Επανάληψη 3: Κρυφές Μνήμες, Εικονική Μνήμη, Περιφερειακά-Επικοινωνία, Συνοχή, Προχ. Επ.

Άνοιξη 2021 – Μανόλης Κατεβαίνης

(Ενημέρωση Άνοιξη '24: διαφάνειες ασκ. 13b, 14a)

Ιδιότητα της Τοπικότητας στα Προγράμματα

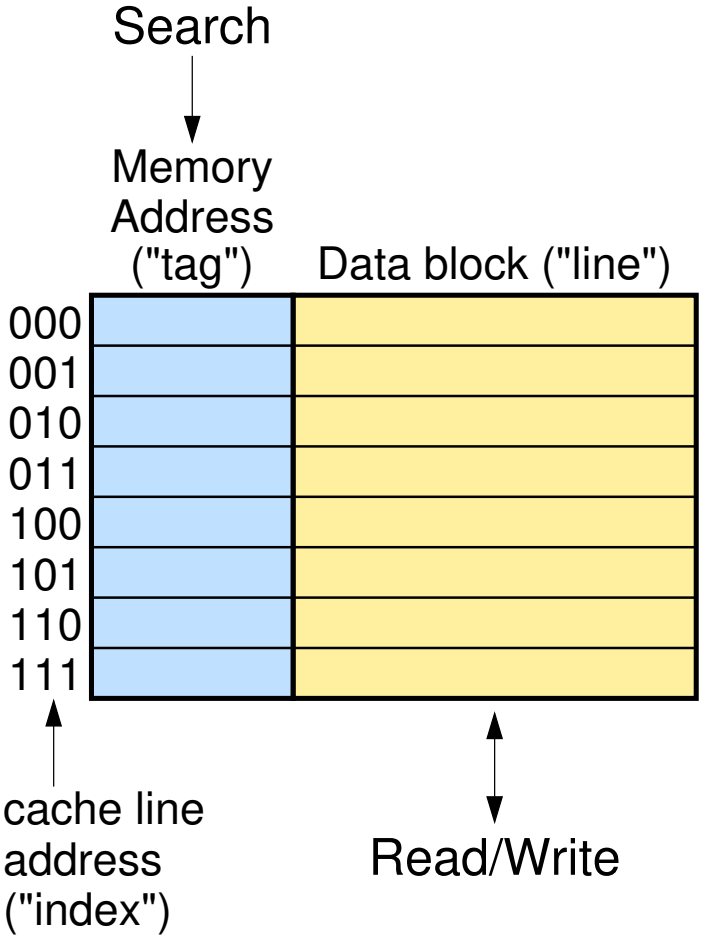
- Κοιτώντας τις προσπελάσεις μνήμης σε ένα σχετικά μικρό *χρονικό παράθυρο*, αυτές συνήθως απευθύνονται σε διευθύνσεις με:
- *Χρονική Τοπικότητα (Temporal Locality)*:
  - πολλαπλές προσπελάσεις στις *ίδιες* θέσεις
  - π.χ. εντολές μέσα σε βρόχο, συχνά χρησιμοπ. μεταβλητές

  ⇒ πρόσφατα προσπελ. λέξεις έχουν αυξ. πιθαν. εκ νέου προσπέλ.
- *Χωρική Τοπικότητα (Spatial Locality)*:
  - προσπελάσεις σε *γειτονικές* θέσεις
  - π.χ. επόμ. εντολές, πίνακες σειριακά προσπ., κορυφή στοίβας

  ⇒ λέξεις κοντά σε πρόσφατα προσπ. έχουν αυξ. πιθαν. προσπέλ.

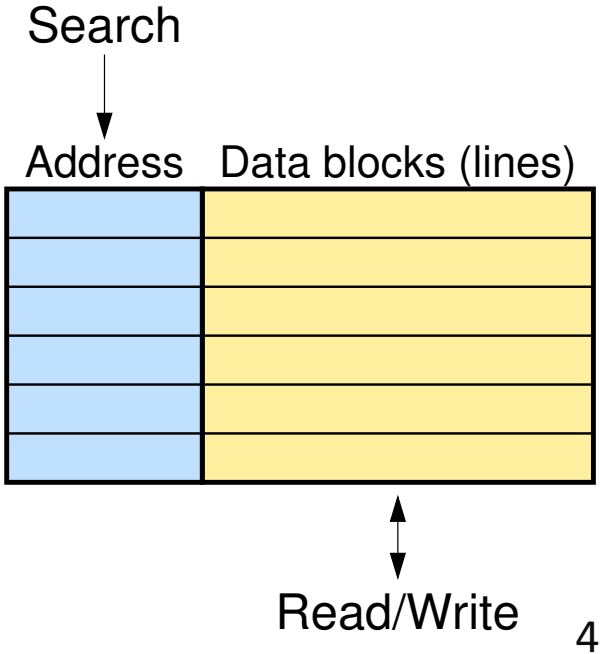
Η γενική δομή μιάς Κρυφής Μνήμης

- Πλήθος «θέσεων» πολύ μικρότερο από χώρο διευθύνσεων μν.
  - cache “lines” or cache “blocks”
  - cache “index”: η θέση (διευθ.) μιάς γραμμής μέσα στην κρυφή μνήμη
- Κάθε «θέση» περιέχει δεδομένα (αντίγραφο γειτονιάς) από τη μνήμη, καθώς και τη διεύθυνση της μνήμης (“tag”) απ’ όπου την φέραμε αυτή τη γειτονιά
- Αναζήτηση βάσει Διευθ. Μνήμης



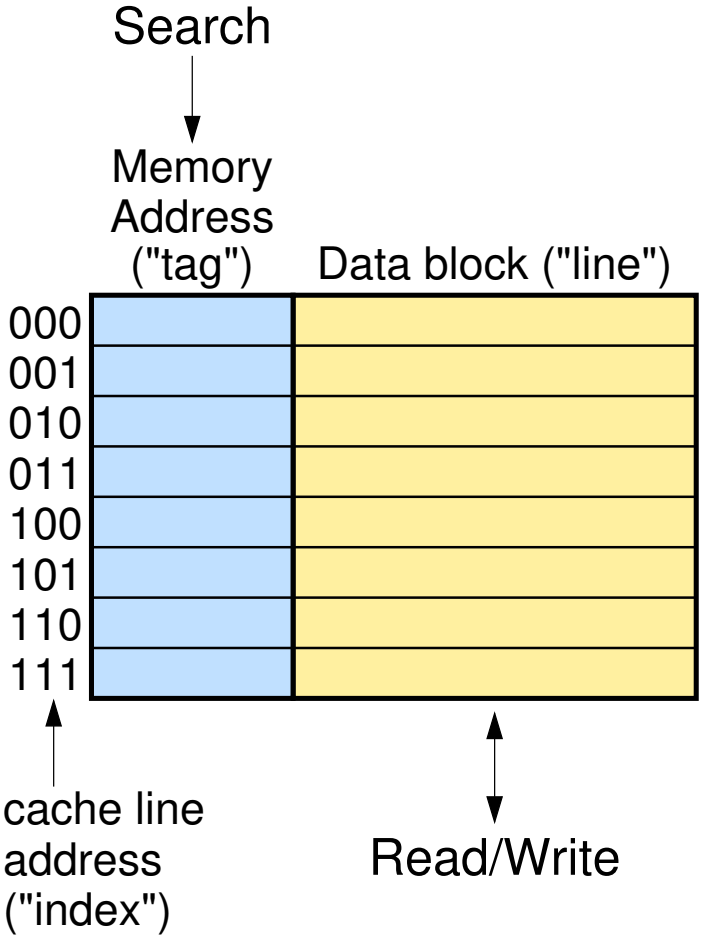
Πού επιτρέπεται και πού θα ψάξουμε δοθείσα Διευθ.

