αυτού του συνόλου
- Με άλλα λόγια:
  - Κάθε μπλοκ μπορεί να βρίσκεται σε οποιαδήποτε γραμμή ενός συγκεκριμένου συνόλου
  - π.χ. για σύνολα των 2 γραμμών (2-Way): ένα μπλοκ μπορεί να είναι σε μία μόνο γραμμή και σε ένα μόνο σύνολο

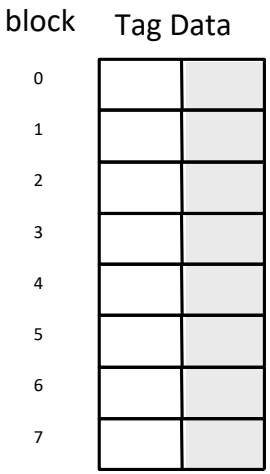
Παράδειγμα


Άμεσης απεικόνισης
Αρ. μπλοκ 0 1 2 3 4 5 6 7
Δεδομένα
Ετικέτα
1
2
Αναζήτηση
Συσχετιστική συνόλου
Αρ. συνόλου 0 1 2 3
Δεδομένα
Ετικέτα
1
2
Αναζήτηση
Πλήρως συσχετιστική
Δεδομένα
Ετικέτα
1
2
Αναζήτηση

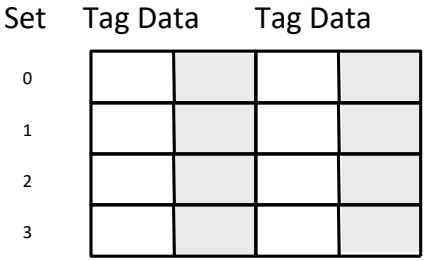
Οργάνωση της Cache : Set Associative Cache

Χωρητικότητα cache : 8 words

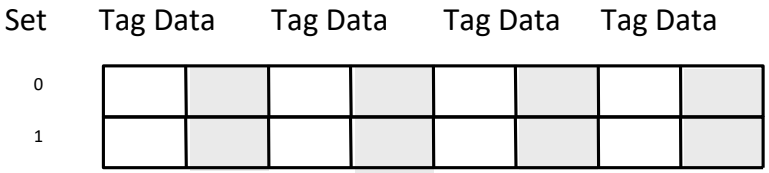
1-way set associative (direct mapped)



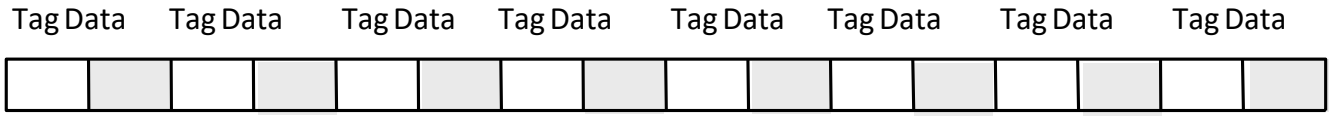
2-way set associative



4-way set associative

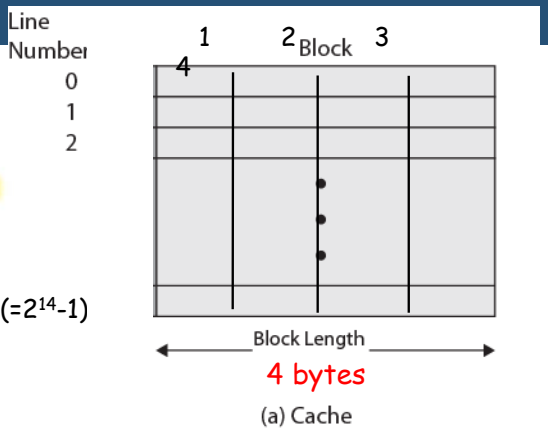


8-way set associative (fully associative)

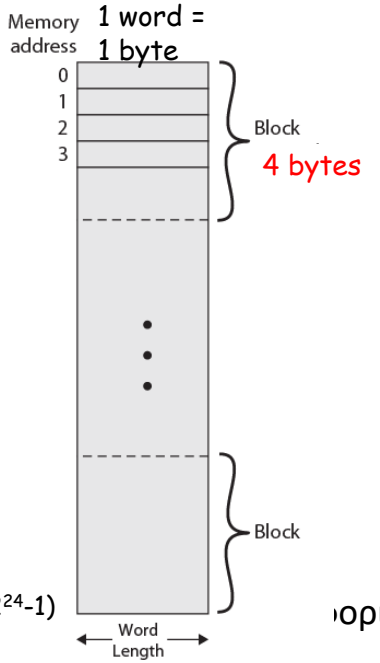


Υπενθύμιση των χαρακτηριστικών του συστήματ

- Έστω Κρυφή Μνήμη μεγέθους 64kBytes
  - 16384 (2^{14}) γραμμές
  - 4 words ανά γραμμή
  - Block μεγέθους 4 bytes
  - 14-bit διευθύνσεις γραμμών



- Έστω Κύρια Μνήμη μεγέθους 16MBytes
  - 16777216 (2^{24}) γραμμές
  - 1 word ανά γραμμή
  - Block μεγέθους 4 bytes
  - 24-bit διευθύνσεις γραμμών



