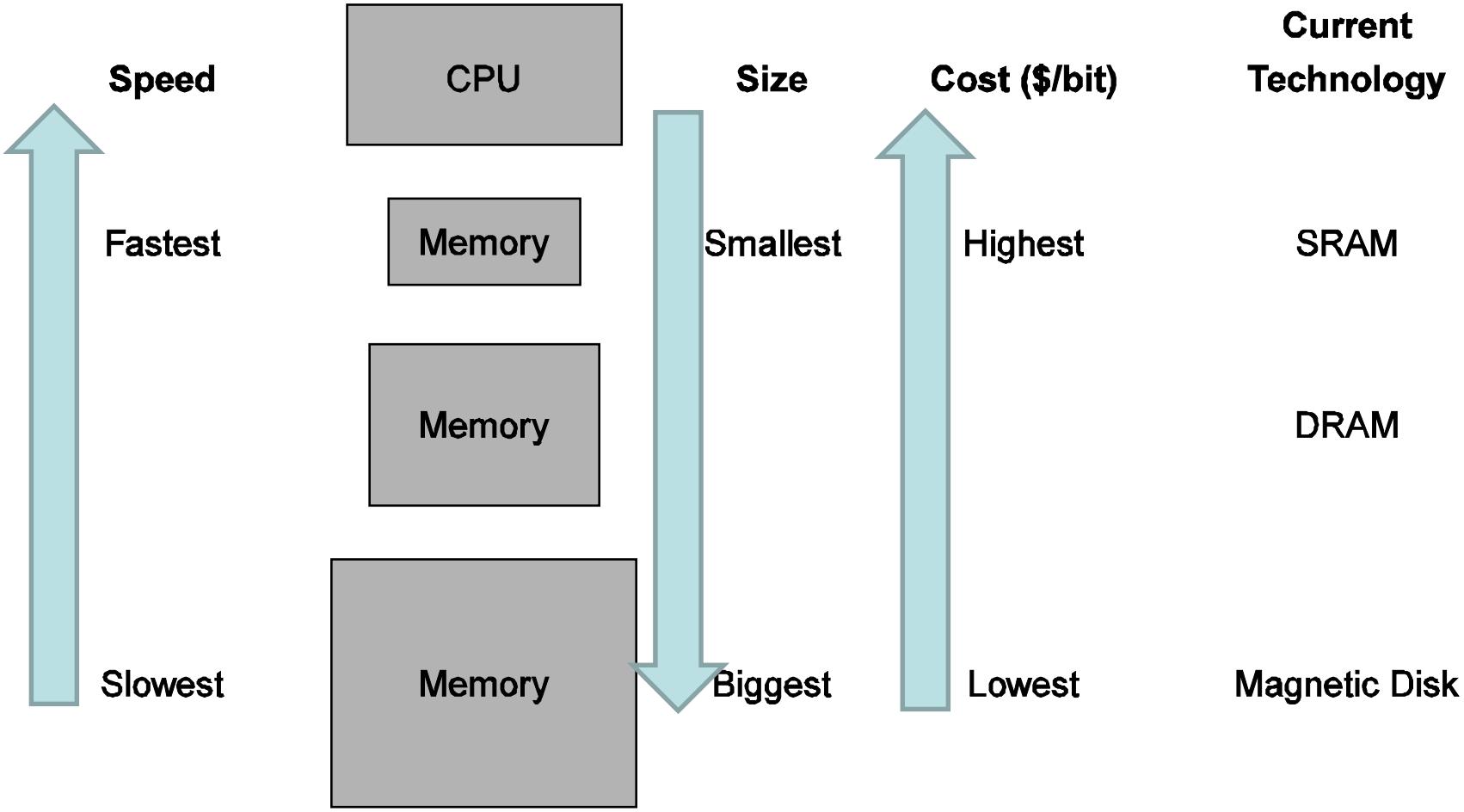
Επανάληψη Ιεραρχία Μνήμης
Memory Hierarchy

Ιδεατά θέλουμε να ισχύει για μια μνήμη:
Άπειρη, γρήγορη και φτηνή μνήμη



Πραγματικότητα ☹

Υπάρχει λύση;

Τοπικότητα Αναφοράς/Χρήσης

(Locality of Reference)

• Προγράμματα τείνουν να χρησιμοποιούν δεδομένα και εντολές που χρησιμοποίησαν πρόσφατα.

• **Κανόνας του 90/10** (90/10 Rule): Ένα πρόγραμμα ξοδεύει 90% του χρόνου εκτέλεσης σε μόνο 10% του κώδικα

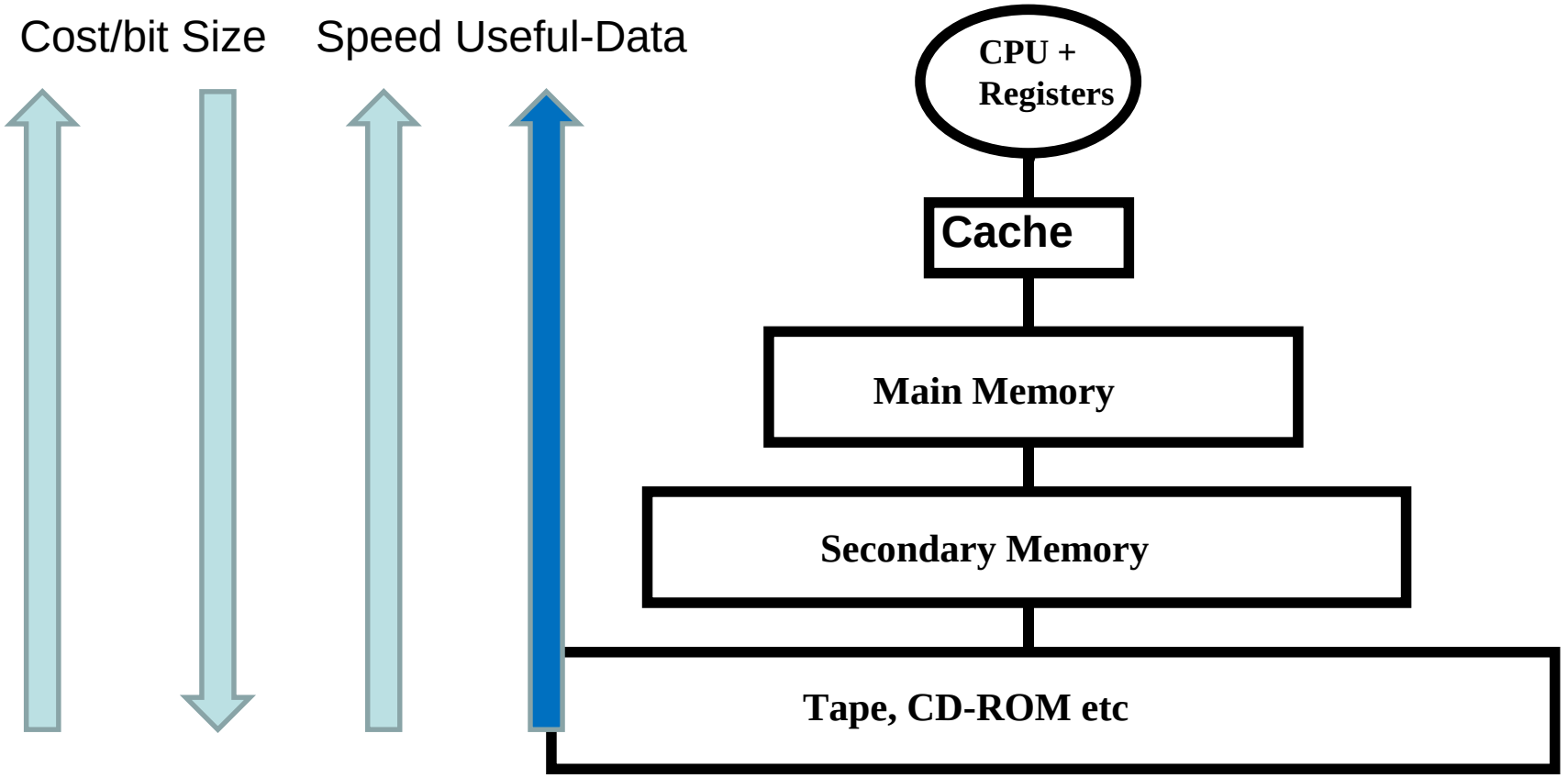
• **Χρονική Τοπικότητα Αναφοράς**: (Temporal Locality): Αντικείμενα που χρησιμοποιήθηκαν πρόσφατα τείνουν να χρησιμοποιηθούν ξανά στο εγγύς μέλλον

• **Χωρική Τοπικότητα Αναφοράς** (Spatial locality): Αντικείμενα που έχουν γειτονικές διευθύνσεις στον χώρο τείνουν να χρησιμοποιούνται και γειτονικά στον χρόνο.

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

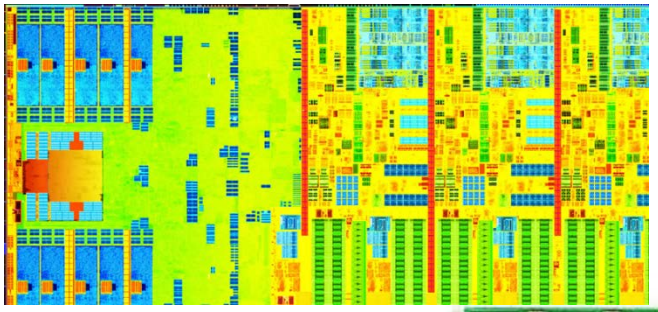
	Spatial	Temporal
i		
n		
a[i]		
s		
for (i=0;i<n;++i) s+=a[i];		

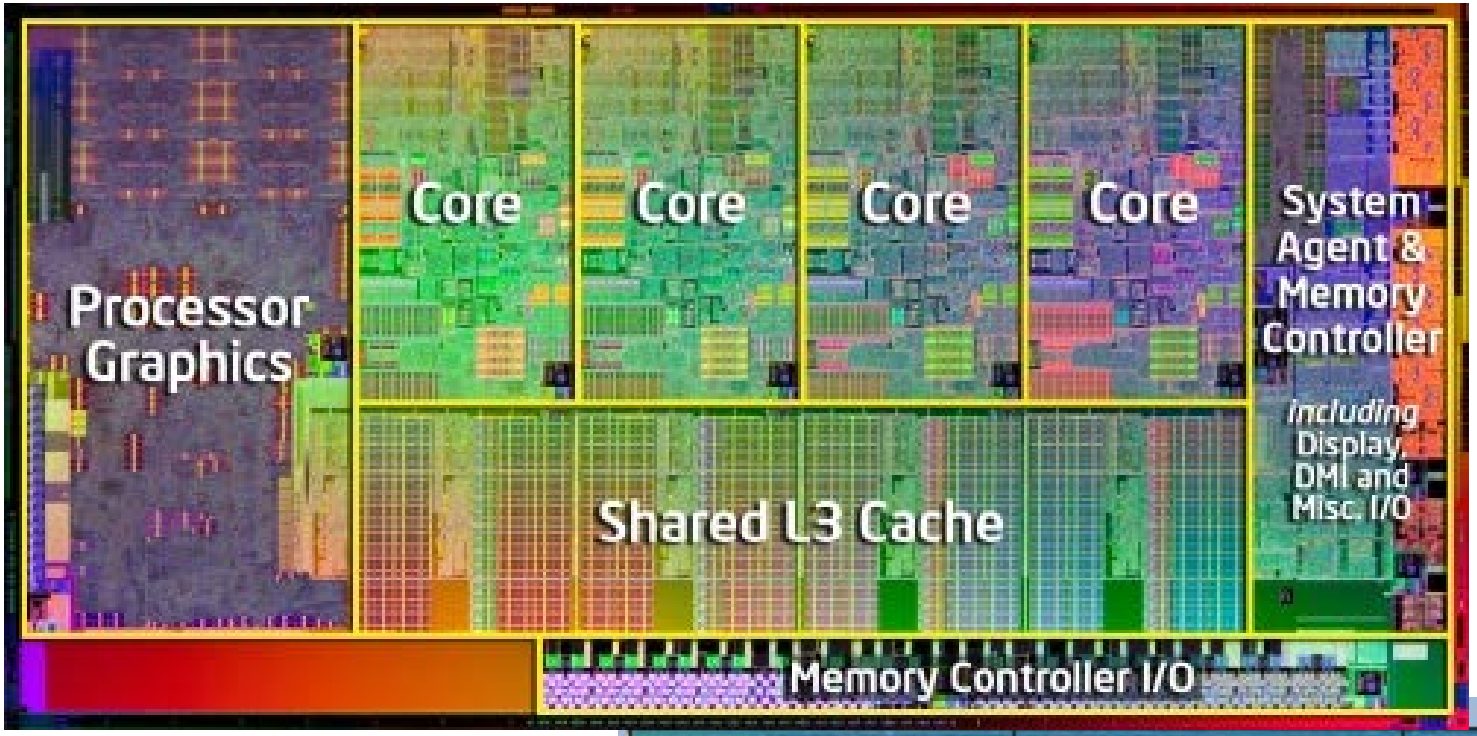
Ιεραρχία ☺


Cost/bit Size
Speed Useful-Data
CPU + Registers
Cache
Main Memory
Secondary Memory
Tape, CD-ROM etc

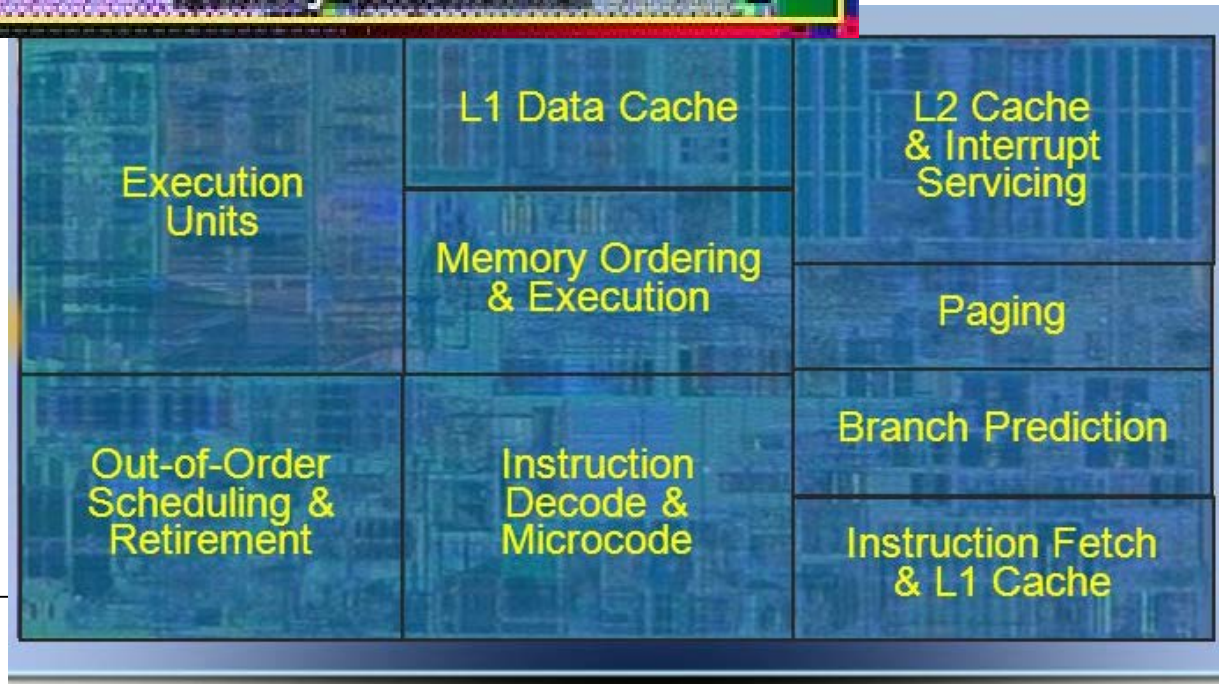
Οργάνωση Συστήματος Μνήμης

- Ένα υπολογιστικό σύστημα συνήθως διαθέτει πολλούς τύπους μνήμης
  - Registers, buffers, caches, main memory, secondary memory (flash/SSD, disk, tape)





Μέσα σε ένα επεξεργαστή:
Registers
L1 Data
L1 instructions
L2
L3 shared
…



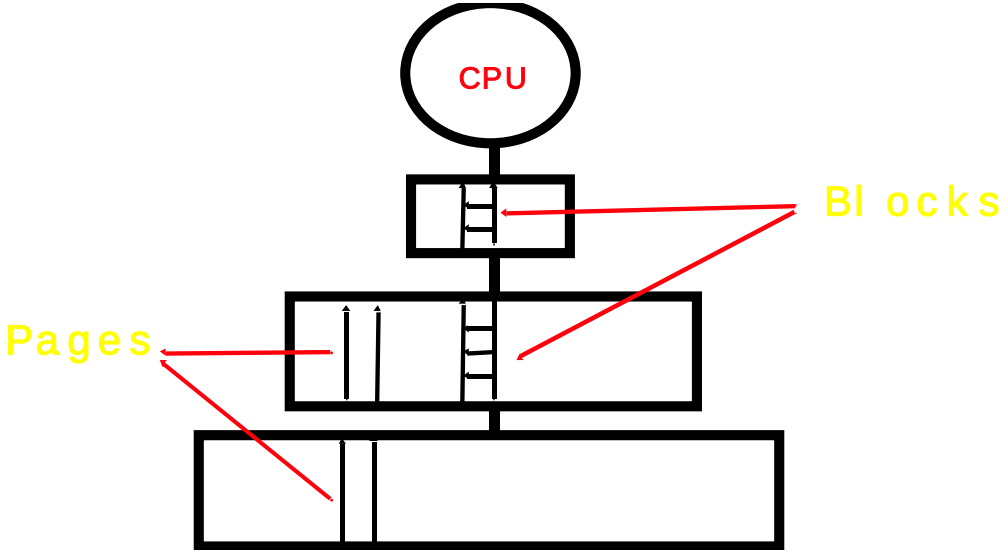
Σύστημα μνήμης (memory system)

- συσκευές μνήμης,
- αλγόριθμοι διαχείρισης και ελέγχου των αποθηκευμένων πληροφοριών.