- Δοθείσας μιάς διεύθυνσης μνήμης, πώς ψάχνουμε εάν και πού την έχουμε; Πού μπορεί να βρίσκεται;
- Επιτρέπεται οπουδήποτε; (“fully associative”)
  - υπερβολικά δαπανηρή αναζήτηση
  - περιττά μεγάλη ευελιξία
- Μόνον σε ορισμένα μέρη επιτρέπεται να την βάλαμε;
  - αρκεί να την ψάξουμε εκεί μόνον
  - όσο λιγότερα τα μέρη, τόσο ευκολότερο το ψάξιμο, αλλά και τόσο μικρότερη ευελιξία ποιόν «παλιόν» θα διώξουμε

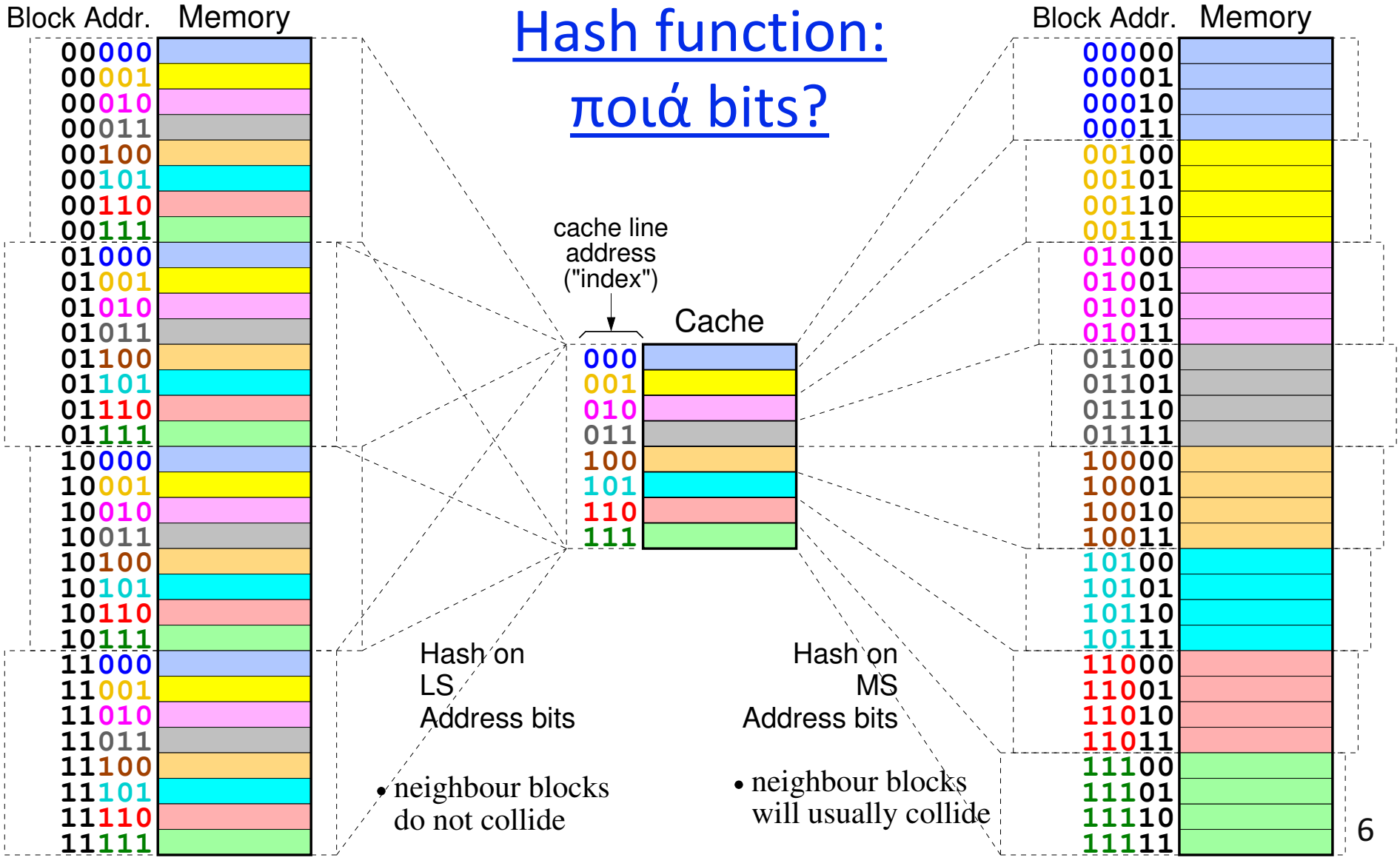


Μονοσήμαντη Απεικόνιση (Direct Mapped Caches)

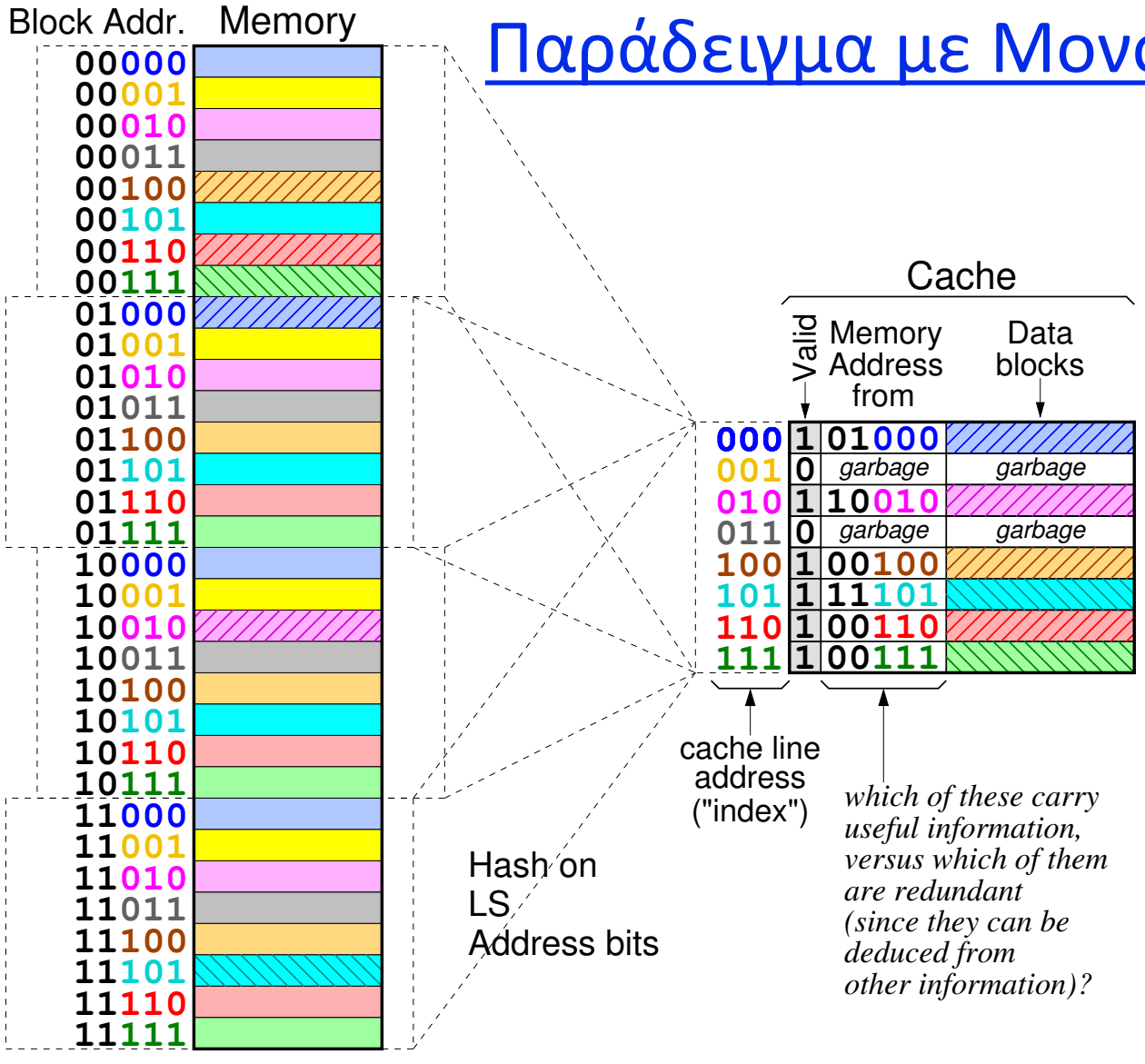
- Η κάθε δοθείσα διεύθ. μνήμης μόνον σε μία θέση (cache index) επιτρέπεται να τοποθετηθεί
  - πολύ περιοριστικό
  - σημαντική απλοποίηση
  - ξεκινάμε με αυτό, και στη συνέχεια θα δούμε την κάπως πιό ευέλικτη και πιό συνηθισμένη οργάνωση
- Πώς προκύπτει η μία, μόνη θέση;
- Index = ***Hash_ f*** (Mem.Addr.)
- Συνάρτ. κατακερματισμού: αρκεί απλώς μερικά bits διευθ. – ποιά;;