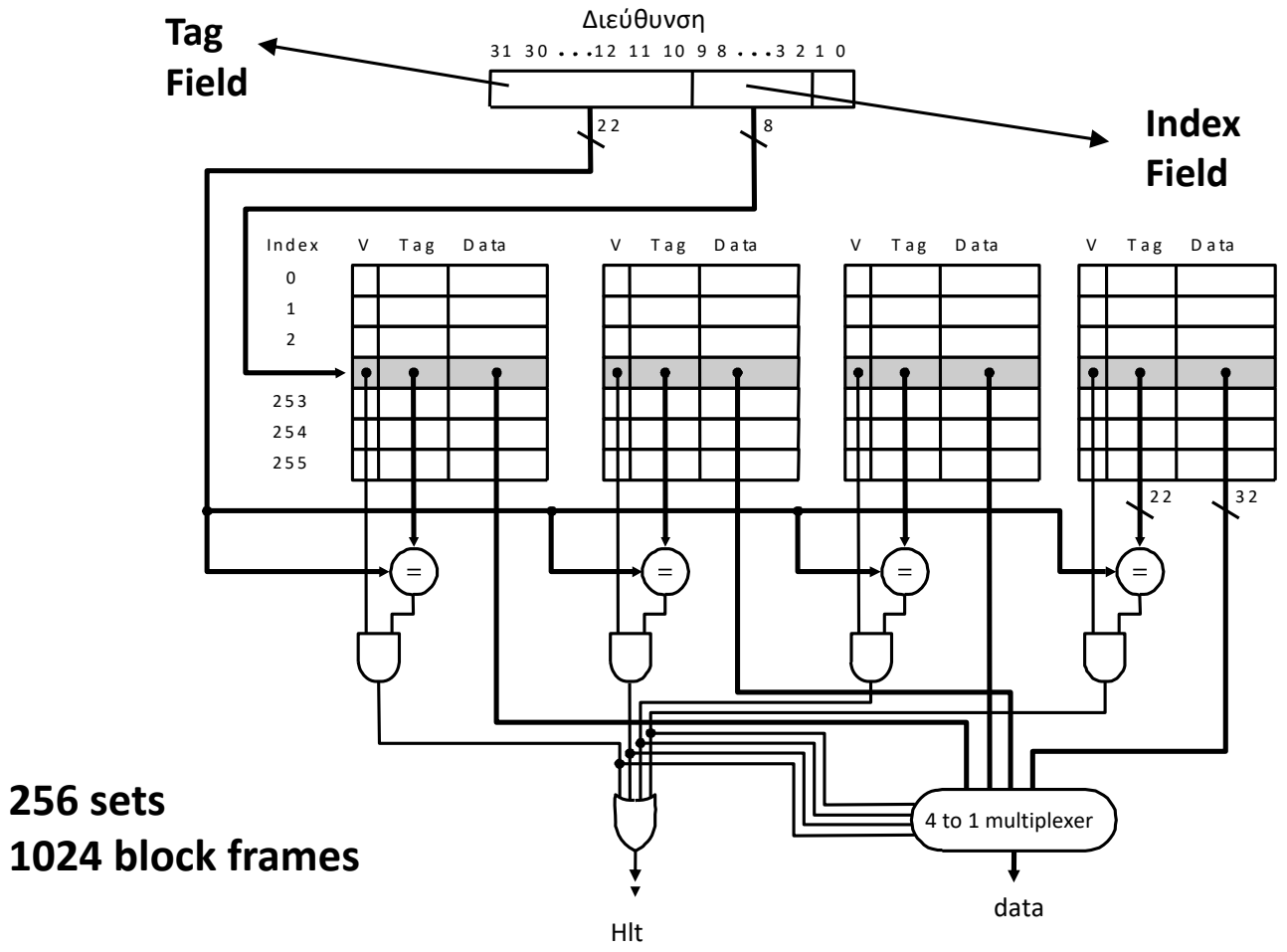
Συσχετιστική Χαρτογράφηση μέσω συνόλων: Δομή Διεύθυνσης για 2-Way

Ετικέτα 9 bit	Σύνολο 13 bit	Λέξη 2 bit

- **Χρησιμοποίηση του πεδίου ενός συνόλου για τον καθορισμό του σετ της κρυφής μνήμης στην οποία θα ψάξει**
- **Σύγκριση πεδίου ετικέτας για έλεγχο επιτυχίας (cache hit)**
- **Π.χ**

Διεύθυνση	Ετικέτα	Δεδομένα	Αριθμός_Συνόλων
1FF7FFC	1FF	12345678	1FFF
0017FFC	001	11223344	1FFF

4-Way Set Associative Cache: (MIPS)



256 sets
1024 block frames

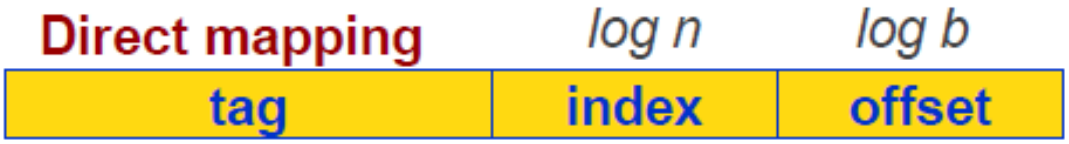
Παράδειγμα

- Έστω ΜΜ με 8192 θέσεις και ένα byte ανά θέση. Έστω cache με χωρητικότητα 64 bytes, μέγεθος block 4 bytes και πλήθος τρόπων ανά σύνολο = 2. Πόσα σύνολα έχει η cache???
  - Απάντηση: 8
- Σύνολο bits διεύθυνσης???
  - Απάντηση: 13

Ετικέτα	Σύνολο	Λέξη
S-R =8	R =3	W =2

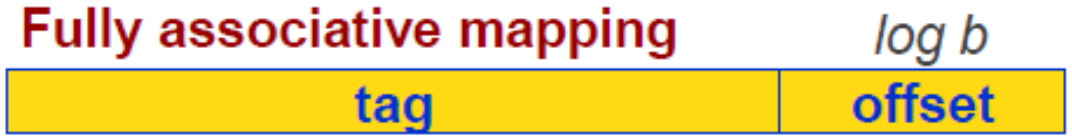
Διευθυσιοδότηση της Cache

- Direct mapped cache: n μπλοκς στην cache



- Set-associative mapping: s sets, n/s blocks ανά set

- Fully associative mapping: πρακτικά η cache έχει ένα set (s = n).