Το σύστημα μνήμης:

- η μεγιστοποίηση της μέσης ταχύτητας μεταφοράς πληροφοριών από/προς τη μνήμη, με το ελάχιστο δυνατό κόστος,
- η αυτοματοποίηση των διαδικασιών μεταφοράς πληροφοριών μεταξύ των διαφόρων μονάδων μνήμης, έτσι ώστε να απλοποιείται το έργο των προγραμματιστών (χρηστών) του υπολογιστή, και
- η παροχή μηχανισμών για την προστασία των αποθηκευμένων πληροφοριών από ανεπίτρεπτες ενέργειες καθώς και σφάλματα των προγραμματιστών.

Η Ιεραρχία μνήμης έχει πολλά επίπεδα αλλά η διαχείρισης γίνεται μεταξύ δυο επιπέδων.



• **Επιτυχία (Hit)** είναι μια πρόσβαση στη μνήμη που ικανοποιείται από το ψηλότερο επίπεδο της Ιεραρχίας

• **Ποσοστό Επιτυχίας** Cache (**Hit Rate**) = Cache Hits/Cache Accesses

•Miss Rate = 1- Hit Rate

• **Χρόνος Επιτυχίας (Hit time)**: Ο χρόνος προσπέλασης στο ψηλότερο επίπεδο της Ιεραρχίας συμπεριλαμβανομένου και του χρόνου που χρειάζεται για τον έλεγχο εάν έχουμε επιτυχία ή αποτυχία.

• **Κόστος Αποτυχίας (Miss Penalty)**: Χρόνος για να αντικαταστήσουμε ένα μπλοκ στο ψηλότερο επίπεδο της Ιεραρχίας με το αντίστοιχο μπλοκ από το χαμηλότερο επίπεδο της Ιεραρχίας και να μεταφέρουμε το μπλοκ στην ΚΜΕ.

- **Μέσος Χρόνος Πρόσβασης Μνήμης**

  (Average memory access time)

$$= \text{Hit time} + \text{Miss ratio} \times \text{Miss Penalty}$$

$$= \text{Hit time} + (1 - \text{Hit ratio}) \times \text{Miss Penalty}$$

Total time

• **Χρόνος Πρόσβασης (Access time)**: Χρόνος μεταφοράς της πρώτης λέξης/τμήματος μπλοκ.

• **Χρόνος Μεταφοράς (Transfer time)**: Χρόνος μεταφοράς του υπόλοιπου μπλοκ.

•Άλλο Μετρικό: Misses/1000 instructions (MPKI, ?PKI)

- •Miss rate παραπλανητικό όταν δεν υπάρχει μεγάλος αριθμός προσβάσεων

  Μια εκτέλεση ενός προγράμματος εκτελεί 10 δισ. εντολές και προκαλεί 10.3 εκ. Misses στην LLC, πόσα είναι τα MPKI της LLC= $(10.3\ 10^6 / 10\ 10^9) \times 1000 = 1.3$ MPKI

•Improved memory bandwidth which time it improves (access or transfer)?

Επιπτώσεις Ιεραρχίας Μνήμης στο σχεδιασμό της ΚΜΕ

-Κόστος Αποτυχίας (Miss Penalty)

- Της τάξης τον 10δων κύκλων
  > Η ΚΜΕ αδρανεί.
- Της τάξης των εκατοντάδων χιλιάδων-εκατομμυρίων κύκλων.
  > Η ΚΜΕ διακόπτεται και εξυπηρετεί κάποια άλλη διεργασία

Η ΚΜΕ πρέπει να μπορεί να χειρίζεται αναφορές μνήμης με μεταβαλλόμενο χρόνο πρόσβασης. Τι σημαίνει αυτό;

1. Η ΚΜΕ πρέπει να έχει ένα μηχανισμό για να μπορεί να ελέγχει εάν είναι επιτυχία.
2. Ο επεξεργαστής πρέπει να έχει ένα μηχανισμό για τη μεταφορά μπλοκ ανάμεσα στα διάφορα επίπεδα μνήμης
3. Context Switch

Κρυφή Μνήμη (CACHES $)

• Το Cache είναι το επίπεδο(α) της ιεραρχίας που βρίσκεται μεταξύ της ΚΜΕ και της κυρίως μνήμης.

•Μοντέρνοι επεξεργαστές πολλαπλά επίπεδα caches

•Ενα cache περιέχει ένα αριθμό blocks (πχ 512)

•Κάθε block έχει μέγεθος (πχ 64B - τοπικότητα)

•Για ένα cache η μνήμη είναι μοιρασμένη σε block,πχ 2^{32} B memory

•Κάθε διεύθυνση ανήκει σε ένα blockμε μοναδικό block address

- **Cache 64B block size => memory 2^{26} blocks, Block address upper 26 bits, offset six bits**

- **Cache 128B/block,….**

Λειτουργία Ιεραρχίας Μνήμης

- Q1: Τοποθέτηση των Μπλοκ (Block placement): Που τοποθετείται ένα μπλοκ στο ψηλότερο επίπεδο της Ιεραρχίας?

- Q2: Αναγνώριση των Μπλοκ (Block Identification) Μέθοδος ελέγχου κατά πόσο ένα μπλοκ βρίσκεται στο ψηλότερο επίπεδο της Ιεραρχίας?

- Q3: Αντικατάσταση των Μπλοκ (Block replacement): Ποιο μπλοκ επιλέγεται για αντικατάσταση σε αποτυχία?

- Q4: Πολιτική Εγγραφής (Write Strategy): Πως γίνονται οι εγγραφές?

- **Q1: Που τοποθετείται ένα μπλοκ στο cache ($)**
  (Where can a block be placed in a cache)

Μια $ έχει ***n*** υποδοχές για blocks

κάθε block έχει συγκεκριμένο μέγεθος

Κάθε διεύθυνση ανήκει σε ένα block

Τα δεδομένα μιας εντολή μνήμης που βρίσκονται στο $

lw r4,4(r5)

r4 = 0x10004000

Address = 0x10004004 βρίσκεται μέσα στην cache;

Το τι θα ψάξουμε είναι για το block που περιέχει την διεύθυνση
Σε ποια υποδοχή είναι ενα block μέσα στο $;

Μνήμη άμεσης Χαρτογράφησης (Direct Mapping)

Κάθε μπλοκ μπορεί να τοποθετηθεί μονό σε μια υποδοχή στο cache

- Αριθμός μπλοκ MOD αριθμός των υποδοχών.

(Block-frame address) modulo (Number of blocks in cache)

- Εξαλείφει το πρόβλημα της αναζήτησης.

- Πολλά μπλοκ αντιστοιχούν στην ιδία υποδοχή (conflicts)

- Δημιουργείται πρόβλημα εάν πολλά blocks που χρησιμοποιούνται τυγχάνει να αντιστοιχούν στην ίδια υποδοχή.

Για δυαδικούς αριθμούς b_i={0,1},

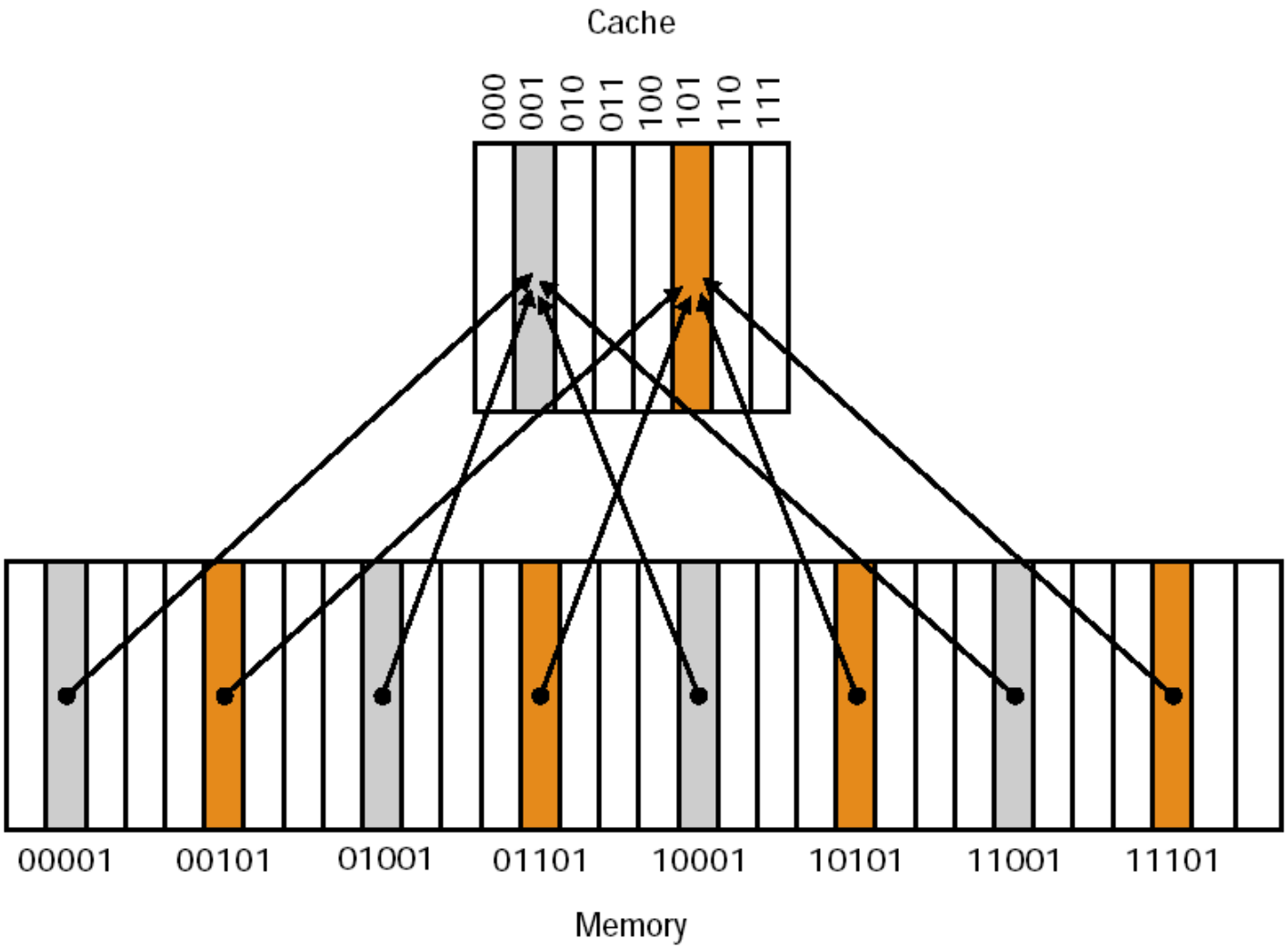
$b_{n+m-1}\ldots b_{n+2}b_{n+1}b_nb_{n-1}\ldots b_2b_1b_0$ MOD 2^n= $b_{n-1}\ldots b_2b_1b_0$

n least significant bits

Πχ 10010100101 MOD 16 ($16=2^4$) =

Στην C/C++/JAVA: x mod 2^n= x & (2^n – 1)

Direct Mapping



Συσχετιστική Μνήμη (Fully Associative):

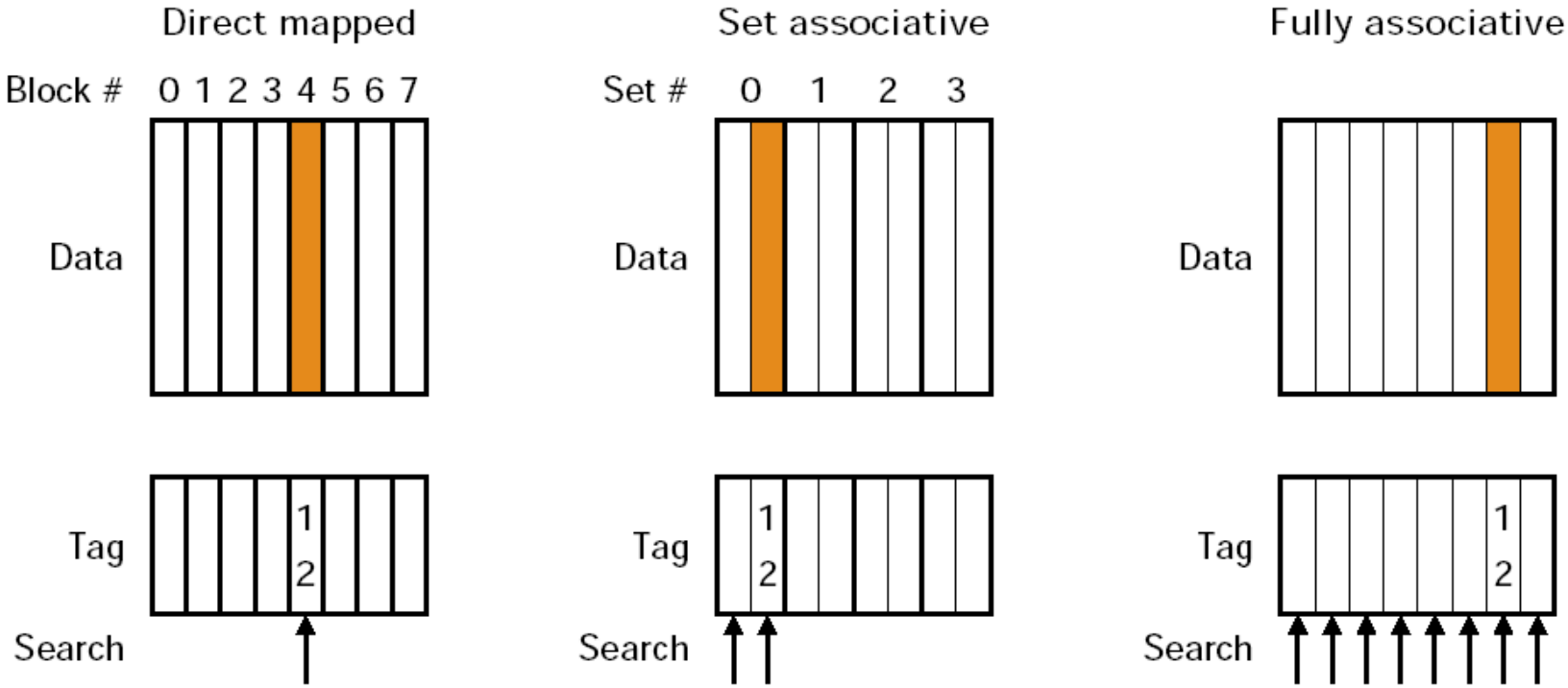
- Κάθε μπλοκ μπορεί να τοποθετηθεί σε οποιαδήποτε υποδοχή.