Hash function: ποιά bits?

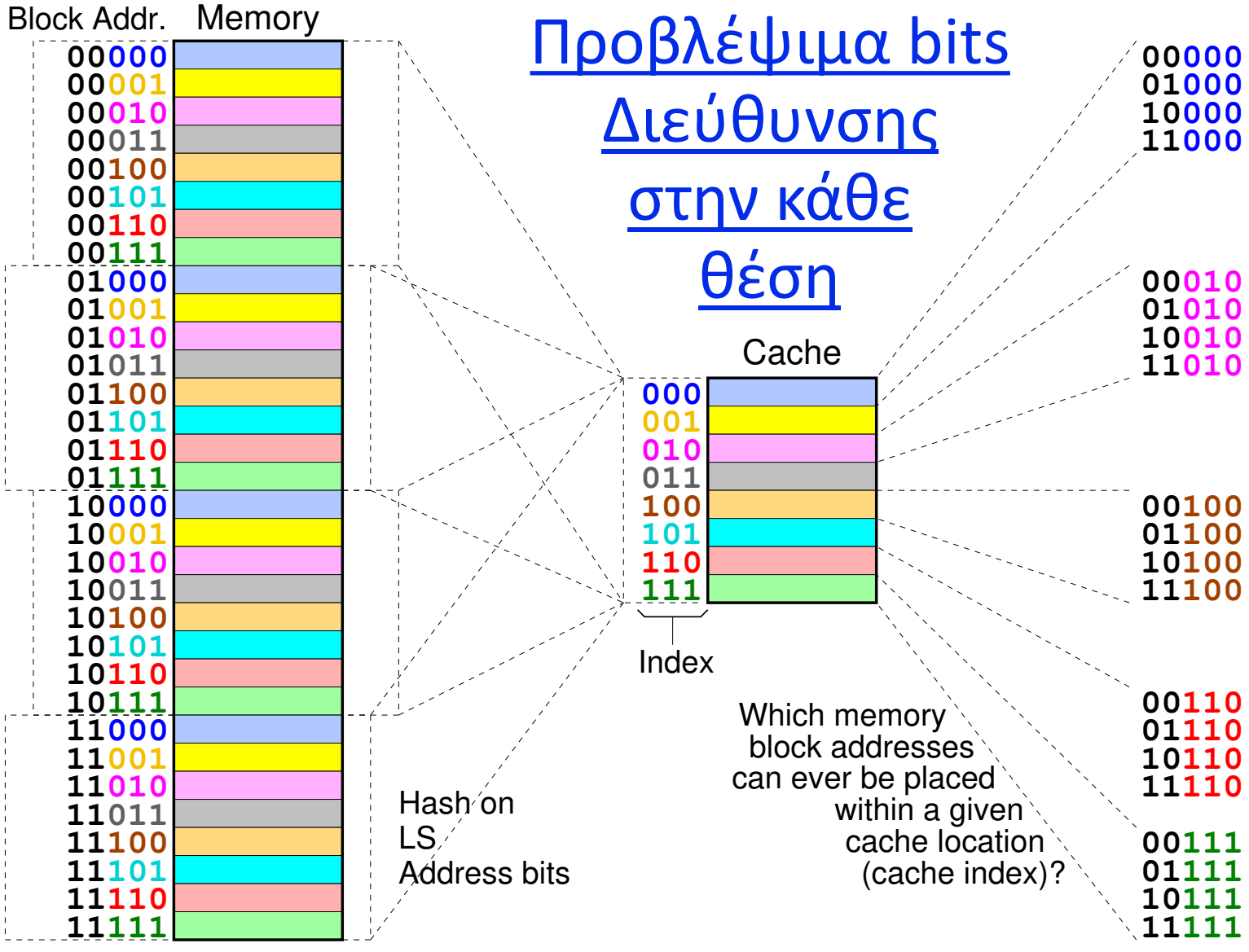


Παράδειγμα με Μονοσημ. Απεικόνιση



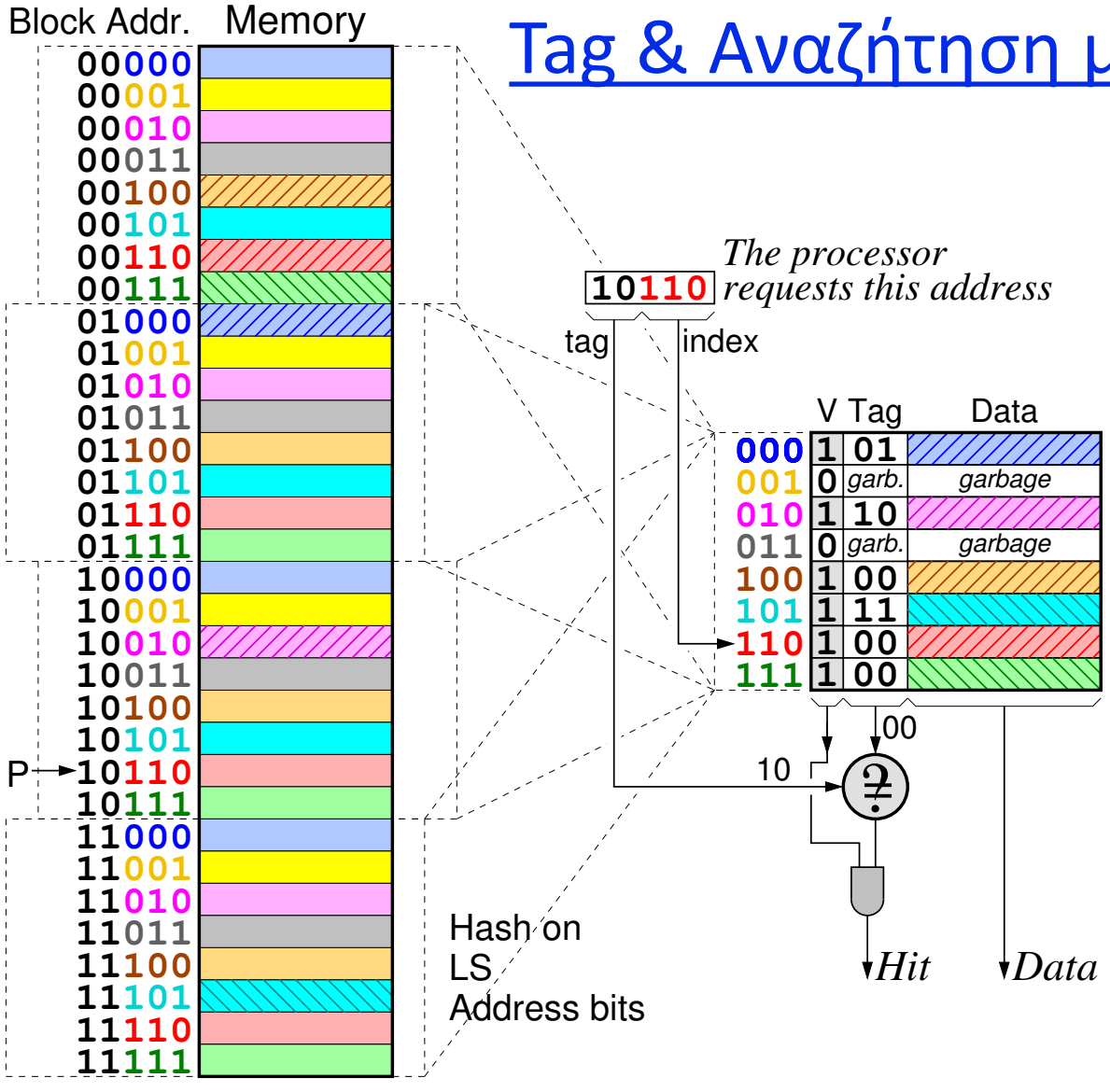
- Κάθε block μνήμης μόνο σε μία θέση επιτρέπεται να μπεί εάν/όταν έλθει στην κρυφή μν. (χρώμα)
- Κάθε block μέσα στην κρυφή μνήμη συνοδεύεται από πληροφ. διεύθυνσης: *τίνος* block μνήμης αποτελεί αντίγραφο;
- *Validity bit* γιά όταν κανένα block μνήμης δεν βρίσκ. τώρα εδώ

Προβλέψιμα bits Διεύθυνσης στην κάθε θέση