Πόση συσχετιστικότητα;

- **Αυξημένη συσχετιστικότητα μειώνει το ρυθμό αστοχίας**
  - **Αλλά με μειούμενα οφέλη όσο αυξάνεται**
- **Προσομοίωση συστήματος με κρυφή μνήμη δεδομένων (D-cache) 64KB, μπλοκ των 16 λέξεων, μετροπρ/τα SPEC2000**
  - **1 δρόμου: 10.3%**
  - **2 δρόμων: 8.6%**
  - **4 δρόμων: 8.3%**
  - **8 δρόμων: 8.1%**

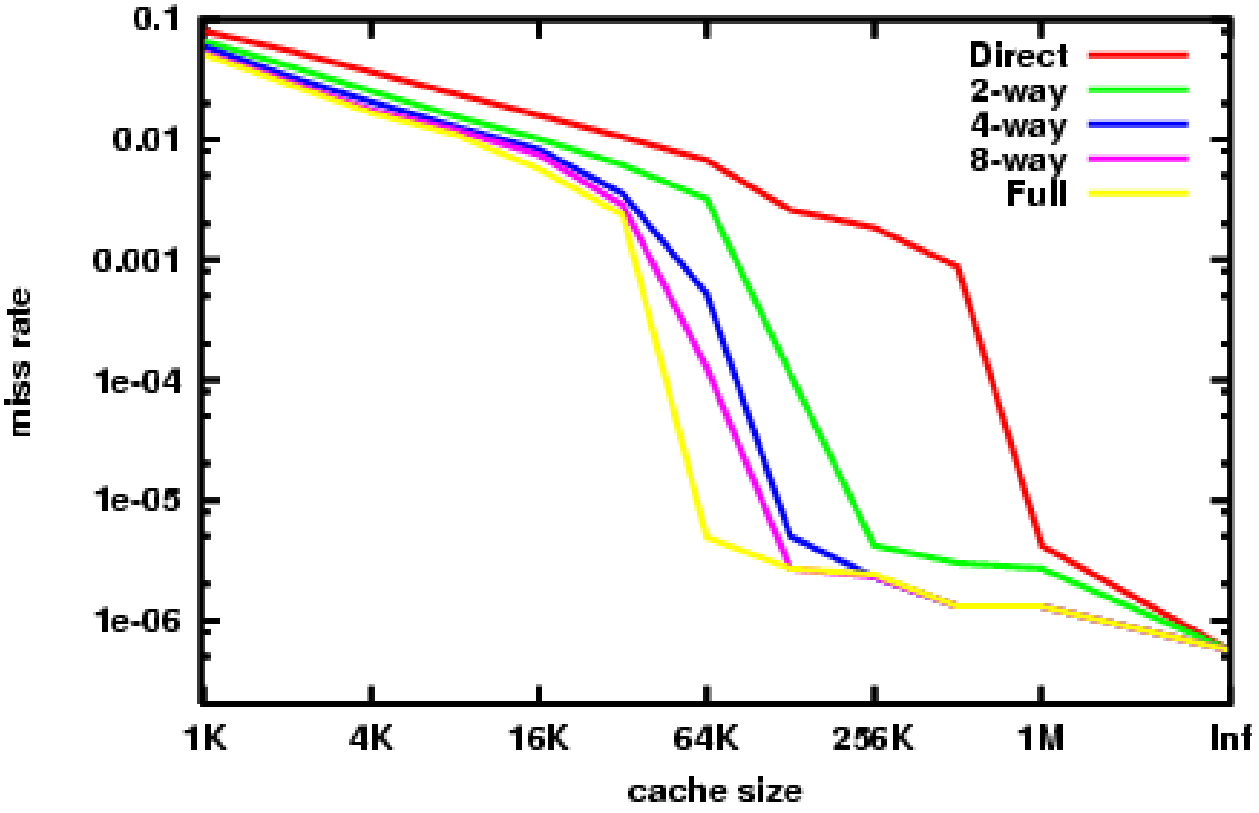
Απόδοση και σύγκριση μεθόδων χαρτογράφησης

- **2 Βασικά Κριτήρια Αξιολόγησης:**
  - **1) Ποσοστό επιτυχίας (Hit ratio)**
  - **2) Ταχύτητα αναζήτησης (search speed)**

Τύπος κρυφής μνήμης	Ποσοστό επιτυχίας	Ταχύτητα αναζήτησης
Direct mapped	Καλό	Η καλύτερη
Fully associative	Το καλύτερο	Μέτρια
N-way set associative, N>1	Πολύ καλό, καλύτερο όσο το N αυξάνεται	Καλή, χειροτερεύει όσο το N αυξάνεται

Σύγκριση μεθόδων χαρτογράφησης ως προς miss rate για SPEC CPU2000 Benchmark suite

- Πλήρως και μερικώς Συσχετιστική cache
- Σημαντική διαφορά έως 64kB για 2-τρόπους
- Διαφορά ανάμεσα σε 2-τρόπους και 4-τρόπους στα 4kB πολύ περισσότερο σε 8kB
- Η πολυπλοκότητα της κρυφή μνήμης αυξάνεται με την συσχέτιση
- Πάνω από 64KB δεν προσφέρει καμία βελτίωση **(αποτελέσματα προσομοίωσης)**