Συσχετιστική Μνήμη συνόλου με n καταχωρήσεις ανά υποδοχή (n-way set associative):

Ένα μπλοκ πρώτα χαρτογραφείται σε ένα σύνολο από υποδοχές και μετά τοποθετείται σε οποιαδήποτε υποδοχή μέσα στο σύνολο.

Αριθμός μπλοκ MOD αριθμός των Συνόλων.

(Block-frame address) modulo (Number of SETS in cache)



Η τοποθέτηση του μπλοκ με διεύθυνση 12 σε cache με 8 μπλοκς

DM, # Sets = 8x1 12 % 8 = 4

2-way # Sets= 4x2 12 % 4 = 0

FA # Sets = 1x8 12 % 1 = 0

Q2: Πως ελέγχουμε κατά πόσο ένα μπλοκ βρίσκεται στο ψηλότερο επίπεδο της Ιεραρχίας.

Για direct cache και set-associative η διεύθυνση χωρίζετε σε 3 τμήματα.

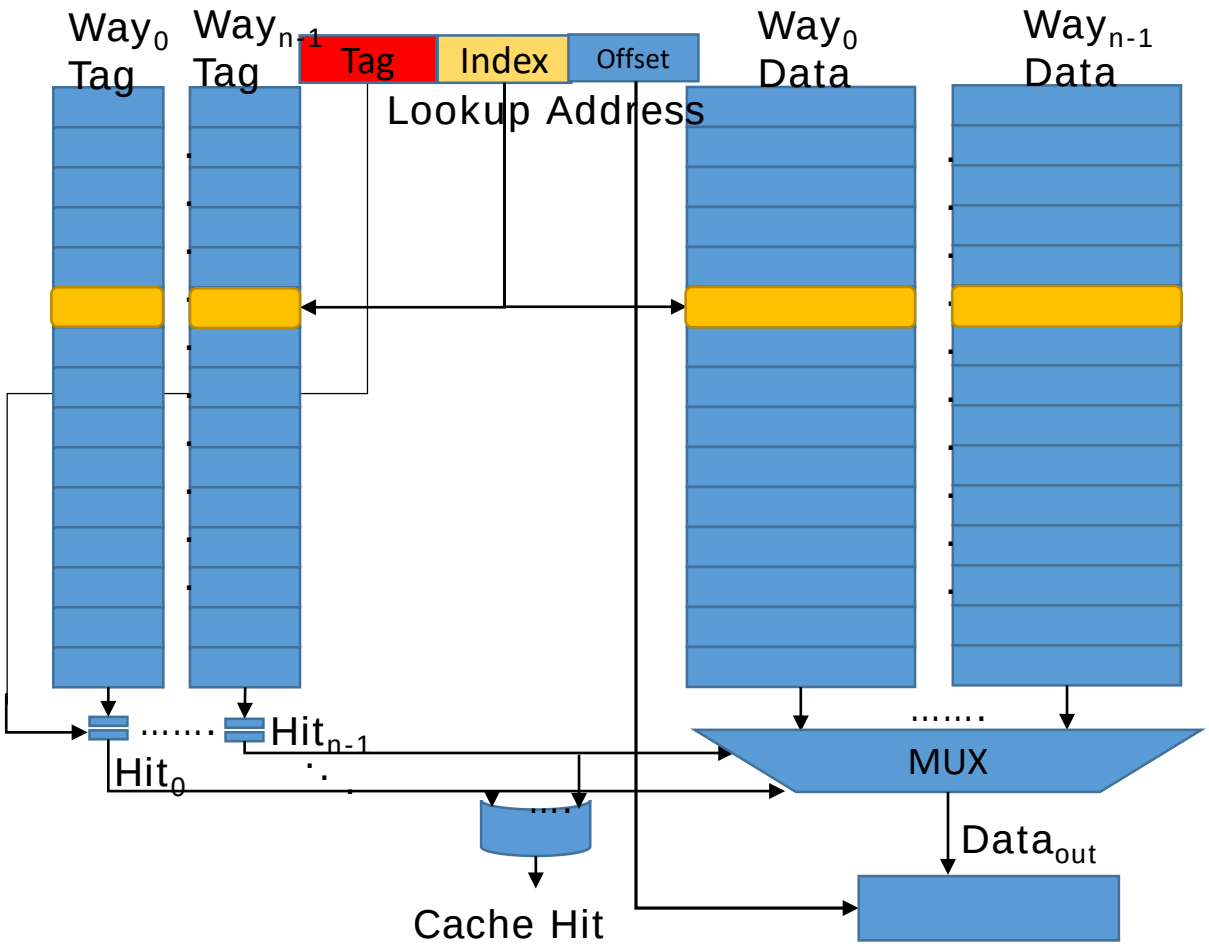
Tag (ετικέτα)	Index (δείκτης)	Block Offset

Αποθηκεύεται το tag του κάθε block μέσα στην $

Cache usually consists:

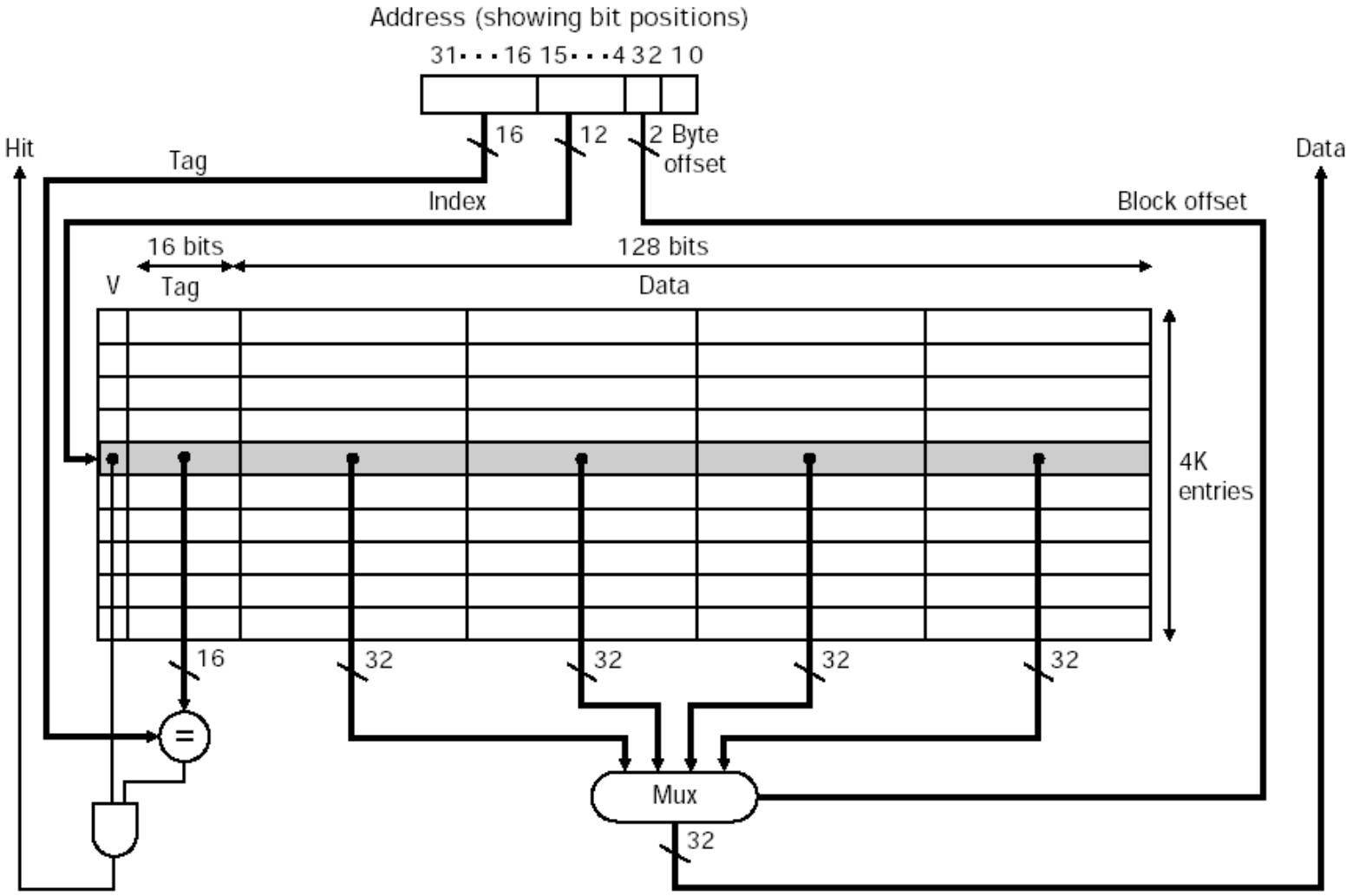
1. Data Array (each entry holding a block of data)
2. Tag Array (as many entries as data array, holding for each data block its tag)

Cache


Way0 Tag
Wayn-1 Tag
Tag
Index
Offset
Lookup Address
Way0 Data
Wayn-1 Data
Hit0
Hitn-1
MUX
Dataout
Cache Hit

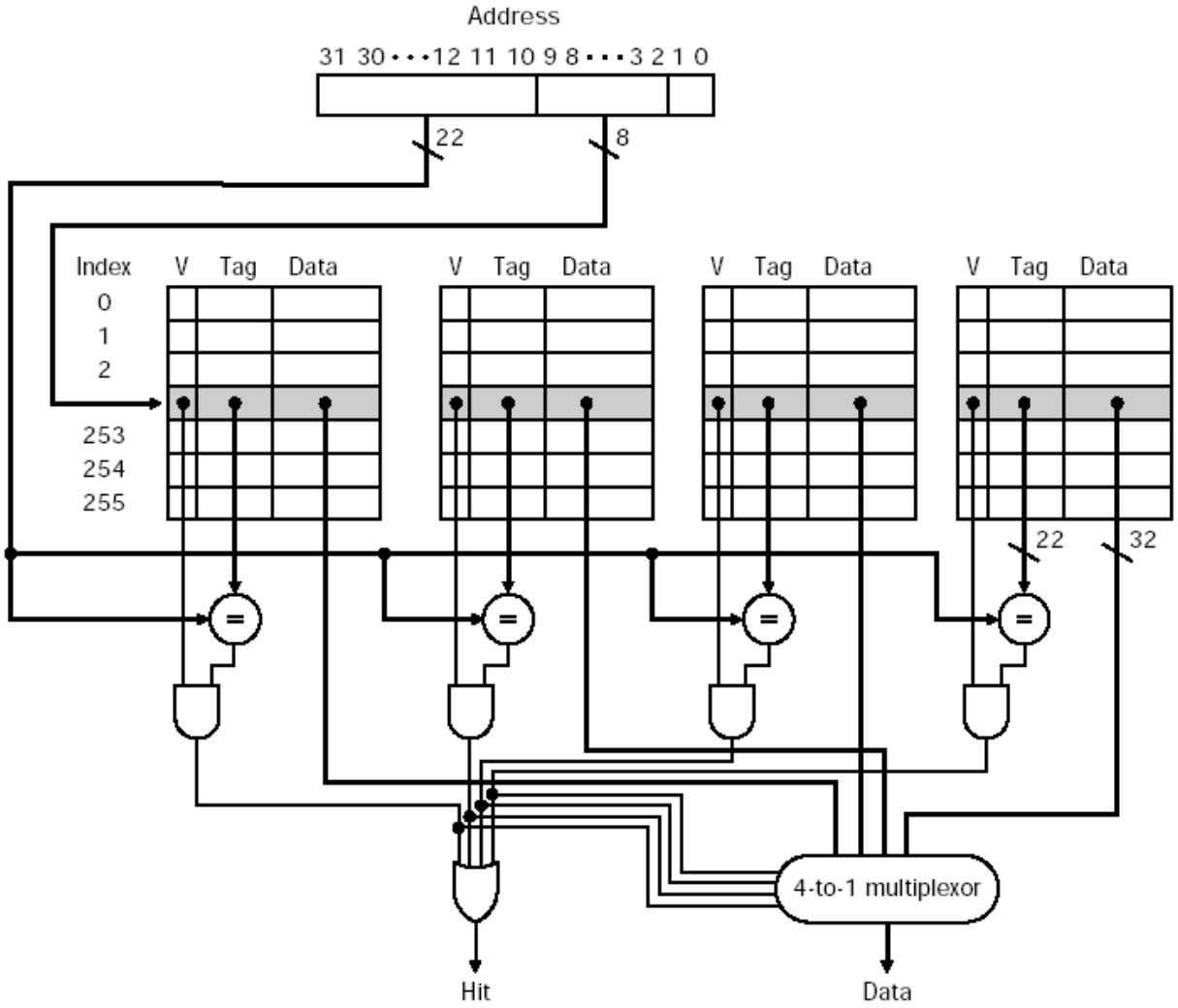
- **direct mapped cache with 4 word (16 bytes) blocks**
- **4096 υποδοχές**
- **32-bit address**

- **4096 = $2^{?}$**
- **Τα δεδομένα = ?KB**
- **Address bits for offset:**
- **Address bits for index:**
- **Address bit for tag:**



direct mapping cache with 4 word (16 bytes) blocks

Τα δεδομένα 2^{12} x 2^4 B = 64KB

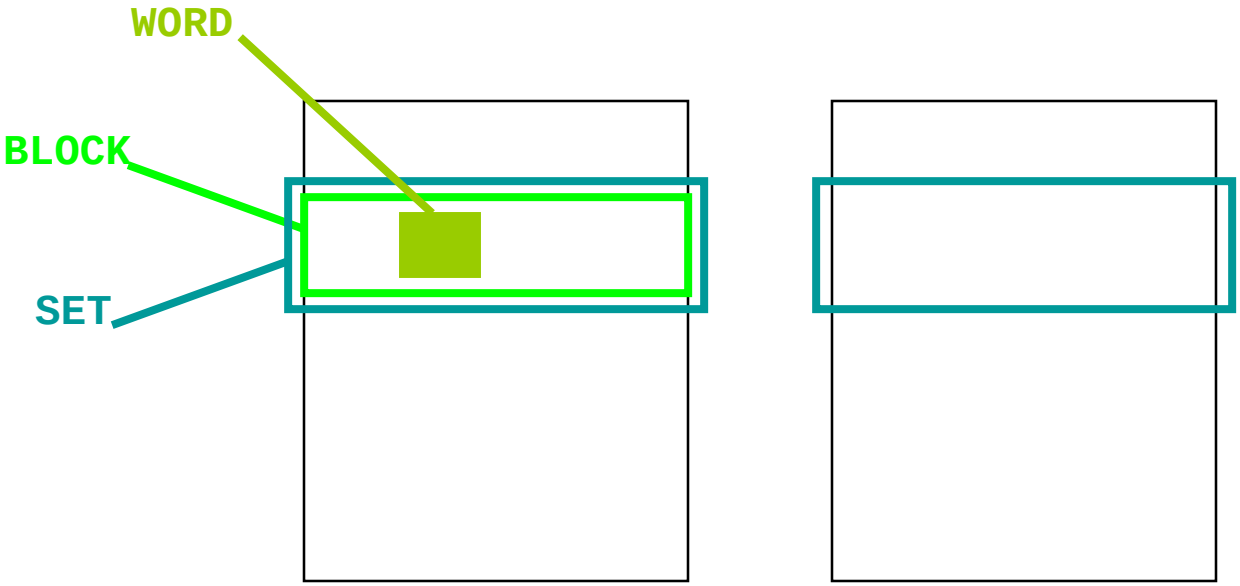


4-way set associative cache

4 comparators and a 4x1 Mux

Τα δεδομένα 2^8 x 2^2 x 2^2 B = 4KB


Tag
Index
Offset


WORD
BLOCK
SET

Q3: Ποιο μπλοκ αντικαθιστάται στην περίπτωση αποτυχίας.