- Τα LS bits διεύθυν. μνήμης (= hash function = cache index), είναι γνωστά στην κάθε θέση της κρ. μνήμ., άρα περιττεύουν στην πληροφορία διεύθυν.

Tag & Αναζήτηση με Μονοσ. Απεικόν.

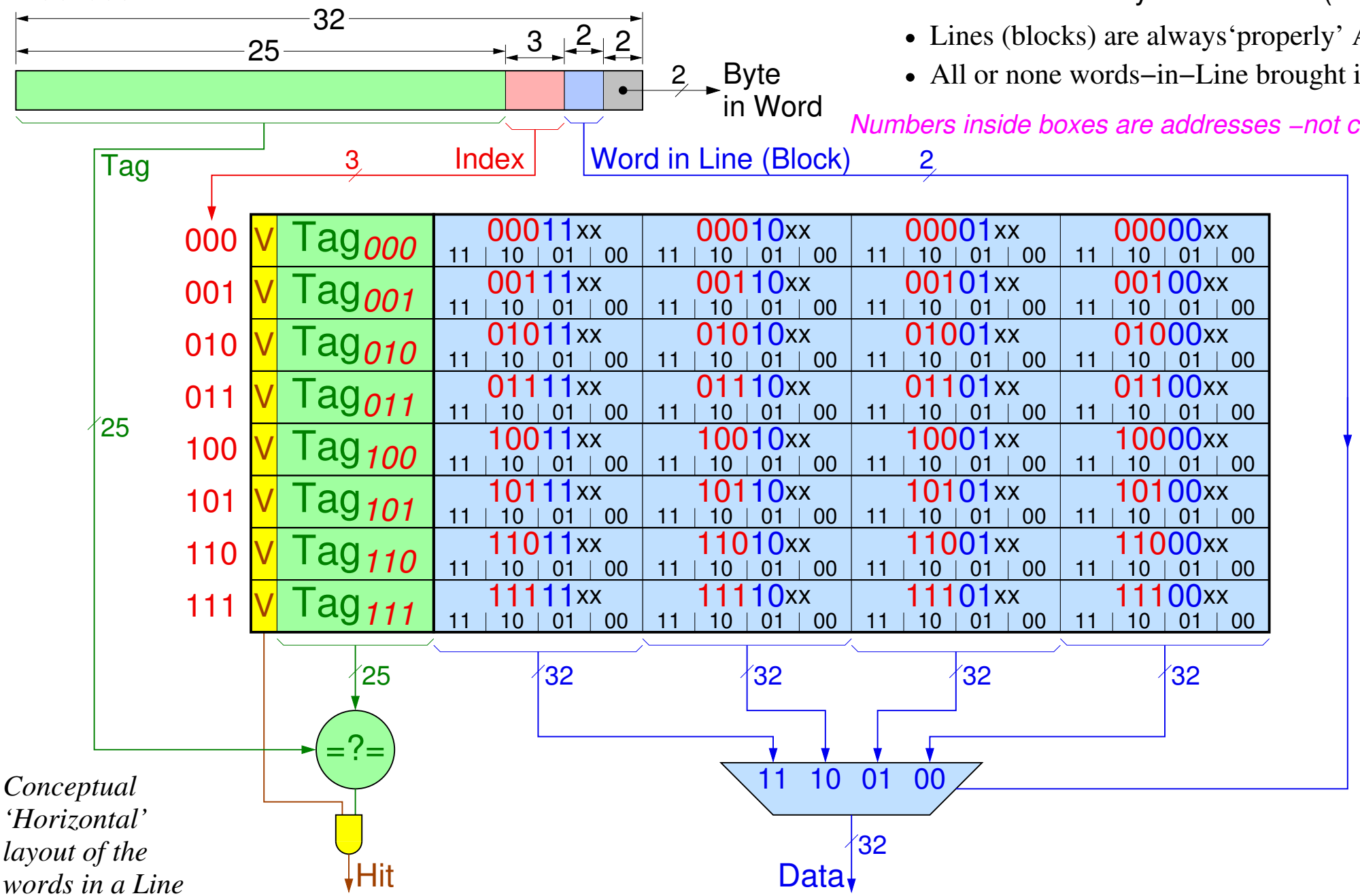


- Tag: περιττό να περιέχει τα Index bits
- Δοθείσας Διεύθυνσης που ψάχνουμε:
- Με το Index από αυτήν, διαβάζουμε: Tag, Valid, και ταυτόχρονα τα Data
- Εάν V==1 και $Tag_{ψάχνουμε} == Tag_{βρίσκουμε}$ τότε Ευστοχία (Hit)
- Έχουμε και τα Data έτοιμα, εάν ευστοχία