Αστοχίες κρυφής μνήμης

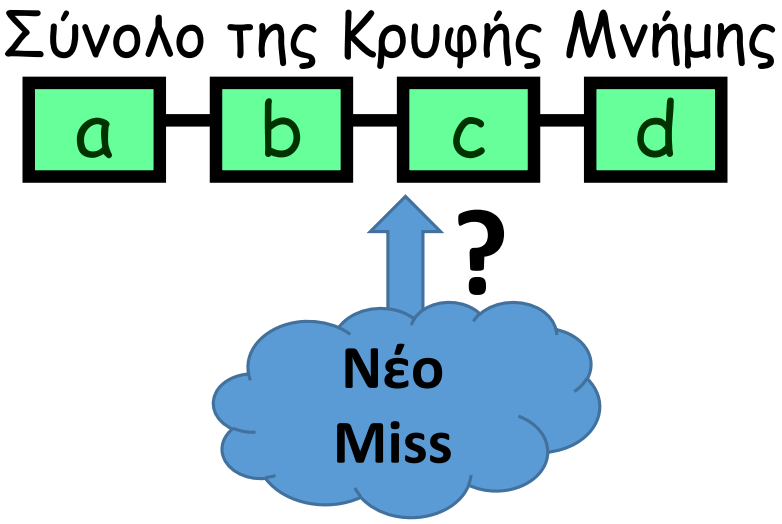
- **Σε περίπτωσης ευστοχίας, η CPU συνεχίζει κανονικά**
- **Σε περίπτωση αστοχίας**
  - **Καθυστερεί η διοχέτευση της CPU**
  - **Προσκομίζει το μπλοκ από το επόμενο επίπεδο της ιεραρχίας**
  - **Αστοχία κρυφής μνήμης εντολών**
    - **Επανεκκίνηση προσκόμισης εντολής**
  - **Αστοχία κρυφής μνήμης δεδομένων**
    - **Ολοκλήρωση προσπέλασης δεδομένων**

4 Ερωτήσεις για τις caches

- **Πού μπορεί να τοποθετηθεί ένα block σε ένα ψηλότερο επίπεδο στην ιεραρχία μνήμης;**
  - **Τοποθέτηση block : direct-mapped, fully associative, set-associative**
- **Πώς βρίσκουμε ένα block στα διάφορα επίπεδα μνήμης;**
  - **Αναγνώριση ενός block : Tag / Block**
- **Ποιο από τα ήδη υπάρχοντα block της cache πρέπει να αντικατασταθεί σε περίπτωση ενός miss;**
  - **Μηχανισμός αντικατάστασης block : Random, Least Recently Used (LRU), FIFO**
- **Τι συμβαίνει όταν μεταβάλλουμε το περιεχόμενο ενός block;**
  - **μηχανισμοί εγγραφής : write-through ή write-back**

Πολιτικές Αντικατάστασης

- Αντικατάσταση του πιο "μη-χρήσιμου" μπλοκ σε περίπτωση αποτυχίας → Στόχος: λόγος επιτυχίας ↑
  - Εφαρμόσιμο μόνο σε κρυφές μνήμες πλήρης και μερικής συσχέτισης
  - Πολύ προσοδοφόρο πεδίο (πολλές δημοσιεύσεις)



Αλγόριθμος Αντικατάστασης

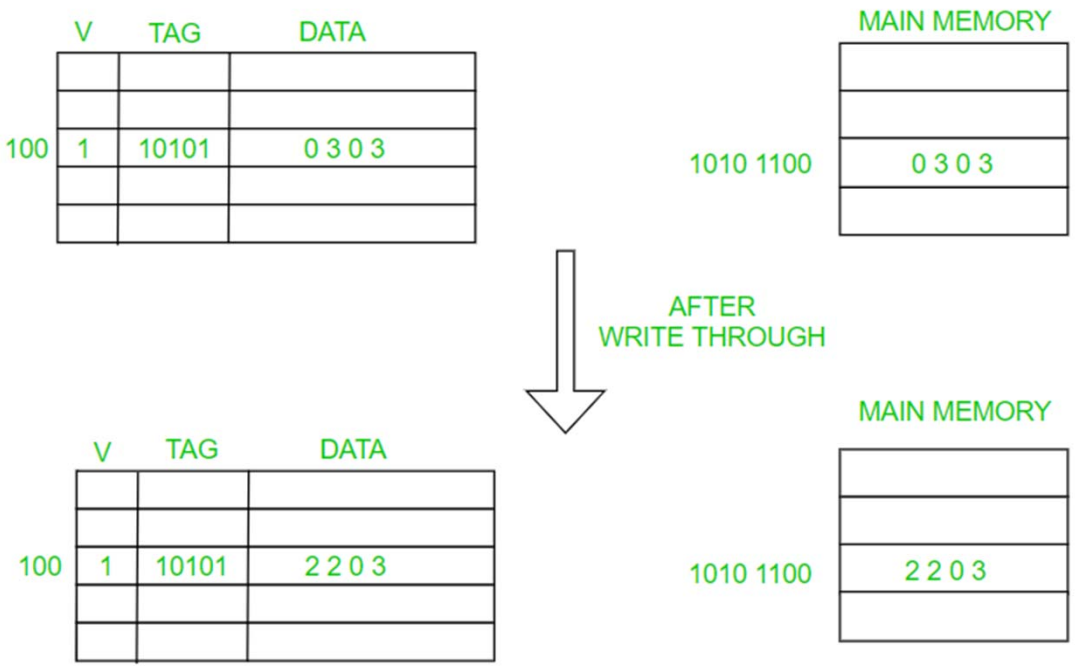
- **Στόχος: Εντοπισμός του "μη-χρήσιμου" μπλοκ**
- **Αλγόριθμος Least Recently Used (LRU)**
  - **- π.χ. συσχέτιση κατά σύνολα με 2 τρόπους: Ποιο από τα δύο μπλοκ έχει να χρησιμοποιηθεί για μεγαλύτερο χρονικό διάστημα?**
  - **Μη χρησιμοποιηθέντος πρόσφατα (Least Recently Used, LRU): καλή απόδοση, μέτριο κόστος υλοποίησης**
- **Αλγόριθμος First in first out (FIFO)**
  - **Αντικατάσταση του μπλοκ που είναι για μεγαλύτερο διάστημα στην κρυφή μνήμη**
  - **Αυτό που μπήκε πρώτο (το πιο "παλιό") (FIFO): καλή απόδοση, μεγαλύτερο κόστος υλοποίησης**
- **Αλγόριθμος Least Frequently Used (LFU)**
  - **Αντικατάσταση του μπλοκ το οποίο έχει τα λιγότερα hits**
  - **Αυτό που έχει τις λιγότερες αναφορές (Least Frequently Used, LFU): καλή απόδοση, ακόμα μεγαλύτερο κόστος υλοποίησης**
- **Αλγόριθμος Random Replacement (RR)**