- **Μνήμη άμεσης Χαρτογράφησης (Direct mapping):** Καμιά επιλογή.

- **Συσχετιστική Μνήμη και Συσχετιστική Μνήμη συνόλου με Ν καταχωρήσεις** (Fully associative and set associative):

  - Τυχαία ή ψευδοτυχαία. Random (or pseudorandom)

  - Least-Recently-Used (LRU): The block replaced is the one that has been unused for the longest time.

  - FIFO (First-In-First-Out)

4-way set

Block Addresses: ABABBCDEDABC

LRU:

Time

4-way set

Block Addresses: ABABBCDEDABC

LRU: - - - - - - - - A- BCE

Q4: Ποια είναι η πολιτική εγγραφής (What happens on write?)

• Οι αναγνώσεις είναι η μεγάλη πλειοψηφία των προσβάσεων στη μνήμη (20% από όλες τις αναφορές μνήμης είναι εγγραφές)

Parallel Access to Tag and Data Array

Η ανάγνωση ενός μπλοκ μπορεί να αρχίσει παράλληλα με την αναζήτηση της κρυφής μνήμης.

• Η ανάγνωση ενός μπλοκ μπορεί να αρχίσει μόλις γίνει γνωστή η διεύθυνση της υποδοχής του μπλοκ.

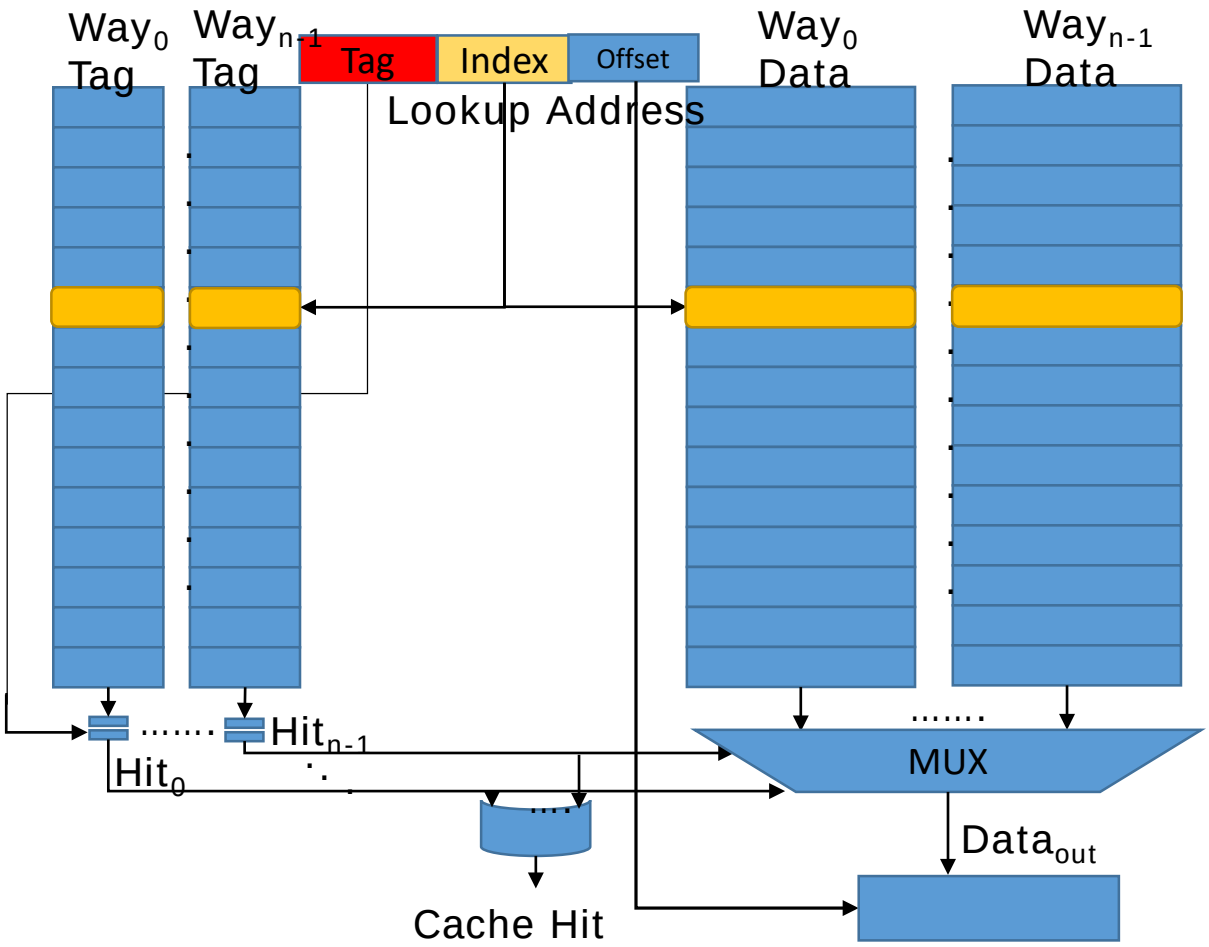
• Στην περίπτωση Επιτυχίας η λέξη διαβιβάζεται στην ΚΜΕ

•Στην περίπτωση αποτυχίας δεν υπάρχει ούτε πλεονέκτημα ούτε μειονέκτημα. Κοστίζει σε τι;

•Parallel Access εφαρμόζεται στα ψηλά επίπεδα ιεραρχίας.

•Στα χαμηλά πρώτα εξετάζουμε το tag array και εάν υπάρχει HIT τότε διαβάζουμε από το data array Serially Accessed Caches 1st Tag and then Data Array

Cache


Way0 Tag
Wayn-1 Tag
Tag
Index
Offset
Lookup Address
Way0 Data
Wayn-1 Data
Hit0
Hitn-1
MUX
Dataout
Cache Hit

•Οι εγγραφές δεν ξεκινούν μέχρι να ελεγχθεί η ετικέτα για επιτυχία.

- Οι εγγραφές παίρνουν περισσότερο χρόνο από τις αναγνώσεις

Υπάρχουν δυο επιλογές για τις εγγραφές

- Υστεροεγραφή (Write Back (WB))

- Διεγγράφη (Write through (WT))

Υστεροεγραφή (Write Back)

- Η μνήμη δεν ενημερώνεται όποτε αλλάζει το cache.
  - Η μνήμη ενημερώνεται μόνο όταν η καταχώρηση αφαιρείται από το cache για να επιτρέψει σε κάποια άλλη καταχώριση να καταλάβει την υποδοχή της.

- Απαιτείται ένα bit (dirty-bit) σε κάθε μπλοκ που να ενημερώνει αν έχει αλλάξει η καταχώρηση του cache αφού φορτώθηκε.
  - Διαφορετικό από το valid-bit

- Οι έγγραφες γίνονται με την ταχύτητα του cache.

- Οι καταχωρήσεις στο cache και στη Μνήμη δεν είναι συναφείς.
  - η τιμή στο cache διαφορετική από ότι στην μνήμη – που βρίσκεται πάντοτε η πιο πρόσφατη τιμή;

Διεγγραφή (Write Through):

• Όταν μια λέξη γράφεται στο cache, γράφεται αυτόματα και στη μνήμη.

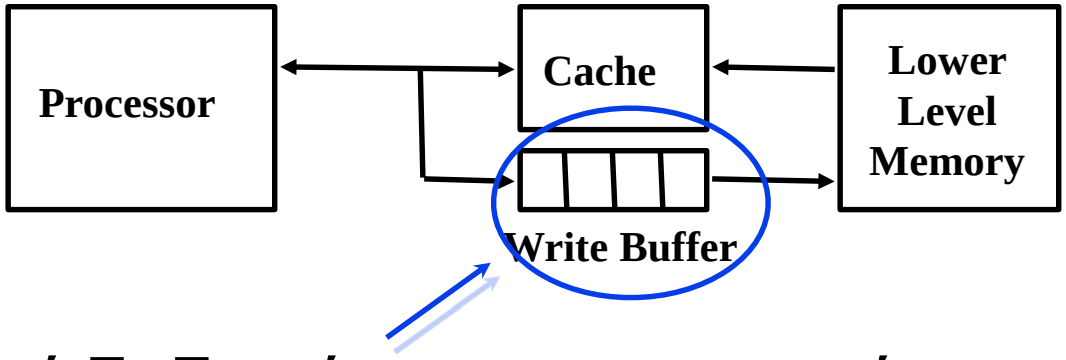
• Οι καταχωρήσεις στο cache και στη Μνήμη είναι συναφείς.

• Οι έγγραφες γίνονται με την ταχύτητα της μνήμης.

- Η χρήση **write-buffer** επιτρέπει στην ΚΜΕ να συνεχίσει την επεξεργασία κατά τη διάρκεια της εγγραφής στη μνήμη.

- Η διεγγραφή προφανώς προκαλεί μεγαλύτερη κυκλοφορία στο **δίαυλο** από την υστεροεγγραφή.
- Η υστεροεγγραφή (writeback) έχει το πρόβλημα της Συνάφειας cache και Μνήμης.
  - You cant read data from memory without checking if data is in the cache
  - Who reads the data from memory without first checking the cache?

Write Buffers για Write-Through Caches



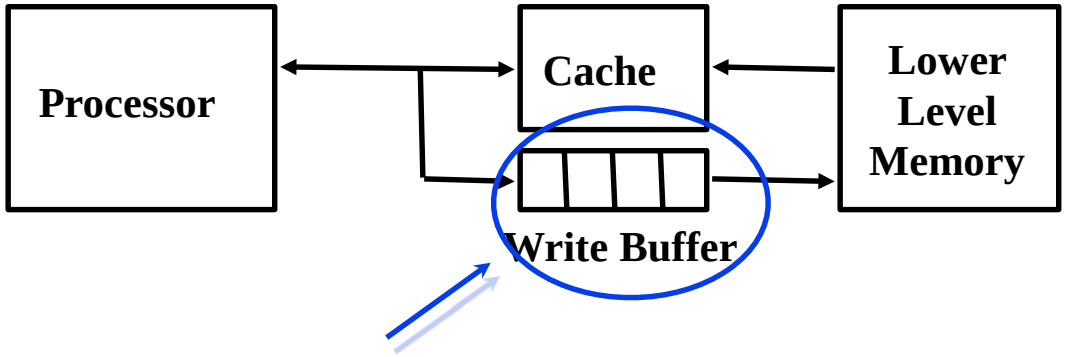
Διατηρεί δεδομένα σε αναμονή για εγγραφή στο χαμηλότερο επίπεδο μνήμης

Q. Why a write buffer ?

Q. Why a buffer, why not just one register ?

Q. Are Read After Write (RAW) hazards an issue for write buffer?

Write Buffers for Write-Through Caches



Q. Why a write buffer ? **A. Ο επεξεργαστής δεν περιμένει**

Q. Why a buffer, why not just one register ? **A. Μαζεμένες εγγραφές**

Q. Are Read After Write (RAW) hazards an issue for write buffer? **A. Ναι! Άδειασε το buffer ή έλεγξε το περιεχόμενο του (τι περιέχει WB)**

Αποτυχία σε Εγγραφές (Write Miss)

• Υπάρχουν δυο Επιλογές

- Write Allocate (Fetch on write)
- No write allocate (write around)

• Και οι δυο στρατηγικές μπορούν να χρησιμοποιηθούν και με τις δυο στρατηγικές εγγραφής.

• Η πρακτική είναι:

- η Υστεροεγγραφή (writeback) να συνδυάσετε με write-allocate και
- η διεγγραφή (writethrough) με no-write-allocate.

Απόδοση του Cache (Cache Performance)

Παράδειγμα 1:

Miss penalty = 6 clock cycles

CPI = 9.0

Miss rate = 10%

3.0 references/instruction

Ποσό % του CPI προκαλείτε από τα misses;

Παράδειγμα 2

Miss Penalty = 10 cycles

CPI = 2.5

Miss rate 10 %

1.5 references/instruction

Ποσό % του CPI προκαλείτε από τα misses;

Απόδοση της Κρυφής Μνήμης

CPU execution time = (CPU clock cycles + Memory stall cycles) x Clock cycle time
= (CPI_{cpu} + CPI_{misses})x I x CT

Memory Stall Cycles = Number of misses x Miss penalty
= I x (Misses / Instruction) x Miss penalty
= I x (Memory Accesses / Instruction) x Miss rate x Miss penalty

Memory stall Cycles = I x Reads per instruction x Read miss rate x Read miss penalty
+ I x Writes per instruction x Write miss rate x Write miss penalty

(Misses / Instruction) = (Miss rate x Memory accesses) / Instruction count
= Miss rate x (Memory accesses / Instruction)

Απόδοση της Κρυφής Μνήμης

CPU execution time = (CPU clock cycles + Memory stall cycles) x Clock cycle time
= (CPI_{cpu} + CPI_{misses})x I x CT

Memory Stall Cycles = Number of misses x Miss penalty
= I x (Misses / Instruction) x Miss penalty
= I x (Memory Accesses / Instruction) x Miss rate x Miss penalty

Memory stall Cycles = I x Reads per instruction x Read miss rate x Read miss penalty
+ I x Writes per instruction x Write miss rate x Write miss penalty

(Misses / Instruction) = (Miss rate x Memory accesses) / Instruction count
= Miss rate x (Memory accesses / Instruction)

Παράδειγμα:

- Ideal CPI=1, load/store=50% instr, miss penalty=25clk, miss rate=2% (ίδιο για εντολές και δεδομένα)
- speedup=?

Πήγες Αποτυχίας Cache

• Υποχρεωτικές (Compulsory): Πρώτη αναφορά σε ένα μπλοκ (το οποίο ΔΕΝ ήταν ποτέ στο cache).

• Χωρητικότητας (Capacity): Cache misses που προκαλούνται λόγω του ότι το Cache ΔΕΝ δύναται να περιέχει όλα τα blocks που χρειάζονται για την εκτέλεση ενός προγράμματος.

• Συγκρούσεις (Collision): Cache misses που προκαλούνται λόγω του ότι σε set associative ή direct-mapped caches πολλαπλά blocks συναγωνίζονται για το ίδιο set.

Αυξάνοντας τη χωρητικότητα του cache μειώνονται οι αποτυχίες λόγω συγκρούσεων και οι αποτυχίες λόγω χωρητικότητας.

Μέθοδοι Βελτίωσης Επίδοσης CACHE

1. Αυξάνοντας το μέγεθος του συνόλου της Συσχετιστικής Μνήμης Συνόλου μειώνει τις αποτυχίες από Συγκρούσεις.

- Η πλήρως Συσχετιστική Μνήμη εξαλείφει της αποτυχίες από Συγκρούσεις.

- Δαπανηρή η υλοποίηση τους σε υλικό

- Μπορεί να αυξήσει τον χρόνο πρόσβασης και να μειώσει την απόδοση του συστήματος.

2. Μεγαλώνοντας τα Μπλόκς μειώνουμε τις υποχρεωτικές αποτυχίες.

- Είναι πιθανό να αυξήσει αποτυχίες από Συγκρούσεις.

3. Ξεχωριστό Cache Εντολών, Cache Δεδομένων (I$ και D$)

Ξεχωριστά Cache χρησιμοποιούν διαφορετικές θύρες για εντολές και δεδομένα.

- Διπλασιάζετε το εύρος.
- Βελτιστοποιούμε κάθε cache ξεχωριστά:
- Διαφορετικές χωρητικότητες, μέγεθος μπλοκ και βαθμός συσχετισμού.
- Εξαλείφουν τις Συγκρούσεις μεταξύ μπλοκ δεδομένων και μπλοκ εντολών

• Cache-Εντολών έχουν μικρότερο ποσοστό αποτυχίας από τα Cache- δεδομένων.