Increased Line (Block) Size, to exploit Spatial Locality

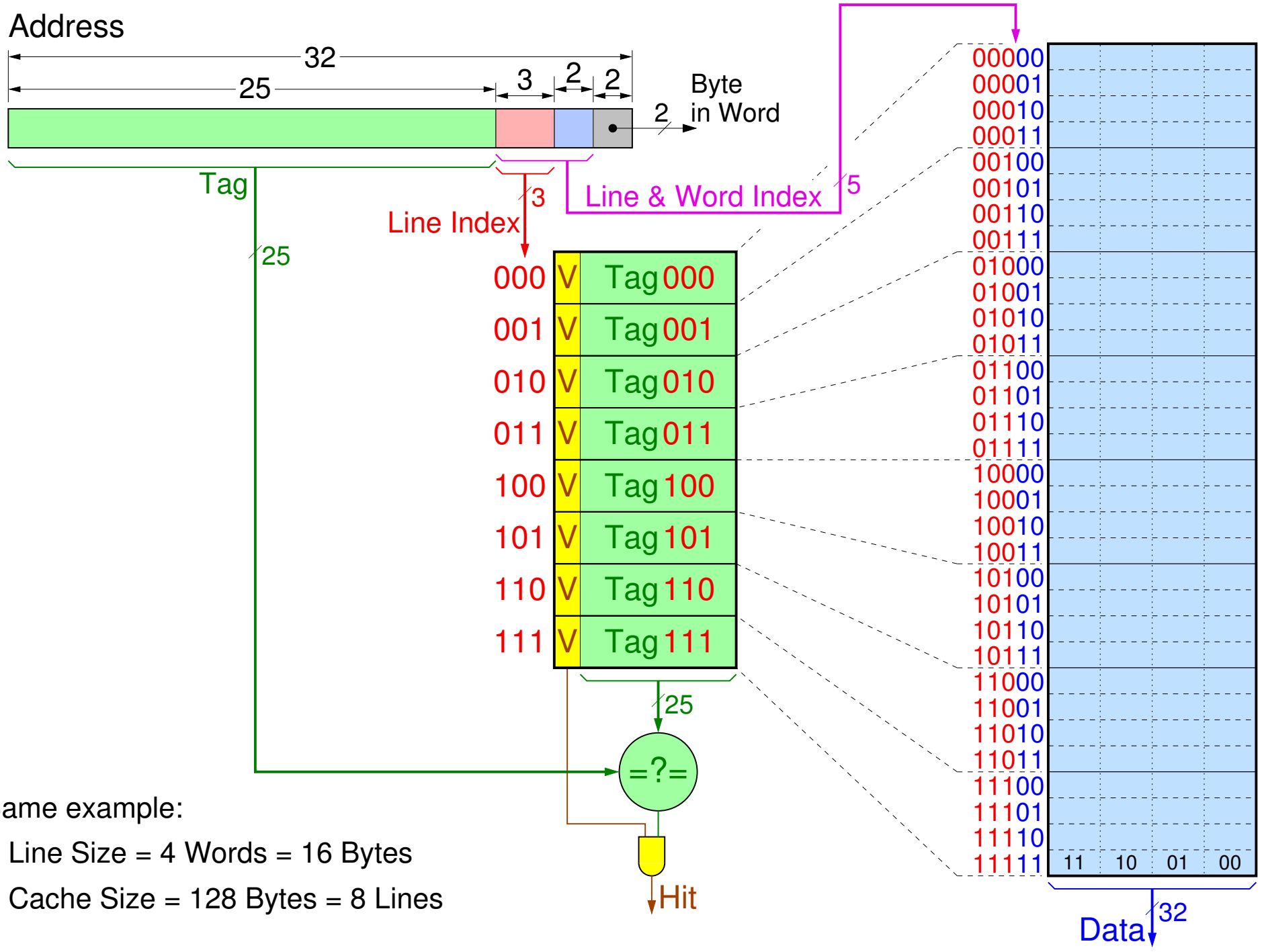
Example:
- Line (Block) Size = 4 Words = 16 Bytes
- Cache Size = 128 Bytes = 8 Lines (Blocks)
  - Lines (blocks) are always‘properly’ Aligned
  - All or none words–in–Line brought in Cache

Numbers inside boxes are addresses –not contents!



Conceptual ‘Horizontal’ layout of the words in a Line

‘Vertical’ Layout of the Words in a Line(Block)



Same example:

- Line Size = 4 Words = 16 Bytes
- Cache Size = 128 Bytes = 8 Lines

Block Size Considerations

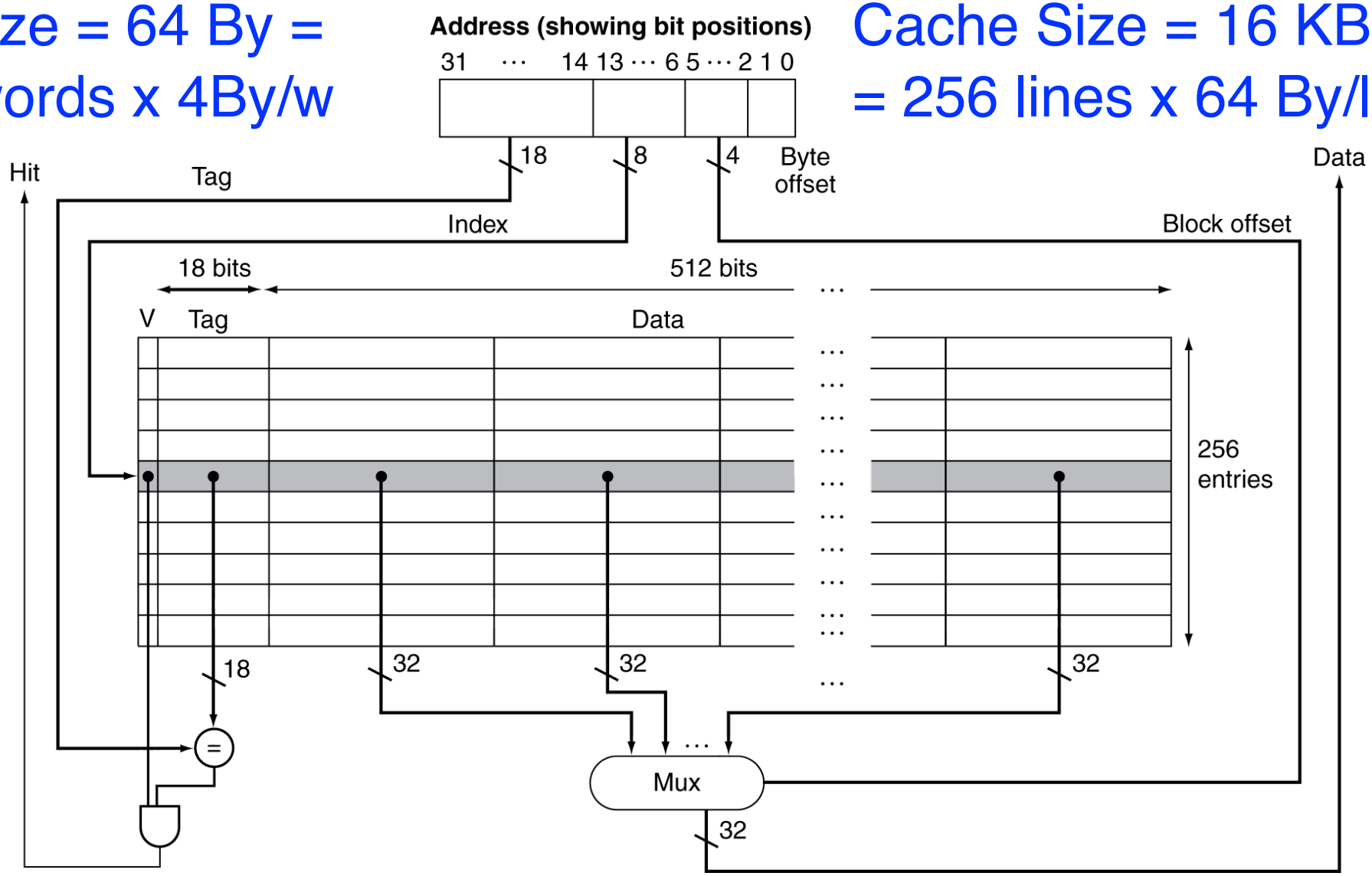
- Larger blocks should reduce miss rate
  - Due to spatial locality
- But in a fixed-sized cache
  - Larger blocks ⇒ fewer of them
    - More competition ⇒ increased miss rate
  - Larger blocks ⇒ pollution
- Larger miss penalty
  - Can override benefit of reduced miss rate
  - Early restart and critical-word-first can help