Πολιτική Εγγραφής

- Τι γίνεται όταν τροποποιούμε τα δεδομένα μίας γραμμής της cache?
- Σε ευστοχία εγγραφής δεδομένων, θα μπορούσε να γίνει μόνο ενημέρωση του μπλοκ στην κρυφή μνήμη
  - Αλλά τότε η κρυφή μνήμη και η μνήμη θα είναι ασυνεπείς
- Πότε πρέπει να ενημερωθεί η κύρια μνήμη για την αλλαγή?
- Τι γίνεται αν έχουμε πολλούς πυρήνες στη CPU που έχουν πολλές ατομικές κρυφές μνήμες - άρα και πολλαπλά αντίγραφα των ίδιων δεδομένων?
- Πρέπει να αποφύγουμε την απώλεια των δεδομένων

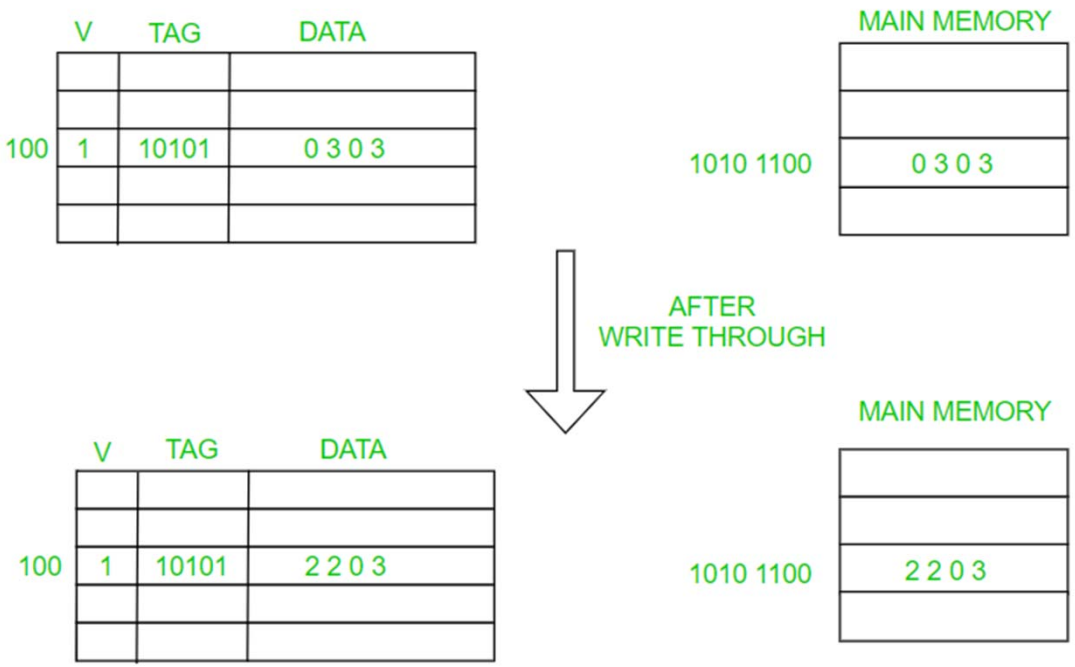
Πολιτική “Write through”

- **Ταυτόχρονη εγγραφή (write through): ενημέρωσε και τη μνήμη**
  - **Όλες οι εγγραφές πηγαίνουν στην κύρια μνήμη ΚΑΙ στην κρυφή μνήμη**
  - **Πολλαπλές CPUs παρακολουθούν την κυκλοφορία στην κύρια μνήμη για να κρατούν ενημερωμένη την τοπική κρυφή μνήμη**
  - **Αύξηση της κυκλοφορίας στους διαύλους**



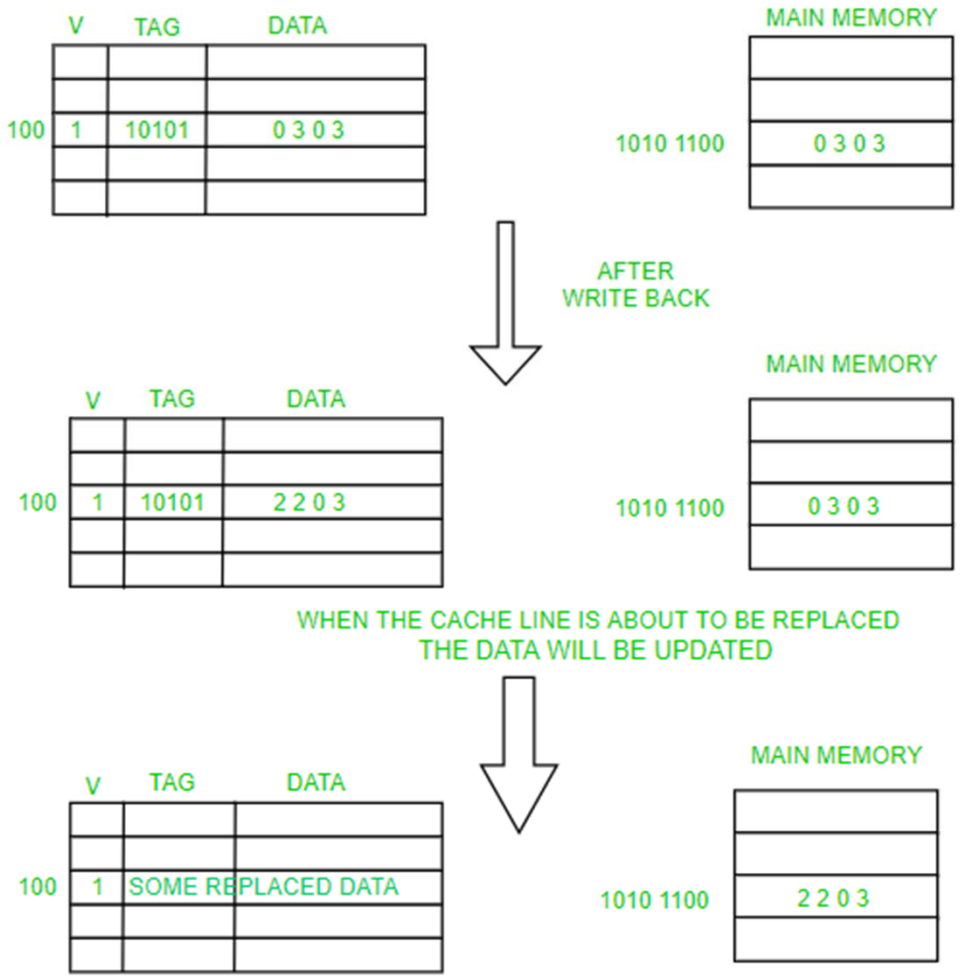
Πολιτική “Write through”