Prefetching Lines

Prefetch more blocks on a miss (avoid cold misses)

But prefetching can be off (too early or too late)

Flag prefetched lines as prefetched when inserted

If line has hit reset its prefetch bit

On replacement

Line is marked prefetched prioritize replacement

If (lines replaced with prefetched bit set / prefetched lines > **Threshold**)

block prefetching for **X** Cycles

Υποθέστε σύστημα με IL1, DL1, L2, Μνήμη ίδιο block size

lw $6,0($8)

Πόσες προσβάσεις στην μνήμη χρειάζεται η πιο πάνω εντολή;

Υποθέστε σύστημα με IL1, DL1, L2, Μνήμη ίδιο block size

lw $6,0($8)

Access στο IL1 με το PC της εντολής
Access στο DL1 με την διεύθυνση στο $8+0

Access στο L1

Hit στο L1 διαβάζει μία λέξη

Miss στο L1 Access στο L2

Hit στο L2 διαβάζει ένα block

εάν L1 line replaced dirty: write block to L2

Miss στο L2 Access στο Main Memory

Διάβασε ένα block από την Μνήμη

εάν L2 line replaced dirty: write a block to memory

[Miss στην μνήμη (page fault)]

Κύρια Μνήμη (Main Memory)

Η μνήμη που είναι απευθείας προσπελάσιμη από την ΚΜΕ ονομάζεται **κύρια μνήμη** και χρησιμοποιείται για την αποθήκευση των προς εκτέλεση προγραμμάτων και των δεδομένων

Μέτρα Απόδοσης (Performance Measures)

- Χρόνος Προσπέλασης (Access Latency)
- Εύρος (Bandwidth)

Χρόνος προσπέλασης (access time) Ο χρόνος που απαιτείται για την ανάκτηση ενός συγκεκριμένου τμήματος δεδομένων από μια θέση αποθήκευσης.

- Ισοδυναμεί με το χρόνο που μεσολαβεί από τότε που καλούνται τα δεδομένα από τη μνήμη μέχρι τότε που είναι έτοιμα για χρήση.

Εύρος (Bandwidth): Bytes/second, ρυθμός μεταφοράς δεδομένων

DRAM ως Κυρία Μνήμη

•Χρόνος κύκλου μνήμης (Memory Cycle Time): Ελάχιστος χρόνος μεταξύ δυο διαδοχικών προσβάσεων στην μνήμη.

• Δυναμική RAM (DRAMs) έχει line multiplexing

  •Ίδια σήματα διαφορετική σημασία στον χρόνο

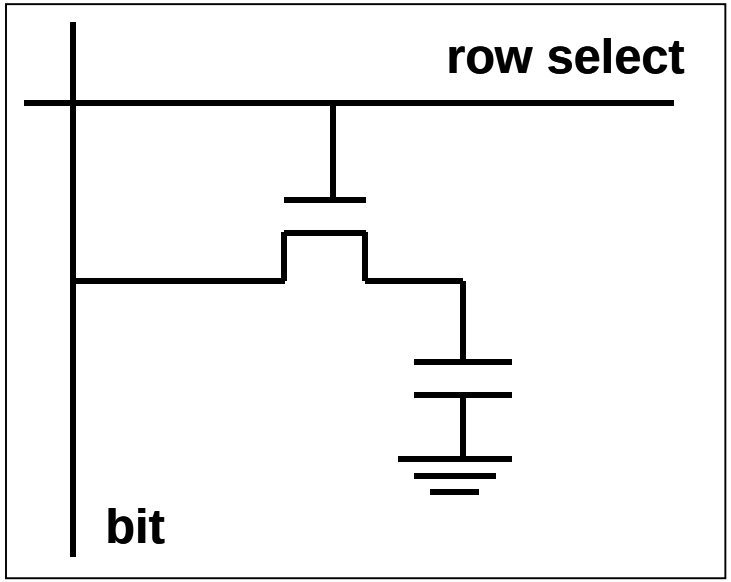
•RAS (Row-Access strobe) & CAS (Column-Access strobe)

  •RAS-CAS μοιράζονται τα ίδια σήματα σε διαφορετικό χρόνο

• Δυναμική RAM χρειάζεται περιοδικό φρεσκάρισμα (refresh)

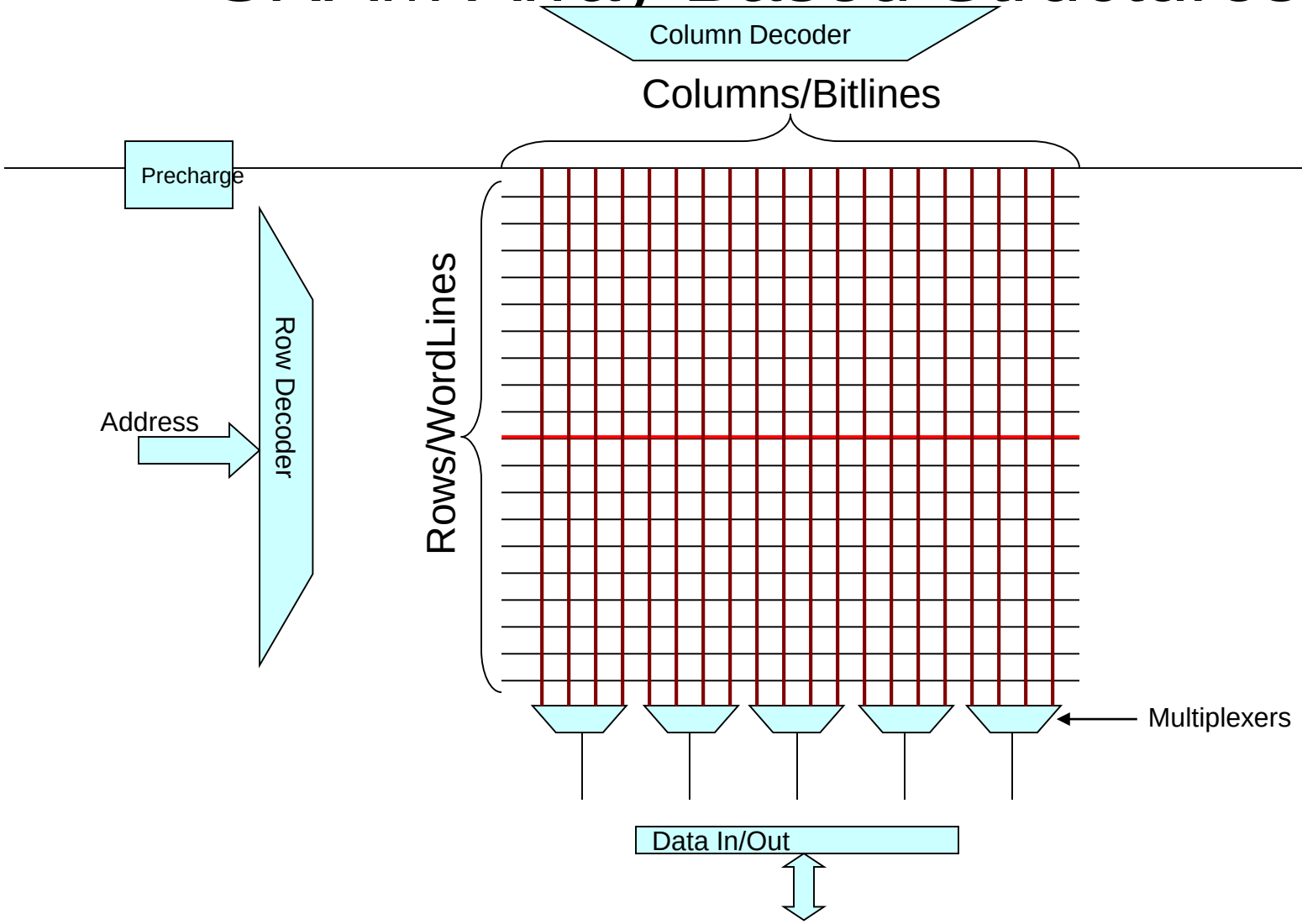
• Το κόστος του φρεσκαρίσματος είναι συνήθως το κόστος μιας προσβάσεως στην μνήμη (RAS and CAS)

• Όταν χρησιμοποιούμε DRAMs τα δεδομένα πρέπει να ξαναγράφονται μετά από κάθε ανάγνωση (κύκλος μνήμης)



- **Write**:
    1. Drive bit line
    2. Select row
- **Read**:
    1. Precharge bit line to Vdd
    2. Select row
    3. Cell and bit line share charges
       Very small voltage changes on the bit line
    4. Sense (sense amp)
       Can detect changes
    5. Write: restore the value
- **Refresh**:
    1. Just do a dummy read to every cell.

SRAM Array Based Structures



Στατική RAM (Static RAM (SRAM))

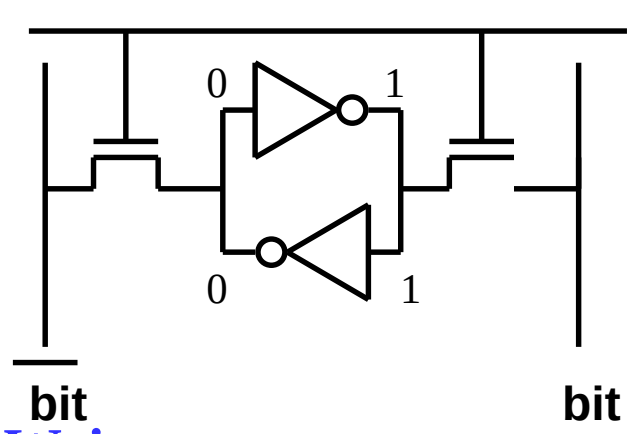
- Δεν υπάρχει address line multiplexing
- Δεν χρειάζονται φρεσκάρισμα (No need for refreshing)

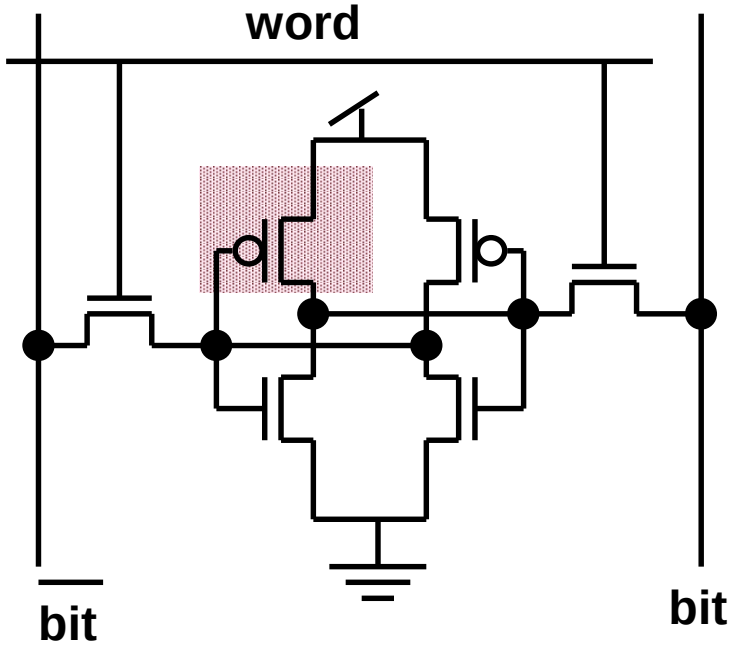
Μνήμες κατασκευασμένες με τι ίδια τεχνολογία (Memories designed with comparable technologies)

- DRAM's Capacity = 16 times the SRAM capacity
- SRAM's cycle time = 8-10 times faster tan DRAM's cycle time

- Main Memory: DRAMs
- Caches: SRAMs
- Standard Cells (latches, F/F) για άλλα arrays

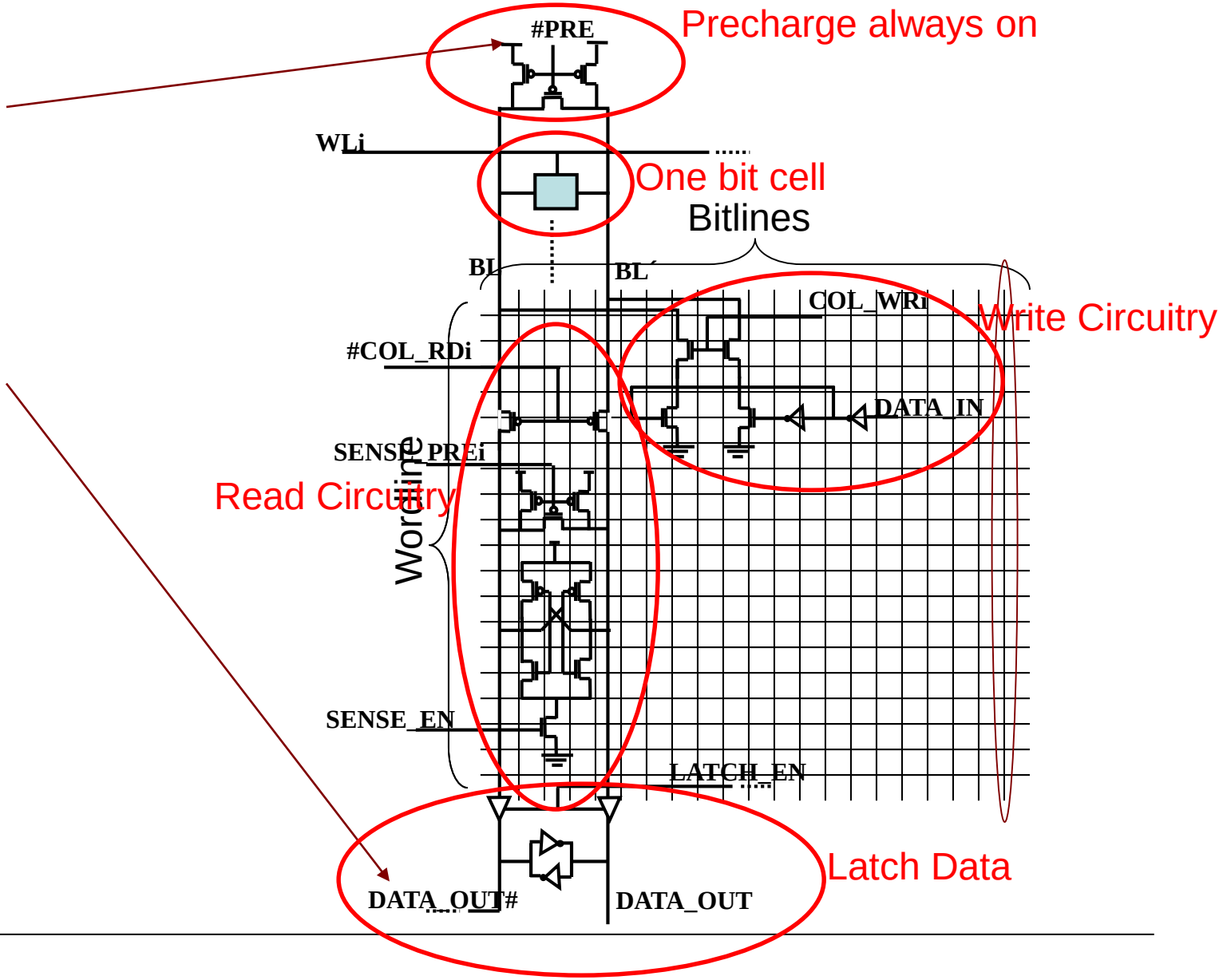
6-Transistor SRAM Cell





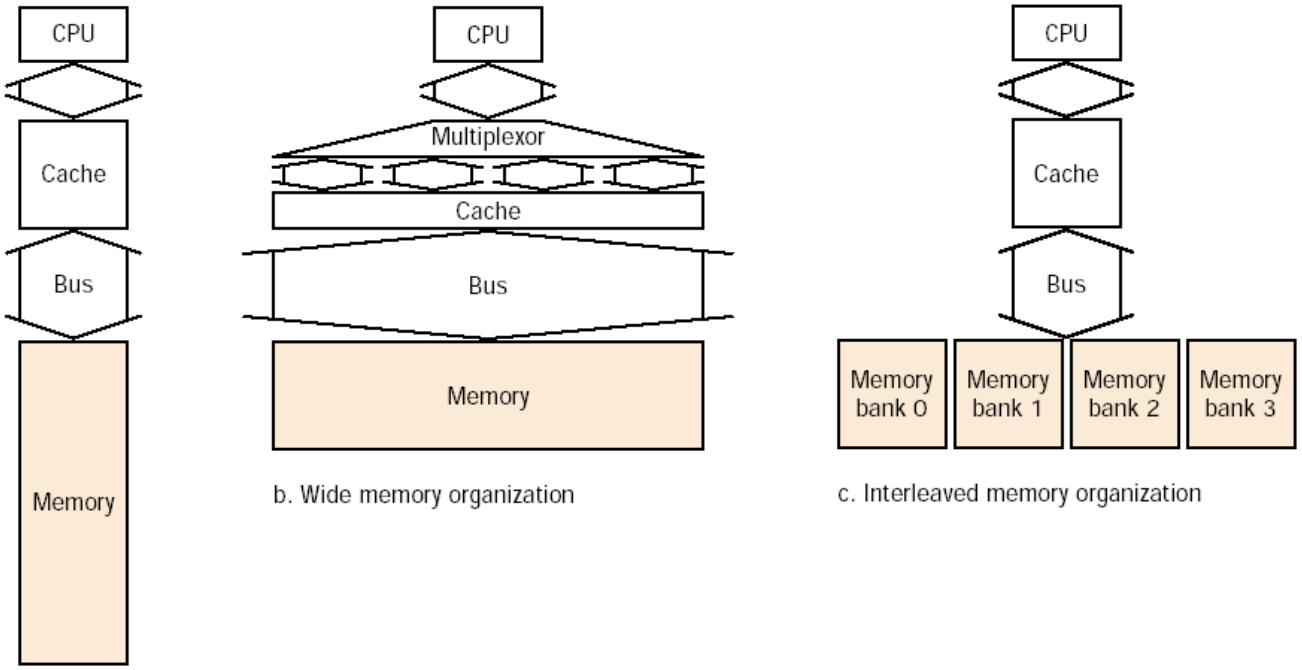
- **Write**:
  1. **Drive bit lines (bit=1, bit=0)**
  2. **Select row**
- **Read**:
  1. **Precharge bit and bit to Vdd**
  2. **Select row**
  3. **Cell pulls one line low**
  4. **Sense amp on column detects difference between bit and bit´**

Column (Bitline Pair)



Οργάνωση Μνήμης (Memory Organizations)

- Μονολεκτική Οργάνωση Μνήμης
- Πλατιά Οργάνωση Μνήμης
- Παρεμβαλλόμενη Οργάνωση Μνήμης (Interleaved memory organization)



a. One-word-wide memory organization

b. Wide memory organization

c. Interleaved memory organization

Τι χρειάζετε…

- 1 κύκλο για την μεταγωγή της διεύθυνση

  (1 clock cycle to send address)

- 6 κύκλους για την πρόσβαση ανά λέξη.