They also reduce the number of Tags, hence speed up Tag look-up

Example: Intrinsity FastMATH

Line size = 64 By =
= 16 words x 4By/w

Cache Size = 16 KBy =
= 256 lines x 64 By/line



Write-Through Ταυτόχρονη Εγγραφή

- On data-write hit, could just update the block in cache
  - But then cache and memory would be inconsistent
- Write through: also update memory
- But makes writes take longer
  - e.g., if base CPI = 1, 10% of instructions are stores, write to memory takes 100 cycles
    - Effective CPI = 1 + 0.1×100 = 11
- Solution: write buffer
  - Holds data waiting to be written to memory
  - CPU continues immediately
    - Only stalls on write if write buffer is already full

Write-Combining: sequential accesses to DRAM take shorter for subsequent words beyond the first one

Write-Back Ετερόχρονη Εγγραφή

- Alternative: On data-write hit, just update the block in cache
  - Keep track of whether each block is dirty
- When a dirty block is replaced
  - Write it back to memory
  - Can use a write buffer to allow replacing block to be read first

Main Memory is inconsistent with Cache

We will revisit this when talking about I/O, then about multicores…

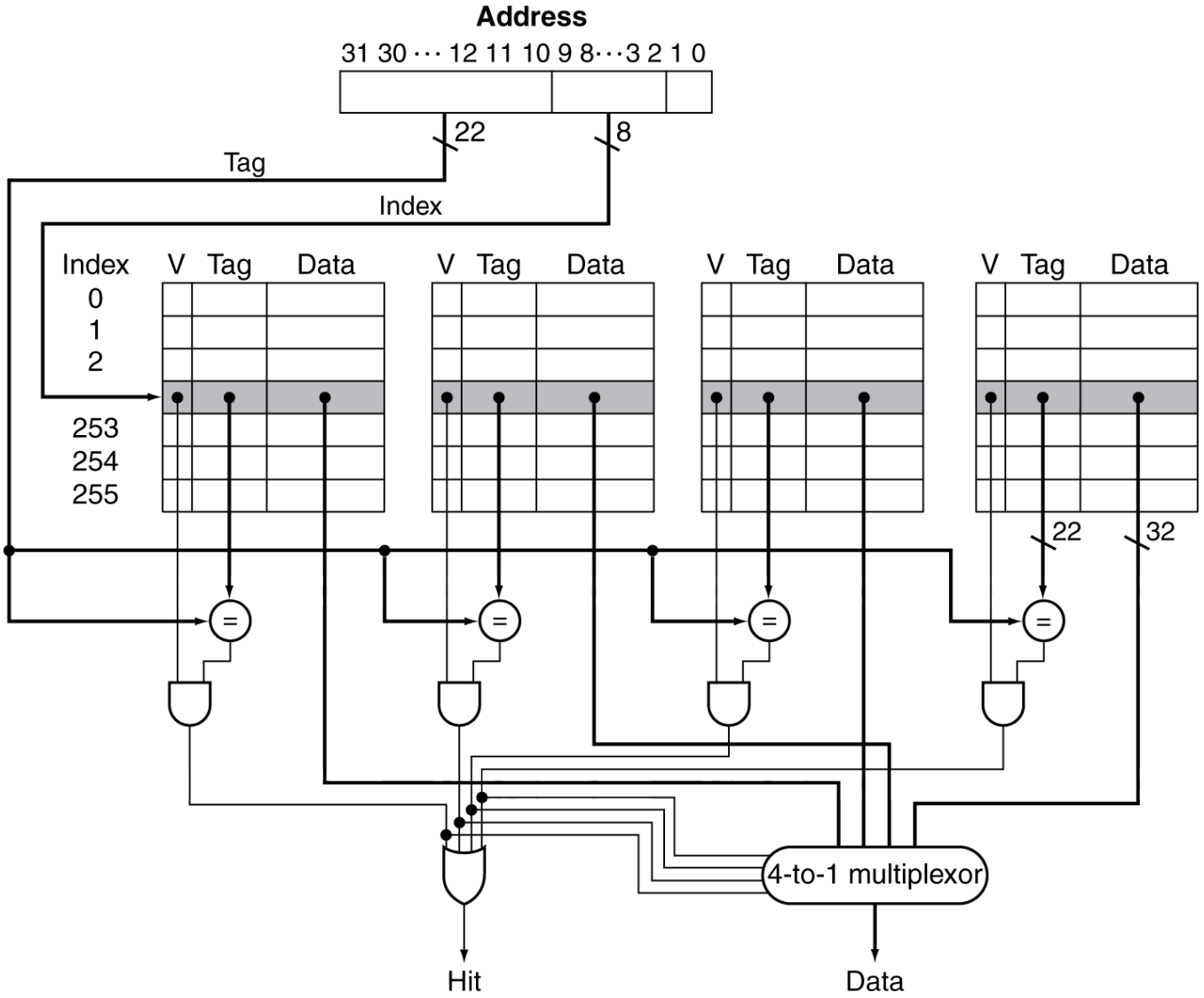
Associative Caches

Μερικώς Προσεταιριστικές Κρυφές Μνήμες

- Fully associative
  - Allow a given block to go in any cache entry
  - Requires all entries to be searched at once
  - Comparator per entry (expensive)
- *n*-way set associative
  - Each set contains *n* entries
  - Block number determines which set
    - (Block number) modulo (#Sets in cache)
  - Search all entries in a given set at once
  - *n* comparators (less expensive)

"Block number" = MS part of memory address, after discarding the LS addr. bits corresponding to byte-within-block (line) offset