- **Αποτέλεσμα: οι εγγραφές να διαρκούν περισσότερο**
  - **π.χ., αν το βασικό CPI είναι 1, το 10% των εντολών είναι STORE και η εγγραφή στη μνήμη διαρκεί 100 κύκλους**
  - **Πραγματικό CPI = 1 + 0.1×100 = 11**



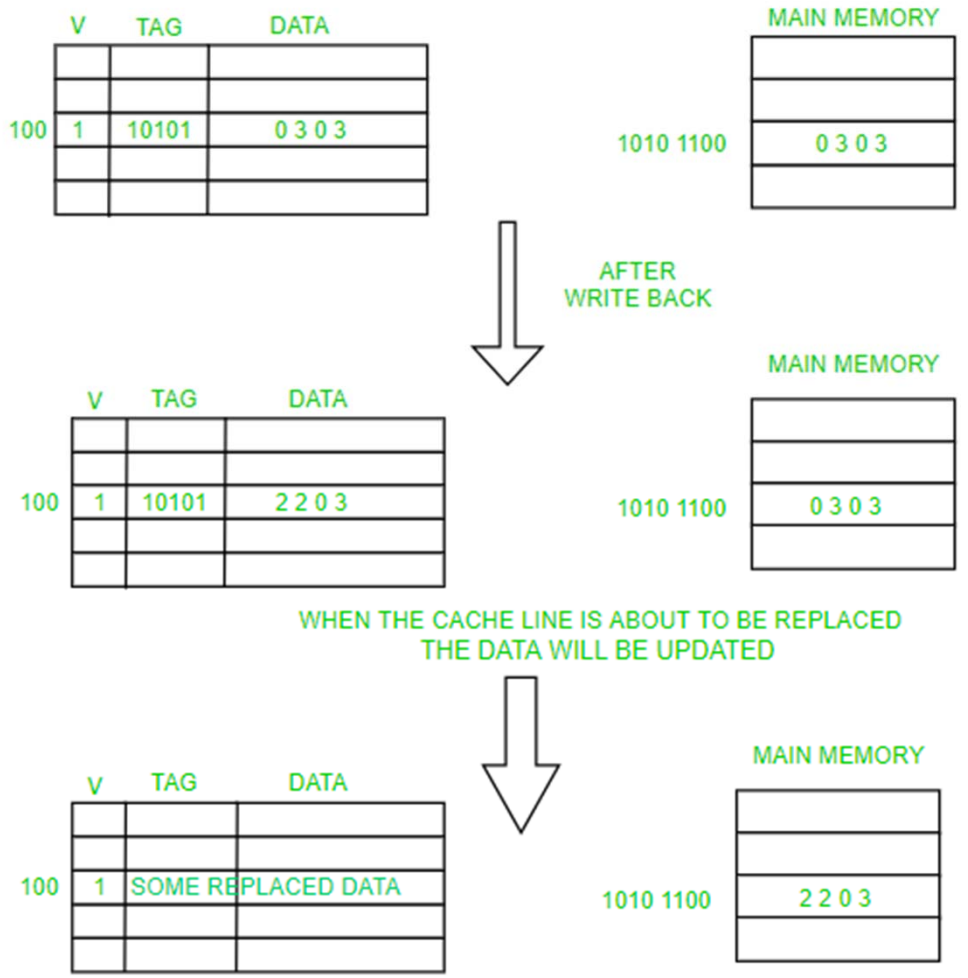
Πολιτική “Write back”

- **Εναλλακτική: σε hit εγγραφής δεδομένων, ενημέρωσε μόνο το μπλοκ στην κρυφή μνήμη**
  - **Διάκριση «καθαρών» (clean – δεν έχουν γραφεί) και «ακάθαρτων» (dirty – έχουν γραφεί) μπλοκ**
  - **Bit “ενημέρωσης” (ή Dirty Bit) ορίζεται όταν μια ενημέρωση πραγματοποιείται**



Πολιτική “Write back”

- **Εάν ένα μπλοκ πρόκειται να αντικατασταθεί, πραγματοποιείται εγγραφή στην κύρια μνήμη μόνο αν το Dirty Bit έχει την τιμή ένα**
- **Το 15% των αναφορών στην μνήμη είναι εγγραφές**



Κατανομή εγγραφών (Write allocation)

- Τι πρέπει να γίνει σε αστοχία εγγραφής;
- Εναλλακτικές για ταυτόχρονη εγγραφή
  - Κατανομή σε αστοχία (allocate on miss): προσκόμιση του μπλοκ
  - Εγγραφή από γύρω (write around): όχι προσκόμιση του μπλοκ
    - Αφού τα προγράμματα συχνά γράφουν ένα ολόκληρο μπλοκ πριν το διαβάσουν (π.χ., απόδοση αρχικών τιμών)