  (6 clock cycles for the access per word)

- 1 κύκλο για την μεταγωγή μια λέξης

  (1 clock cycle to send a word of data)

- Μέγεθος του Cache μπλοκ = 4 λέξεις

  (Cache Block = 4 words)

I. Μονολεκτική Οργάνωση Μνήμης (One word wide memory)

- Κόστος Αποτυχίας (Miss penalty) = 32 κύκλους (clock cycles)
- Εύρος Ζώνης (Memory bandwidth) = 1/2 byte/cycle

II. Πλατιά Οργάνωση Μνήμης (Wider main Memory)

- Πλάτος μνήμης= 2 λεξεις (Double Width):
  - Κόστος Αποτυχίας (Miss penalty) = 16 clock cycles ,
  - Εύρος Ζώνης (Bandwidth) = 1 byte/cycle
- Πλάτος μνήμης= 4 λεξεις (Quadruple width):
  - Κόστος Αποτυχίας (Miss penalty) = 8 clock cycles
  - Εύρος Ζώνης (Bandwidth) = 2 bytes/cycle

Μειονεκτήματα (Drawbacks):

- Πλατύς Δίαυλος (Wider Bus): ένα πολυπλέκτη τοποθετείτε μεταξύ του Cache και του CPU.
- Ο πολυπλέκτης μπορεί να είναι στο κρίσιμο μονόπατι

III. Παρεμβαλλόμενη Οργάνωση Μνήμης (Interleaved memory)

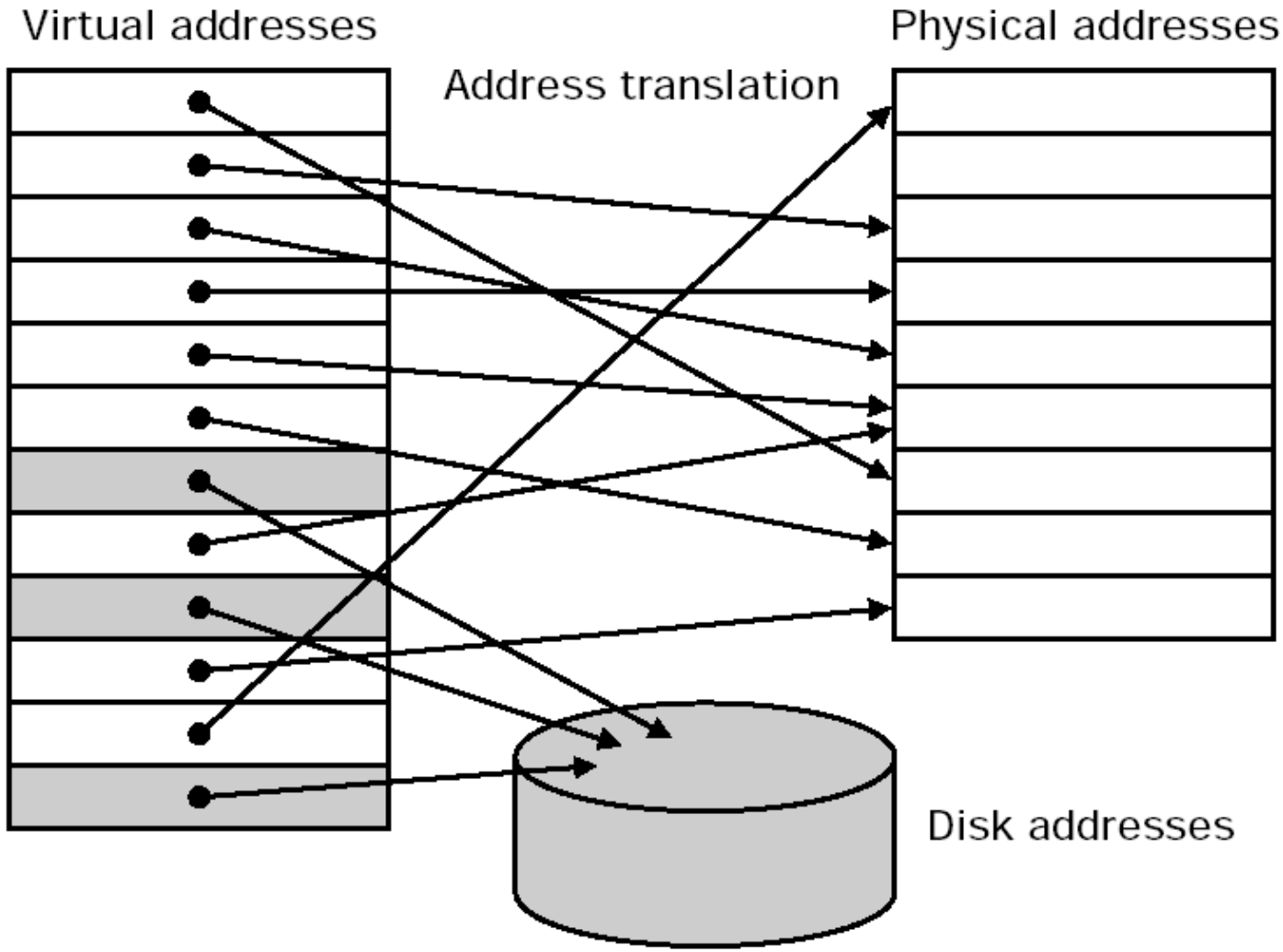
Ένα σύστημα μνήμης όπου διαδοχικές λέξεις βρίσκονται σε διαφορετικές μονάδες. Πολλαπλές ανάγνωσης ή έγγραφες μπορούν να πραγματοποιηθούν.

- Μονάδες έχουν πλάτος μια λέξη.

- 4 Μονάδες και cache μπλοκ 4 λέξεων.

- Κόστος Αποτυχίας (Miss penalty) = 1 + 6 + 4*1 = 11.
- Εύρος Ζώνης (Bandwidth) = 1.5 bytes/cycle

Ενότητα 7(β)
Εικονική Μνήμη


Virtual addresses
Physical addresses
Address translation
Disk addresses

- Virtual and Physical Memory
  - Virtual: το τι βλέπει ο προγραμματιστής
  - Physical: *το RAM του συστήματος*

- Δεν είναι απαραίτητο Virtual == Physical
  - Πιο οικονομικό/πρακτικό

- Συνήθως Virtual > Physical

- Εικονική Μνήμη είναι ένα ιεραρχικό σύστημα μνήμης με τουλάχιστον δύο επίπεδα.

- Η διαχείρισή του γίνεται από το Λ.Σ. και κάθε διεργασία έχει την εντύπωση ότι χρησιμοποιεί ένα μεγάλο επίπεδο πεδίο διευθύνσεων.

- Οι διευθύνσεις (Εικονικές) πρέπει να μεταφραστούν σε φυσικές διευθύνσεις πριν χρησιμοποιηθούν.

- Εικονική μνήμη διαιρεί τη φυσική μνήμη σε σελίδες και τις κατανέμει σε διάφορες διεργασίες.

- virtual και physical pages

• Η εικονική μνήμη χωρίζεται σε δύο κατηγορίες.

- ***Σελιδοποίηση***: Μπλοκς σταθερού μήκους(Pages:Fix size blocks)

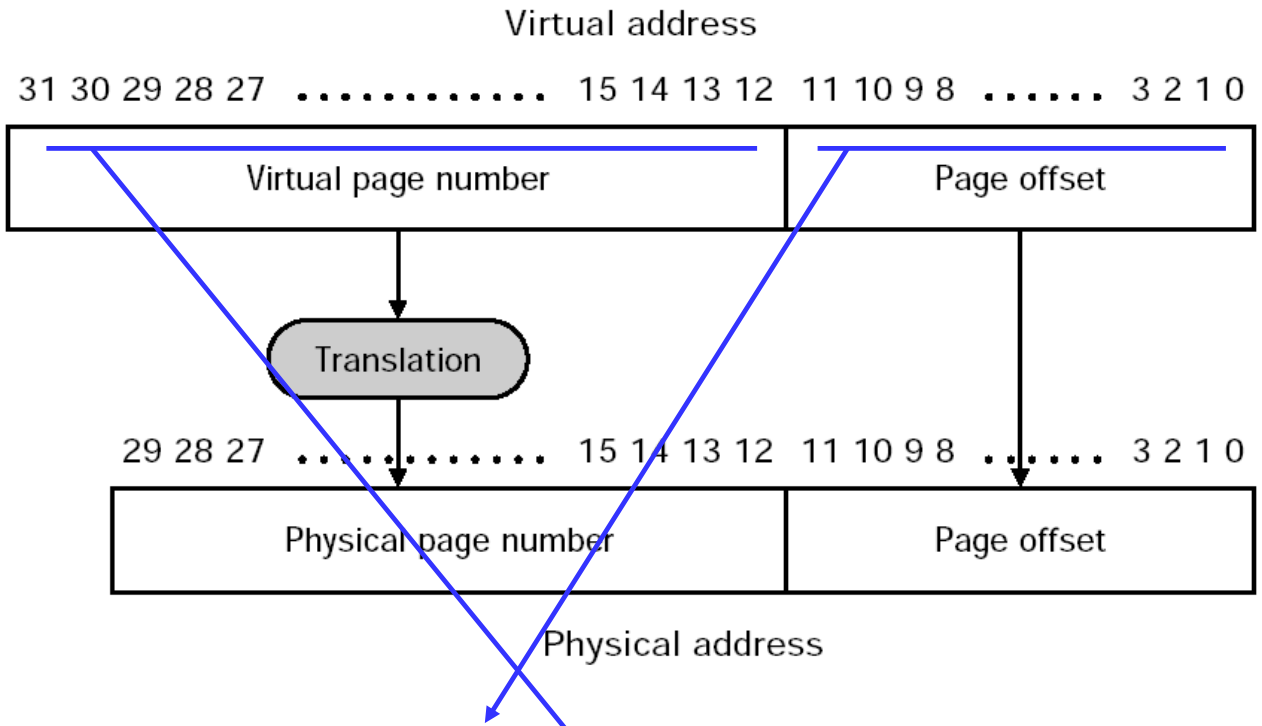
- ***Τμηματοποίηση***: Μεταβαλλόμενο μέγεθος μπλοκ (segments)

Σελιδοποιημένη Εικονική Μνήμη: (Paged Virtual Memory:)

- **Σταθερό μήκος διευθύνσεων**: αριθμός σελίδας και διεύθυνση.

- **Εσωτερικός Κατακερματισμός** (Internal Fragmentation)

- **Σελιδοποίηση μετά από Αίτηση**: Οι σελίδες προσκομίζονται μόνο όταν πραγματικά χρειαστεί μια σελίδα και όχι προκαταβολικά.



- Μέγεθος Σελίδας 2^{12} = 4 KB
- Αριθμός Εικονικών Σελίδων= 2^{20} (*καθορίζετε από μέγεθος address*)
- Κυρίως Κνήμη = 1GB, Αριθμός Φυσικών Σελίδων= 2^{18}

Block (page) size	4-64 kbytes (Large, Huge pages)
Hit time	100s clock cycles
Miss penalty	100K-1M clock cycles (για δίσκο)
(Access time)	80% penalty
(Transfer time)	20% penalty
Miss rate	0.00001%-0.001%
Main memory size	GBs

Cache και ΕΙΚΟΝΙΚΗ ΜΝΗΜΗ

- ***Το μέγεθος της διεύθυνσης του Επεξεργαστή προσδιορίζει το μέγεθος της Εικονικής μνήμης.***
  - Η αντικατάσταση ενός μπλοκ στις αποτυχίες του cache γίνεται από το υλικό.
  - Η αντικατάσταση σελίδων στην εικονική μνήμη είναι η δουλειά του Λειτουργικού Συστήματος.
- Η δευτερεύουσα μνήμη χρησιμοποιείται και για το σύστημα αρχείων.