Set Associative Cache Organization



Replacement Policy

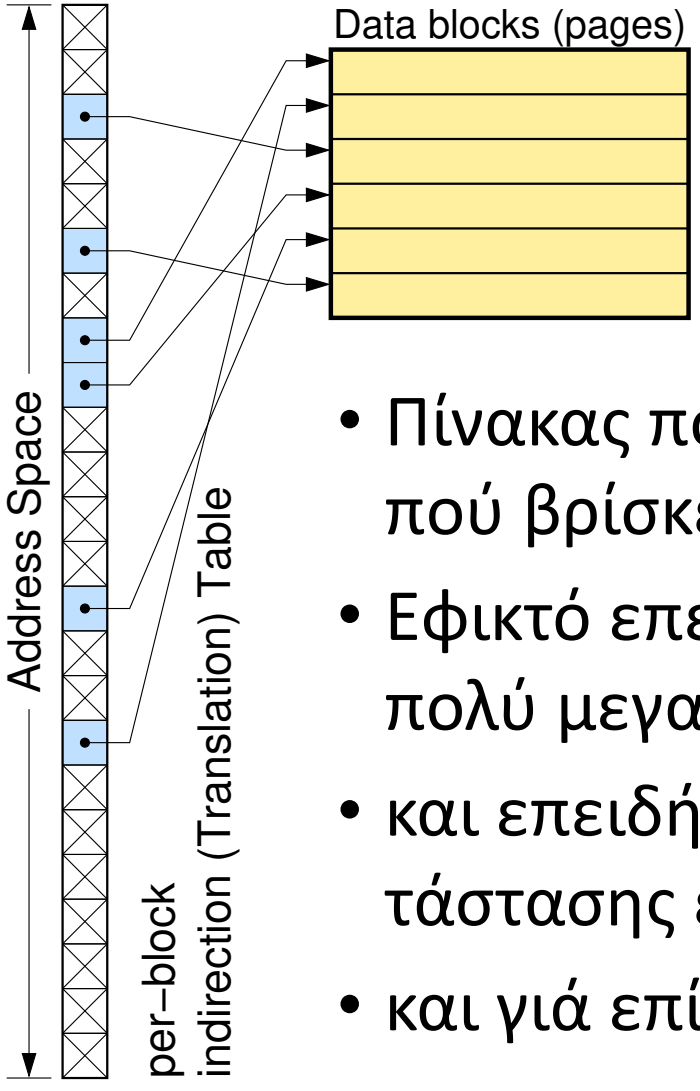
Πώς να προβλέψουμε το μέλλον;;
Συχνά, το πρόσφατο παρελθόν αποτελεί καλή ένδειξη γιά το προσεχές μέλλον!...

- Direct mapped: no choice
- Set associative
  - Prefer non-valid entry, if there is one
  - Otherwise, choose among entries in the set
- Least-recently used (LRU)
  - Choose the one unused for the longest time
    - Simple for 2-way, manageable for 4-way, too hard beyond that
- Random
  - Gives approximately the same performance as LRU for high associativity

Στόχοι Εικονικής Μνήμης: 3 σε 1

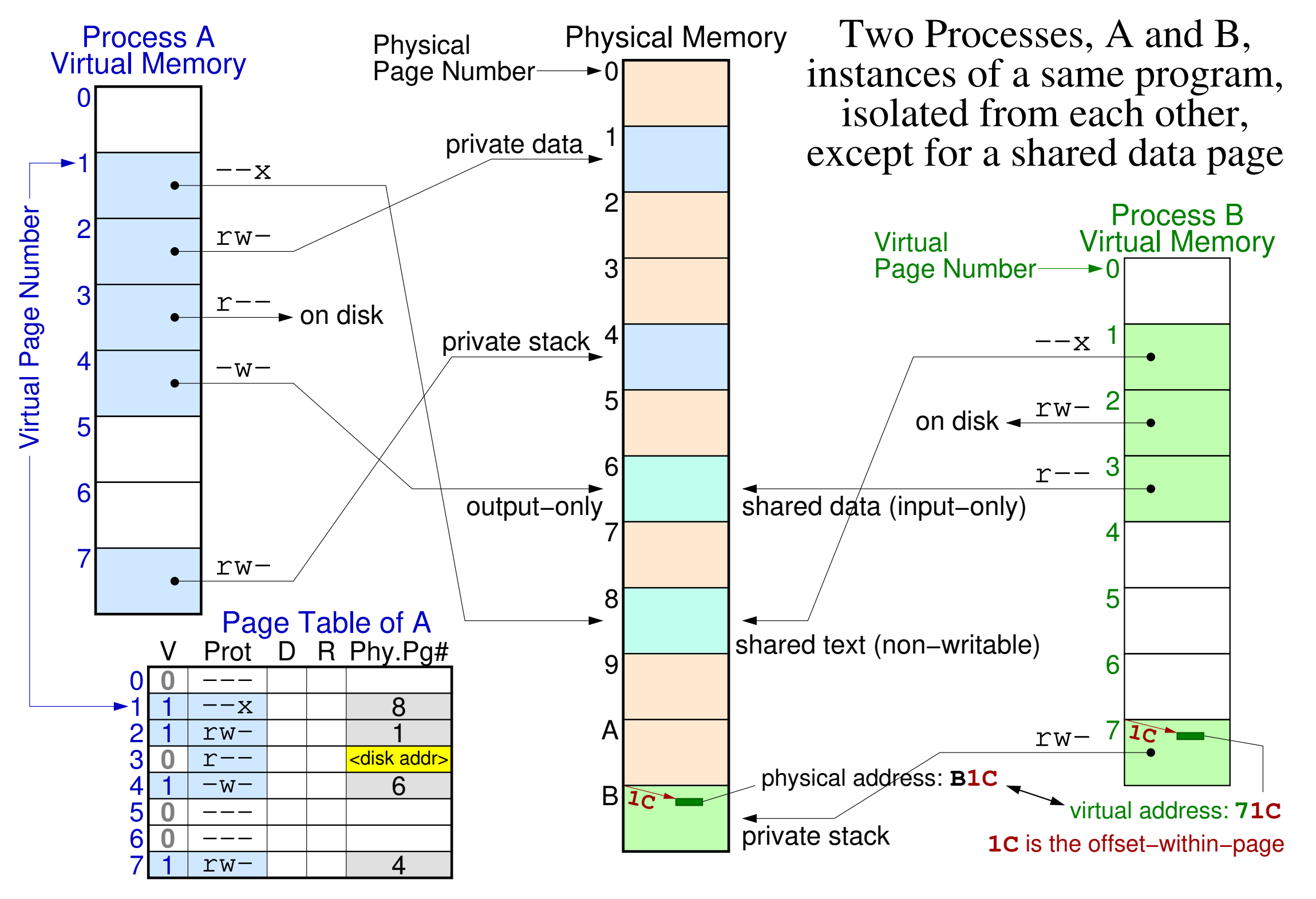
1. Εικονική Μηχανή / Προστασία (Virtual Machine – VM, Protection): κάθε *Διεργασία (Process)* νομίζει ότι έχει όλη τη μηχανή (το χώρο διευθύνσεων) δική της, ανεξάρτητα (προστατευμένη) από τις άλλες
2. Επίπεδο *Ιεραρχίας Μνήμης* πριν την Αποθήκευση/Δίσκο
3. Επίλυση του προβλήματος *Fragmentation*: ο διαθέσιμος χώρος μνήμης γιά νέα διεργασία είναι κομματιασμένος