Μέτρηση απόδοσης κρυφής μνήμης

- **Συστατικά του χρόνου CPU**
  - **Κύκλοι εκτέλεσης προγράμματος**
    - **Περιλαμβάνει το χρόνο ευστοχίας κρυφής μνήμης**
  - **Κύκλοι καθυστέρησης (stall) μνήμης**
    - **Κυρίως από αστοχίες κρυφής μνήμης**
- **Με απλουστευτικές παραδοχές:**

$$\text{Memory stall cycles} = \frac{\text{Memory accesses}}{\text{Program}} \times \text{Miss rate} \times \text{Miss penalty}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Misses}}{\text{Instruction}} \times \text{Miss penalty}$$

Παράδειγμα απόδοσης κρυφής μνήμης

- Δίνονται
  - Ρυθμός αστοχίας κρυφής μνήμης εντολών (I-cache) = 2%
  - Ρυθμός αστοχίας κρυφής μνήμης δεδομένων (D-cache) = 4%
  - Ποινή αστοχίας = 100 κύκλοι
  - Βασικό CPI (ιδανική κρυφή μνήμη) = 2
  - Οι εντολές load & store είναι το 36% των εντολών
- Κύκλοι αστοχίας ανά εντολή
  - I-cache: 0.02 × 100 = 2
  - D-cache: 0.36 × 0.04 × 100 = 1.44
- Πραγματικό CPI = 2 + 2 + 1.44 = 5.44
  - Η ιδανική CPU είναι 5.44/2 =2.72 φορές ταχύτερη

Μέσος χρόνος προσπέλασης

- Ο χρόνος ευστοχίας είναι επίσης σημαντικός για την απόδοση
- Μέσος χρόνος προσπέλασης μνήμης (Average memory access time – AMAT)
  - AMAT = Χρόνος ευστοχίας + Ρυθμός αστοχίας × Ποινή αστοχίας
- Παράδειγμα
- CPU με ρολόι του 1 ns, χρόνο ευστοχίας = 1 κύκλος, ποινή αστοχίας = 20 κύκλοι, ρυθμός αστοχίας I-cache = 5%
  - AMAT = 1 + 0.05 × 20 = 2ns
    - 2 κύκλοι ανά εντολή

Περίληψη της απόδοσης

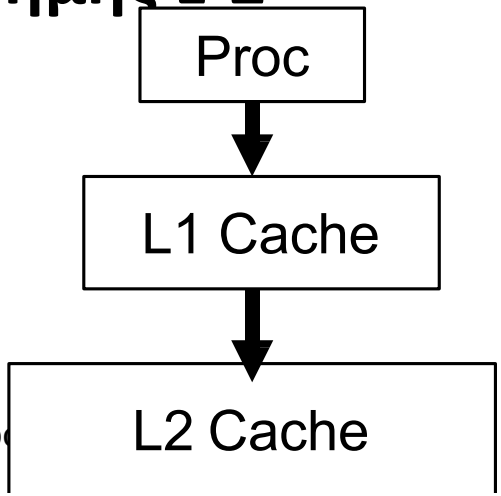
- Όταν αυξάνει η απόδοση της CPU
  - Η ποινή αστοχίας γίνεται πιο σημαντική
- Μείωση του βασικού CPI
  - Μεγαλύτερο ποσοστό του χρόνου δαπανάται σε καθυστερήσεις μνήμης
- Αύξηση του ρυθμού ρολογιού
  - Οι καθυστερήσεις μνήμης (σταθερού χρόνου) αποτελούν περισσότερους κύκλους CPU
- Δεν μπορούμε να αγνοήσουμε τη συμπεριφορά της κρυφής μνήμης όταν αξιολογούμε την απόδοση του συστήματος

Ενοποιημένες vs Διαχωριζόμενες Κρυφές Μνήμε

- **Μία κρυφή μνήμη για δεδομένα και εντολές ή δύο, μια για δεδομένα και μία για εντολές**
- **Πλεονεκτήματα μιας ενοποιημένης κρυφής μνήμης (Unified L1 cache)**
  - **Ισοσταθμίζει την φόρτωση των εντολών και την φόρτωση των δεδομένων (είναι επιθυμητό???)**
  - **Μόνο μία κρυφή μνήμη για σχεδιασμό & υλοποίηση**
- **Πλεονεκτήματα της διαχωριζόμενης κρυφής μνήμης (Separate L1i & L1d caches)**
  - **Σημαντική στην διασωλήνωση**

Πολυεπίπεδες κρυφές μνήμες

- Κύρια κρυφή μνήμη (L-1) συνδέεται με τη CPU
  - Μικρή, αλλά γρήγορη
- Η κρυφή μνήμη δευτέρου επιπέδου (level-2 cache) εξυπηρετεί αστοχίες της κύριας κρυφής μνήμης
  - Μεγαλύτερη, πιο αργή, αλλά και πάλι ταχύτερη από τη κύρια μνήμη
- Η κύρια μνήμη εξυπηρετεί αστοχίες της κρυφής μνήμης L-2
- Μερικά/πολλά συστήματα υψηλών επιδόσεων περιλαμβάνουν και κρυφή μνήμη L-3



Intel x86 Κρυφή Μνήμη

- 80386 – οχι κρυφή μνήμη on chip
- 80486 – 8k χρησιμοποιεί 16 byte γραμμές και 4 τρόπους συσχέτισης κατά σετ
- Pentium (όλες οι εκδόσεις) – δύο on chip L1 κρυφές μνήμες
- δεδομένα & εντολές
- Pentium III – L3 κρυφή μνήμη που προστίθεται σε off chip