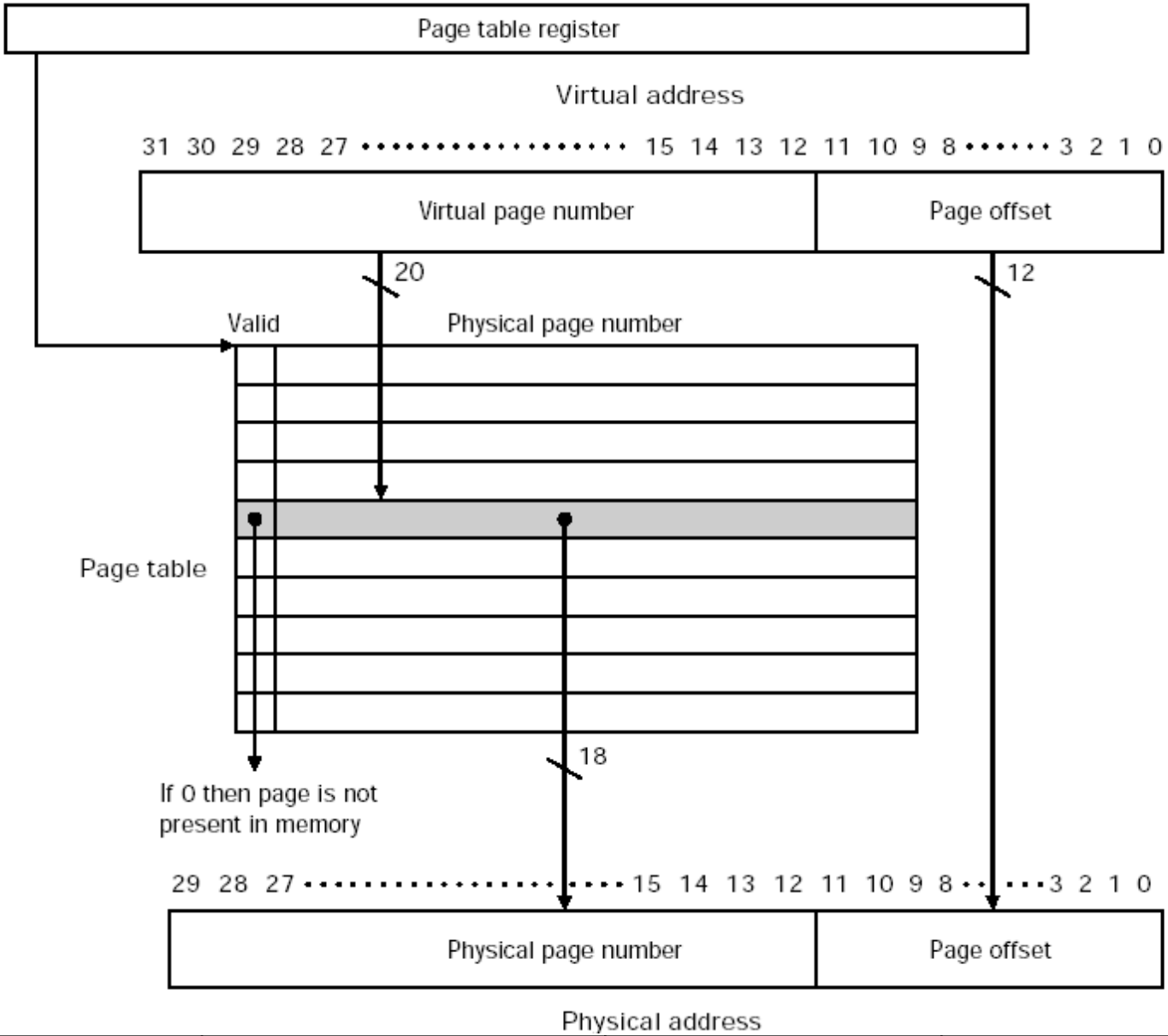
• **Q1: Που τοποθετείται ένα μπλοκ στην κύρια μνήμη**

Οπουδήποτε (Συσχετιστική μνήμη στην ορολογία του cache) (Anywhere, Fully-Associative in cache terminology)

• **Q2: Πως βρίσκουμε εάν μία σελίδα βρίσκεται στην** κύρια μνήμη

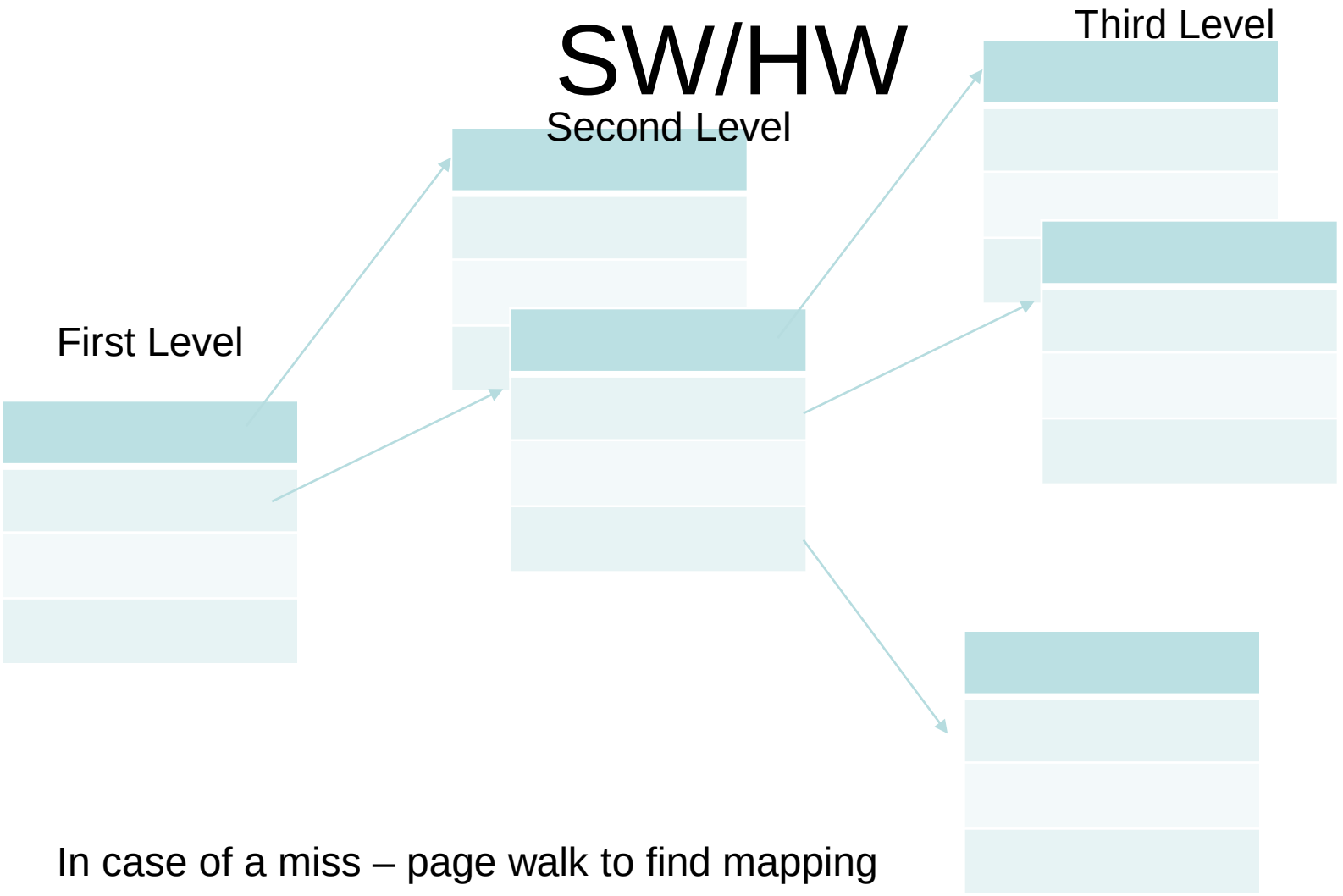
Πίνακας Σελίδων (Page Table)

- Χρησιμοποιά σαν δείκτη το αριθμό τις σελίδας (Indexed by the page number)
- **Περιέχει την αντιστοιχίας Εικονικών σε-φυσικές διευθύνσεις** (Contains the physical address of the block)



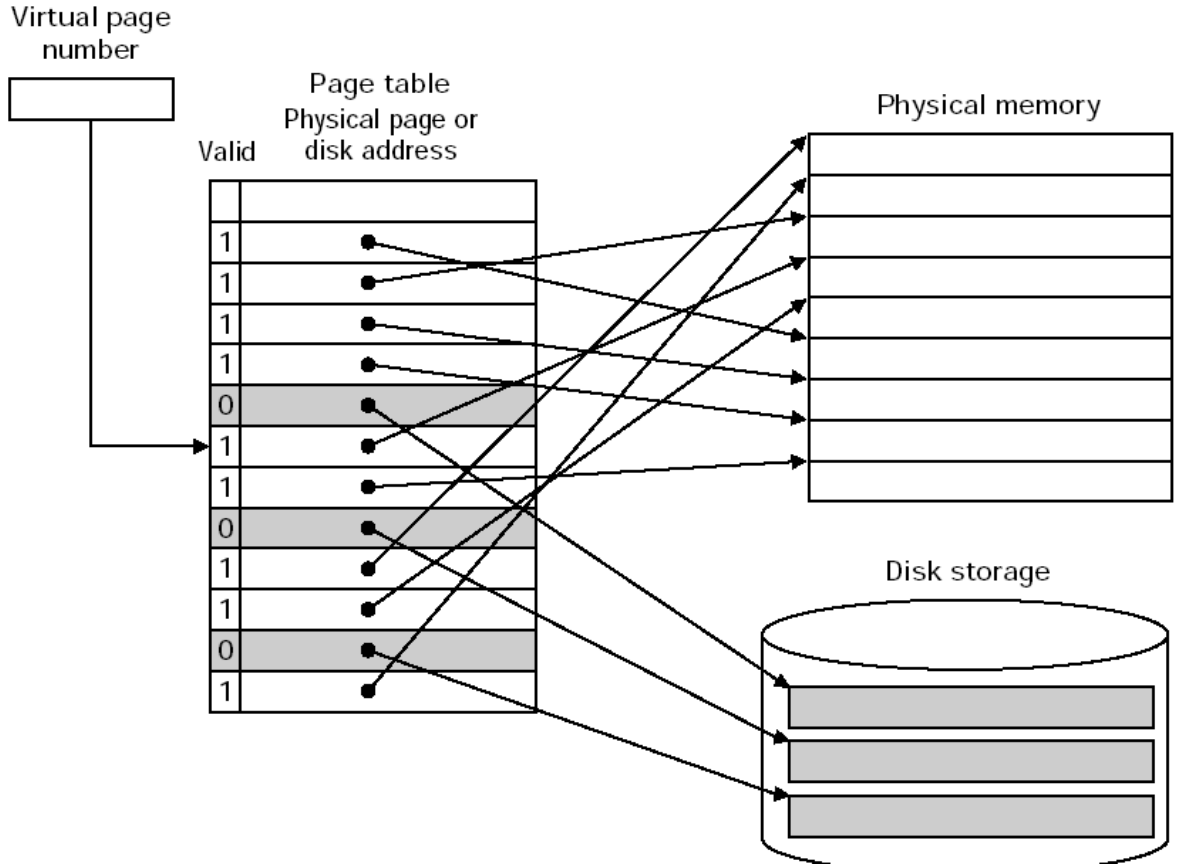
Page size 2^{12} =4kB	**Virtual space 2^{32} =4GB**	**Physical Space= 2^{30}**	**# of Entries in PT=2^{20}**

Multi-level Page Table in SW/HW



In case of a miss – page walk to find mapping

Αποτυχία Σελίδας (Page Fault)

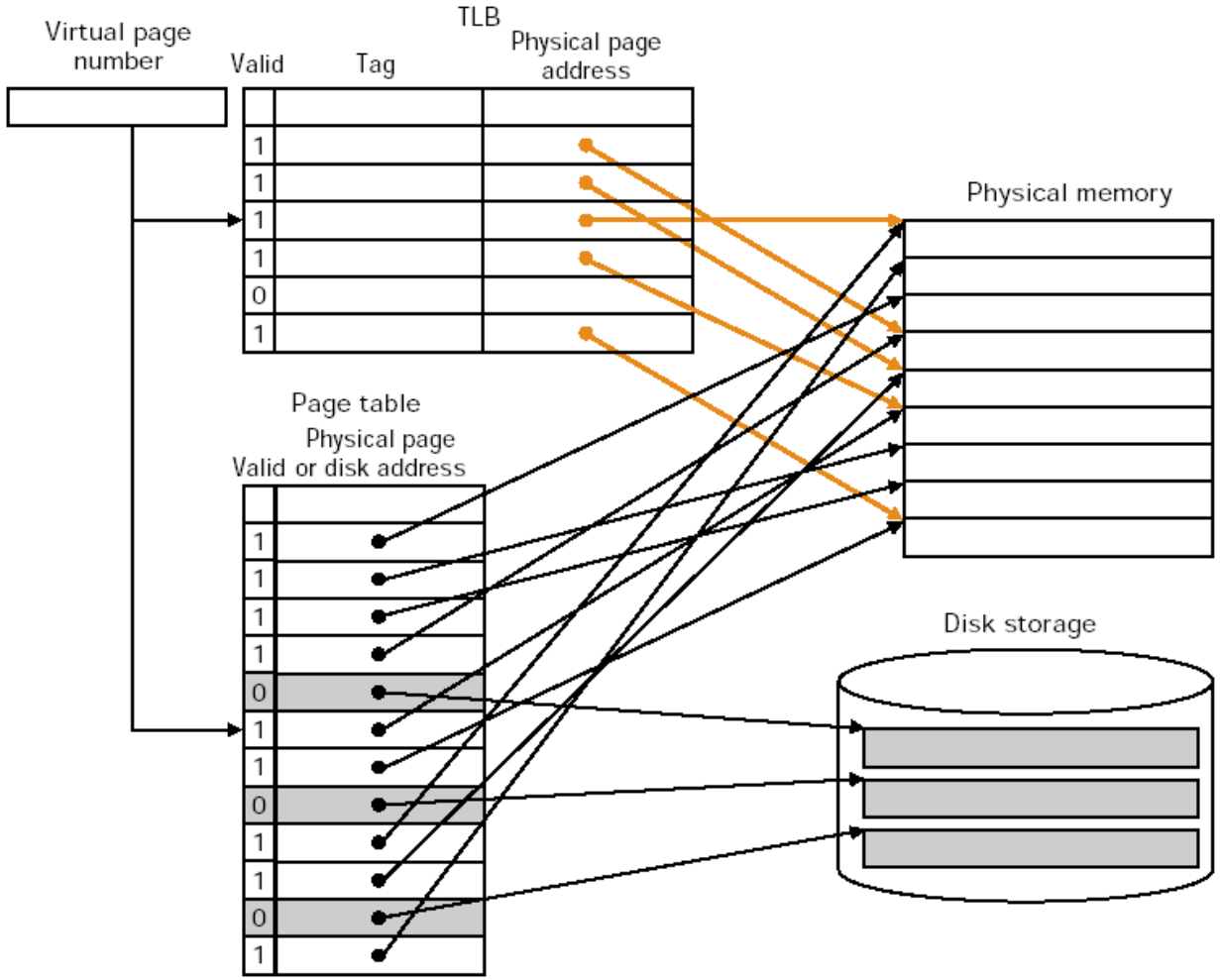


Ο πίνακας σελίδων αντιστοιχεί την κάθε σελίδα

- Σε μια σελίδα στην μνήμην ή
- Σε μια σελίδα στην επόμενη βαθμίδα στην Ιεραρχία μνήμης

Translation Lookaside Buffer (TLB): μία μνήμη cache αφιερώνεται στη μετάφραση διεύθυνσης (a cache dedicated to address translation)

•**Το TLB περιέχει ένα υποσύνολο της αντιστοιχίας Εικονικών σε φυσικές διευθύνσεις**
page frame number, protection field, use bit, dirty bit



- Q3: Ποια σελίδα πρέπει να αντικατασταθεί σε περίπτωση αποτυχίας στην Εικονική μνήμη (Which Block should be replaced on virtual memory miss (page fault)?)