Pentium 4 Κρυφή Μνήμη

Pentium 4

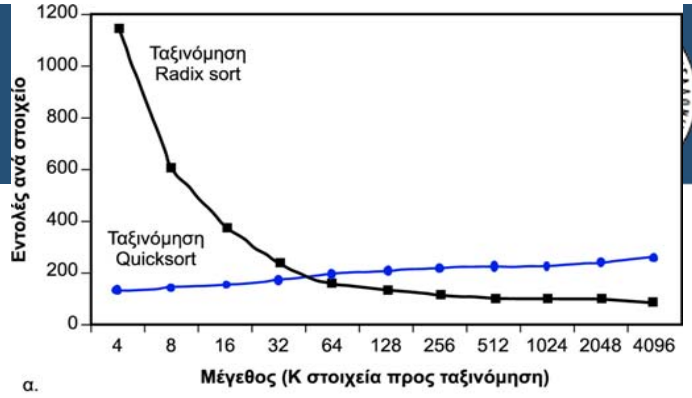
- **L1 κρυφή μνήμη**
  - **8k bytes**
  - **64 byte γραμμές**
  - **4 τρόποι συσχέτισης κατά σετ**
- **L2 κρυφή μνήμη**
  - **Τροφοδοτεί δυο L1 κρυφές μνήμες**
  - **256k**
  - **128 byte γραμμές**
  - **8 τρόποι συσχέτισης κατά σετ**
- **L3 κρυφή μνήμη on chip**

Ζητήματα πολυεπίπεδων κρυφών μνημών

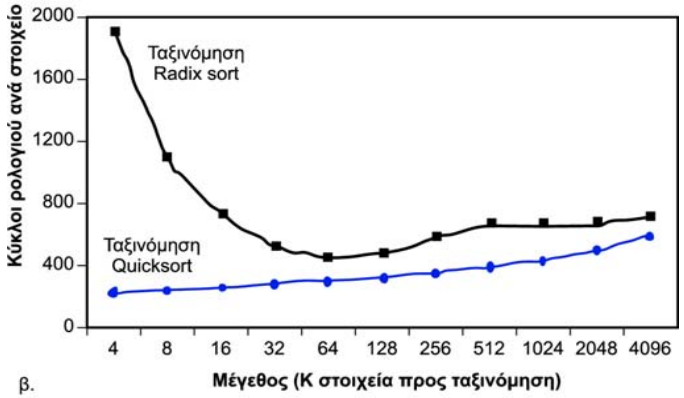
- **Κύρια κρυφή μνήμη L-1**
  - **Εστιάζει στον ελάχιστο χρόνο ευστοχίας**
- **Κρυφή μνήμη L-2**
  - **Εστιάζει στο χαμηλό ρυθμό αστοχίας για να αποφύγει τις προσπελάσεις της κύριας μνήμης**
  - **Ο χρόνος ευστοχίας έχει μικρότερη συνολική επίδραση**
- **Αποτελέσματα**
  - **Η κρυφή μνήμη L-1 είναι μικρότερη**
  - **Το Associativity της L-2 είναι μεγαλύτερο!**
  - **Το μέγεθος μπλοκ της L-1 είναι μικρότερο από το μέγεθος μπλοκ της L-2**

Αλληλεπιδράσεις με λογισμικό

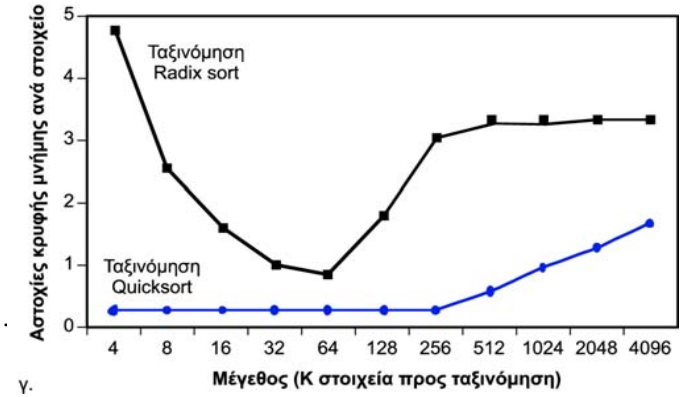
- **Οι αστοχίες εξαρτώνται από τα μοτίβα προσπέλασης μνήμης**
- **Συμπεριφορά του αλγορίθμου**
- **Βελτιστοποίηση του μεταγλωττιστή για προσπελάσεις μνήμης**



α.



β.



γ.

Κλείνοντας …

- **Κεφάλαιο 1: Όλες οι ενότητες**
  - **Εισαγωγή, Απόδοση**
- **Κεφάλαιο 2: Οι ενότητες 2.1 έως 2.14**
  - **Αρχιτεκτονική Συνόλου Εντολών**
- **Κεφάλαιο 3: Όλες οι ενότητες εκτός της ενότητας 3.7**
  - **Αριθμητική Υπολογιστών**
- **Κεφάλαιο 4: Οι ενότητες 4.1 έως 4.7**
  - **Σχεδίαση ενός κύκλου, Σχεδίαση με Διοχέτευση**
- **Κεφάλαιο 5: Οι ενότητες 5.1 έως 5.3**
  - **Ιεραρχίας Μνήμης (κυρίως κρυφές μνήμες)**

Επόμενα Μαθήματα …

- **Συνέχεια στο μάθημα Προχωρημένα Θέματα Αρχιτεκτονικής Συστημάτων (6ο Εξάμηνο)**
  - **Βελτιστοποίηση (ταχύτητα & κατανάλωση ισχύος) των κρυφών μνημών σε επίπεδο υλικού & μεταγλωττιστή**
  - **Τεχνικές αντικαταστάσης και εκ των προτέρων προσκόμιση δεδομένων (prefetching)**
  - **Συστήματα μνήμης πολυπύρηνων επεξεργαστών**
    - **Συνάφεια και συνέπειας μνήμης**
  - **Άλλες μορφές παραλληλισμού: helper threads, thread level speculation, speculative precomputation, run-ahead execution, transactional memories**
  - **Συστήματα μνήμης GPU & Επεξεργαστών Επιτάχυνσης Νευρωνικών Δικτυών**
  - **Η trace cache σε hyperthreading αρχιτεκτονικές.**
  - **Επιθέσεις Ασφαλείας στο Υποσύστημα Μνήμης (Specte και Meltdown)**