  - Least Recently Used (LRU)

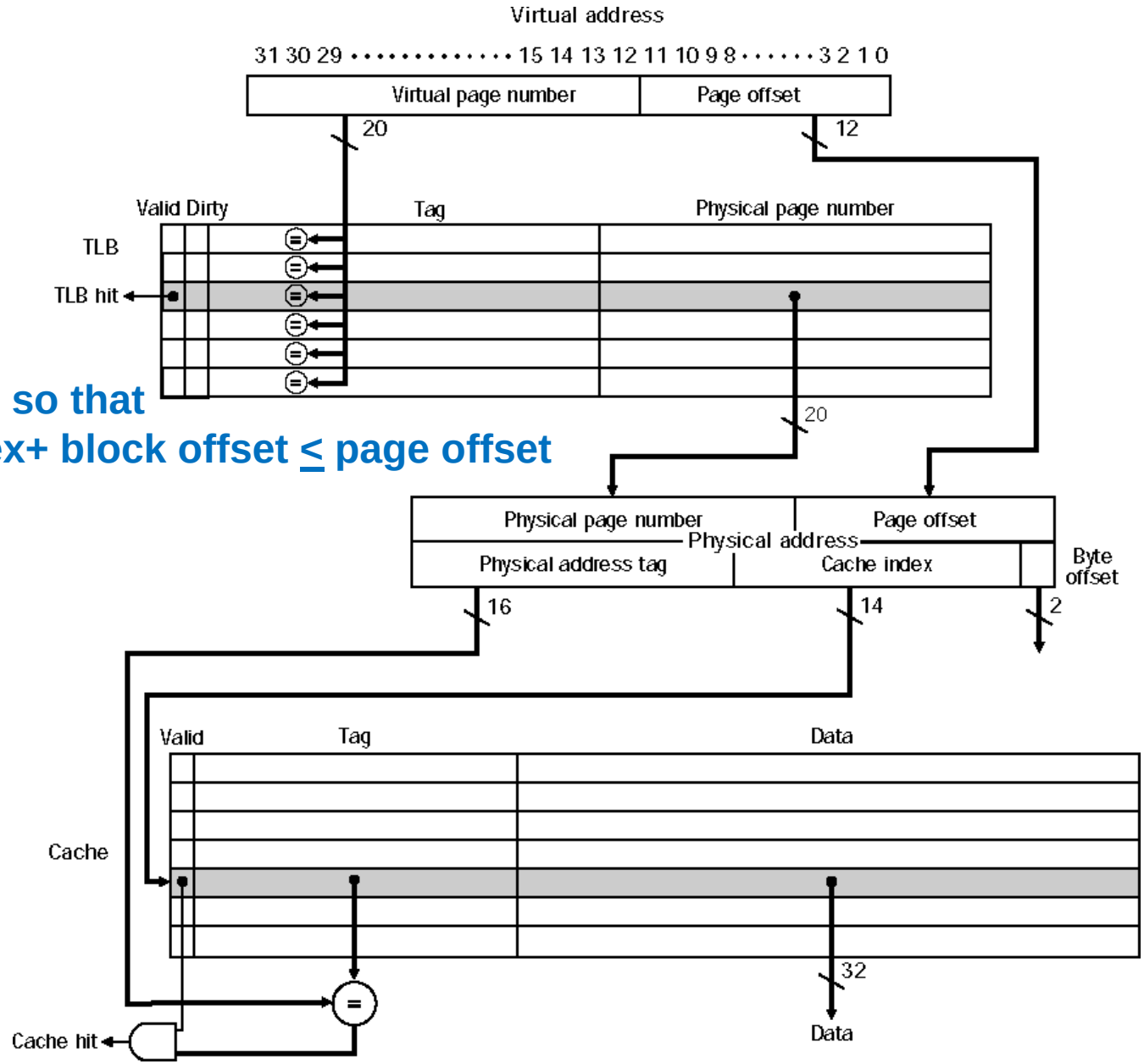
- Q4: Τι γίνεται στην εγγραφή (What happens on write)

  - ΥΣΤΕΡΟΕΓΓΡΑΦΗ (WRITE BACK)

- Μία εικονική διεύθυνση πρέπει πρώτα να περάσει από το TLB πριν καταλήξει στο CACHE (A virtual address must go through the TLB before it goes to the cache)

- Ο χρόνος επιτυχίας αυξάνεται (Cache Hit time is stretched)

- Λύσεις (Solutions):
  - Ταυτόχρονη πρόσβαση στο cache και στο TLB (Access cache with page offset while accessing the TLB)
  - $ με εικονικές διευθύνσεις(Virtually Indexed Cache)
    - Aliasing and synonyms (different threads same virtual address)



Size cache so that
cache index+ block offset ≤ page offset

Cache Index and Tag

VIVT: virtually indexed virtually tagged (L1)

VIPT: virtually indexed physically tagged (L1)

PIPT: physically indexed physically tagged (L2, LLC)

Other Memory Hierarchy Features

Second level TLBs

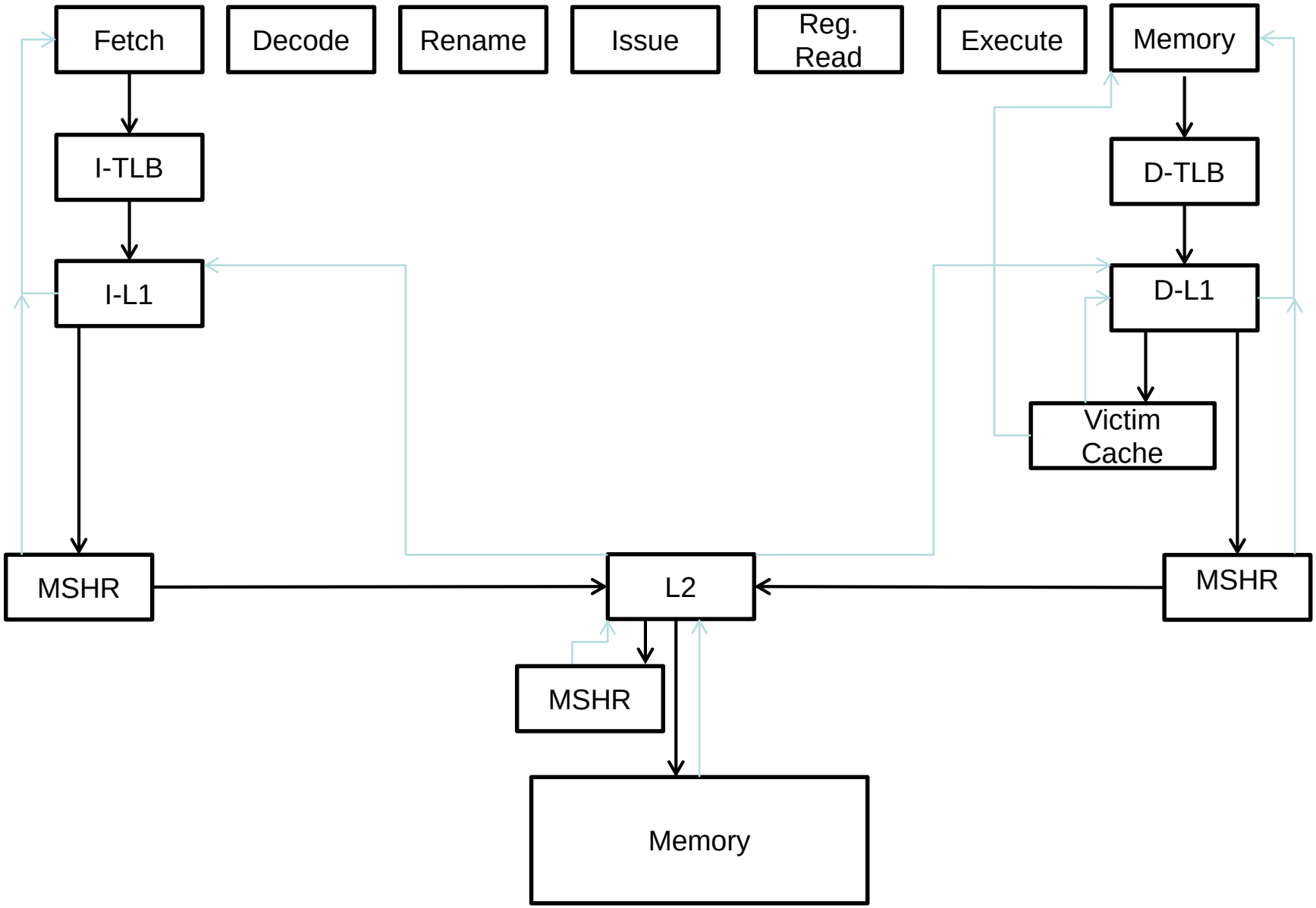
Banked Caches

- parallel access to different banks
- Each bank some sets

MSHRs

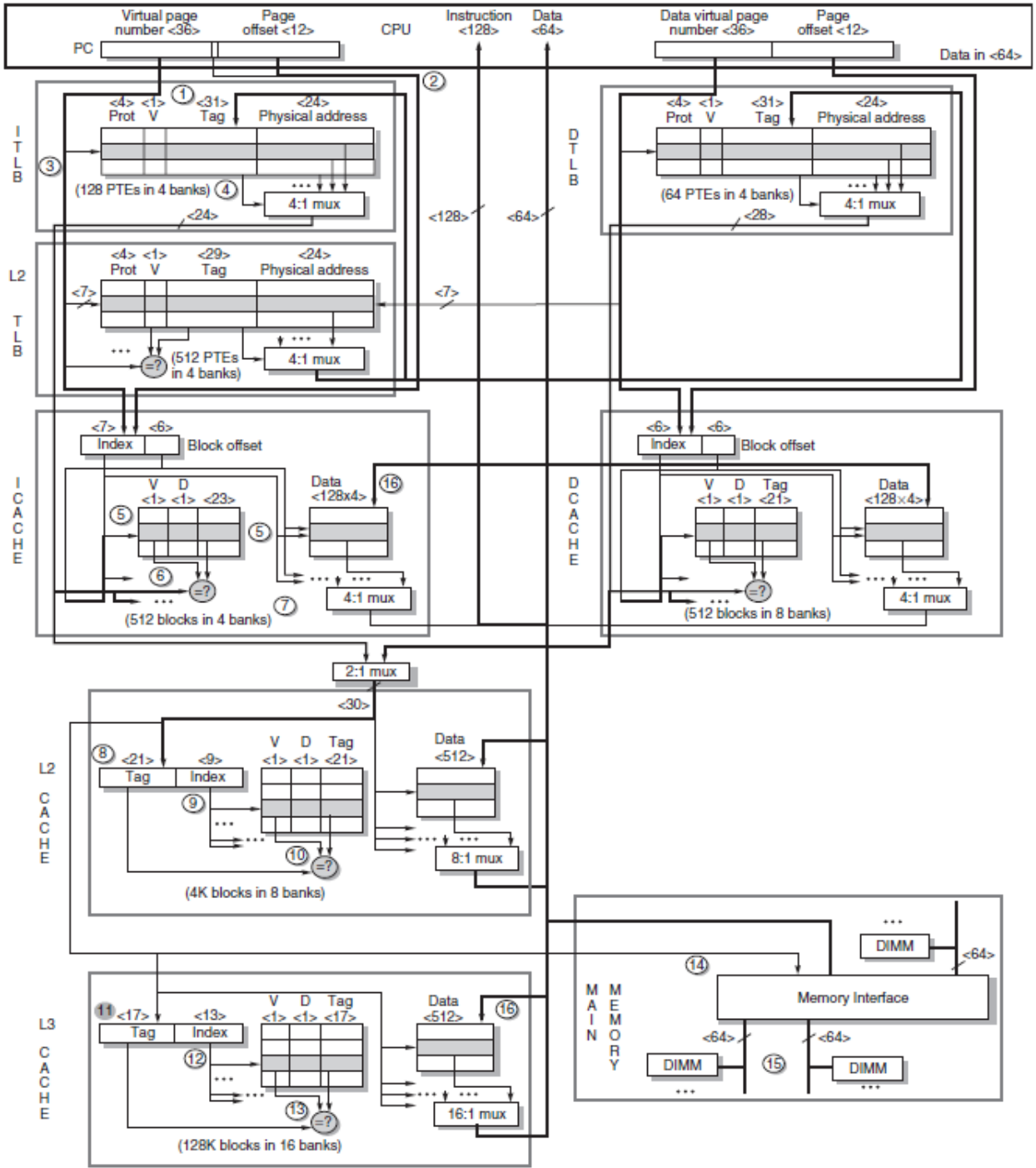
- Non-blocking cache
- Hit under miss
- Miss under miss

Prefetch


Fetch
Decode
Rename
Issue
Reg. Read
Execute
Memory
I-TLB
D-TLB
I-L1
D-L1
Victim Cache
MSHR
L2
MSHR
MSHR
Memory

Typical Cache Hierarchy Values

Feature	Typical values for L1 caches	Typical values for L2 caches	Typical values for paged memory	Typical values for a TLB
Total size in blocks	250–2000	2,500–25,000	16,000–250,000	40–1024
Total size in kilobytes	16–64	125–2000	1,000,000–1,000,000,000	0.25–16
Block size in bytes	16–64	64–128	4000–64,000	4–32
Miss penalty in clocks	10–25	100–1000	10,000,000–100,000,000	10–1000
Miss rates (global for L2)	2%–5%	0.1%–2%	0.00001%–0.0001%	0.01%–2%


Virtual page number <36>
Page offset <12>
CPU
Instruction <128>
Data <64>
Data virtual page number <36>
Page offset <12>
PC
Data in <64>
ITLB
DTLB
<4> Prot
<1> V
<31> Tag
<24> Physical address
(128 PTEs in 4 banks)
4:1 mux
(64 PTEs in 4 banks)
<128>
<64>
<28>
L2 TLB
<29> Tag
<7>
(512 PTEs in 4 banks)
ICACHE
DCACHE
<6> Block offset
Index
Data <128x4>
<23>
<21> Tag
(512 blocks in 4 banks)
(512 blocks in 8 banks)
2:1 mux
<30>
L2 CACHE
<9> Index
Data <512>
8:1 mux
(4K blocks in 8 banks)
L3 CACHE
<17> Tag
<13> Index
16:1 mux
(128K blocks in 16 banks)
MAIN MEMORY
DIMM
Memory Interface

Characteristic	ARM Cortex-A8	Intel Nehalem
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	32 KiB each for instructions/data	32 KiB each for instructions/data per core
L1 cache associativity	4-way (I), 4-way (D) set associative	4-way (I), 8-way (D) set associative
L1 replacement	Random	Approximated LRU
L1 block size	64 bytes	64 bytes
L1 write policy	Write-back, Write-allocate(?)	Write-back, No-write-allocate
L1 hit time (load-use)	1 clock cycle	4 clock cycles, pipelined
L2 cache organization	Unified (instruction and data)	Unified (instruction and data) per core
L2 cache size	128 KiB to 1 MiB	256 KiB (0.25 MiB)
L2 cache associativity	8-way set associative	8-way set associative
L2 replacement	Random(?)	Approximated LRU
L2 block size	64 bytes	64 bytes
L2 write policy	Write-back, Write-allocate (?)	Write-back, Write-allocate
L2 hit time	11 clock cycles	10 clock cycles
L3 cache organization	–	Unified (instruction and data)
L3 cache size	–	8 MiB, shared
L3 cache associativity	–	16-way set associative
L3 replacement	–	Approximated LRU
L3 block size	–	64 bytes
L3 write policy	–	Write-back, Write-allocate
L3 hit time	–	35 clock cycles

Characteristic	ARM Cortex-A8	Intel Core I7
Virtual address	32 bits	48 bits
Physical address	32 bits	44 bits
Page size	Variable: 4, 16, 64 KiB, 1, 16 MiB	Variable: 4 KiB, 2/4 MiB
TLB organization	1 TLB for instructions and 1 TLB for data Both TLBs are fully associative, with 32 entries, round robin replacement TLB misses handled in hardware	1 TLB for instructions and 1 TLB for data per core Both L1 TLBs are four-way set associative, LRU replacement L1 I-TLB has 128 entries for small pages, 7 per thread for large pages L1 D-TLB has 64 entries for small pages, 32 for large pages The L2 TLB is four-way set associative, LRU replacement The L2 TLB has 512 entries TLB misses handled in hardware

Some analysis

How many lines of a page reside in a cache?

How many accesses in a set between evictions?

No pending stores for a page?

(write-read analysis)

(write-write analysis)

Distance between write-miss and read?

(selective write-allocate/no-write-allocate)

Writeback from L1 to L2 (L2 to L3)

What if writeback misses at a lower level?

Fetched block and overwrite it? Why?

[Coherence?]

Cache Example

8-blocks, 1 word/block, direct mapped

Initial state

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Miss	110

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	**Y**	**10**	**Mem[10110]**
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
26	11 010	Miss	010

Index	V	Tag	Data
000	N		
001	N		
010	**Y**	**11**	**Mem[11010]**
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Hit	110
26	11 010	Hit	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
16	10 000	Miss	000
3	00 011	Miss	011
16	10 000	Hit	000

Index	V	Tag	Data
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		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
18	10 010	Miss	010

Index	V	Tag	Data
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		

Associativity Example

Compare 4-block caches

- Direct mapped, 2-way set associative, fully associative
- Block access sequence: 0, 8, 0, 6, 8

Direct mapped

Block address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0	0	miss	**Mem[0]**			
8	0	miss	**Mem[8]**			
0	0	miss	**Mem[0]**			
6	2	miss	Mem[0]		**Mem[6]**	
8	0	miss	**Mem[8]**		Mem[6]	

Associativity Example

2-way set associative

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	**Mem[0]**			
8	0	miss	Mem[0]	**Mem[8]**		
0	0	hit	**Mem[0]**	Mem[8]		
6	0	miss	Mem[0]	**Mem[6]**		
8	0	miss	**Mem[8]**	Mem[6]		

- Fully associative

Block address		Hit/miss	Cache content after access			
0		miss	**Mem[0]**			
8		miss	Mem[0]	**Mem[8]**		
0		hit	**Mem[0]**	Mem[8]		
6		miss	Mem[0]	Mem[8]	**Mem[6]**	
8		hit	Mem[0]	**Mem[8]**	Mem[6]	

Πως επιλέγουμε ένα σύνολο στο cache;

Χρήση ενός index-to-2^{index} decoder