- Διαχείριση από Λειτουργικό Σύσ. με βοήθεια από το Υλικό



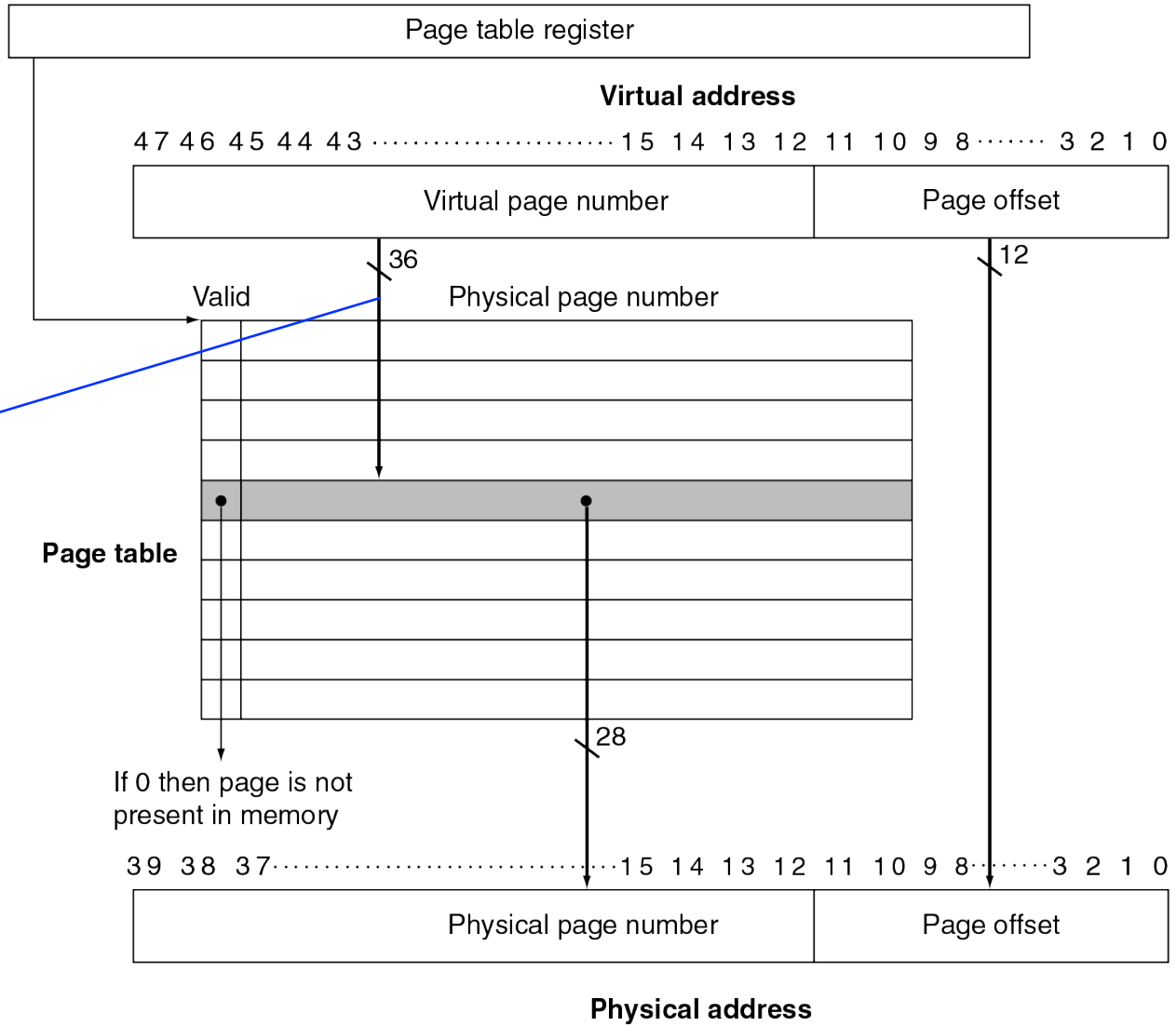
Γενική δομή Εικονικής Μνήμης

- Σε αντίθεση με την οργ. των κρυφών μνημών, όπου ψάχναμε στις υποψήφιες θέσεις γιά την επιθυμητή γραμμή
- Πίνακας που δείχνει, γιά κάθε block (“page”), πού βρίσκεται στην (μικρότερη) Φυσική Μνήμη
- Εφικτό επειδή εδώ τα blocks («σελίδες») είναι πολύ μεγαλύτερα από τις γραμμές κρυφών μν.
- και επειδή η διαχείριση τοποθέτησης / αντικατάστασης είναι σε λογισμικό (Λειτουργικό – OS)
- και γιά επίτευξη εξαιρετικά μικρού miss rate



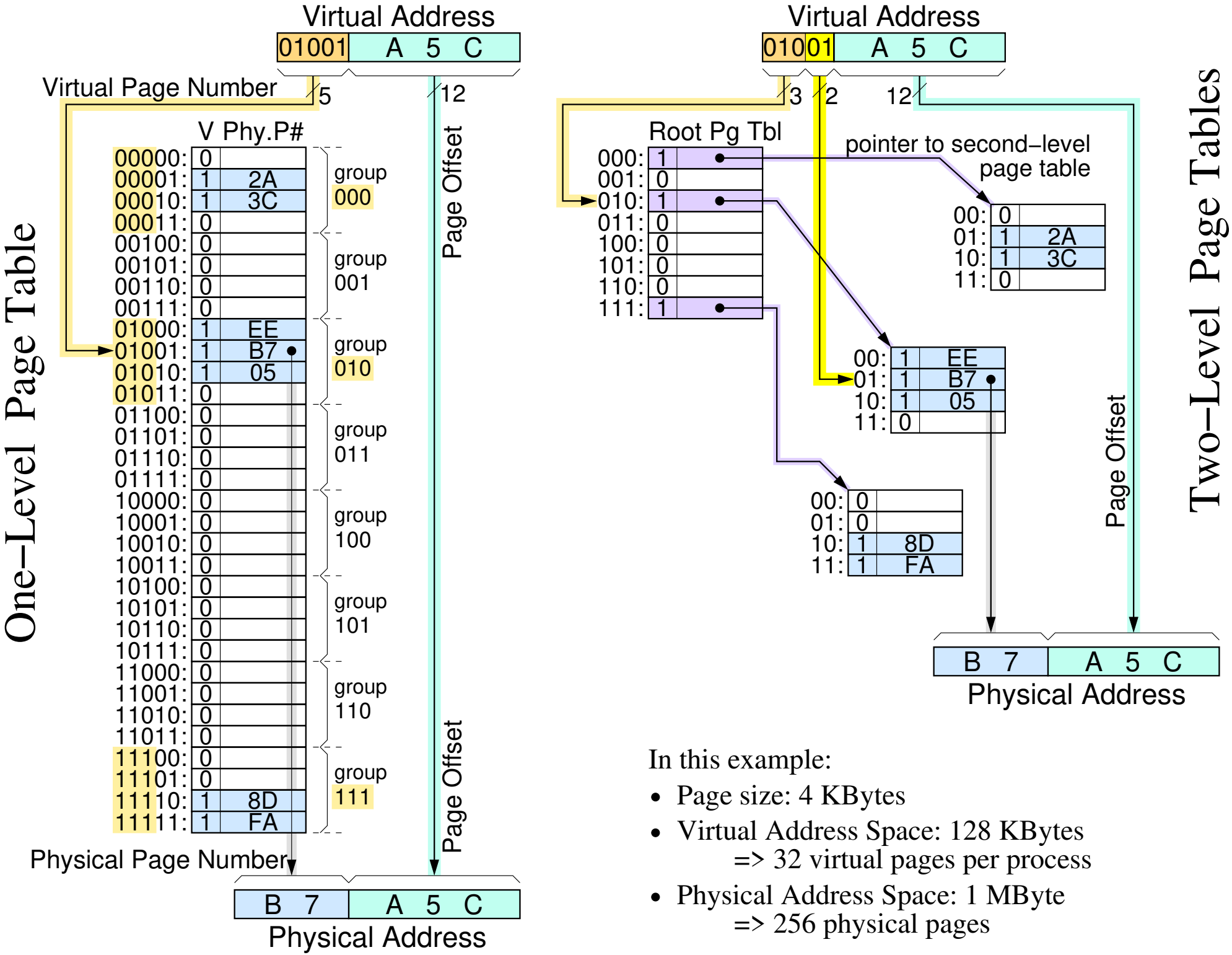
Two Processes, A and B, instances of a same program, isolated from each other, except for a shared data page

Translation Using a Page Table



index within Page Table

Problem: Very Large Size of single-level Page Table; Solution: Multi-Level Page Tables.



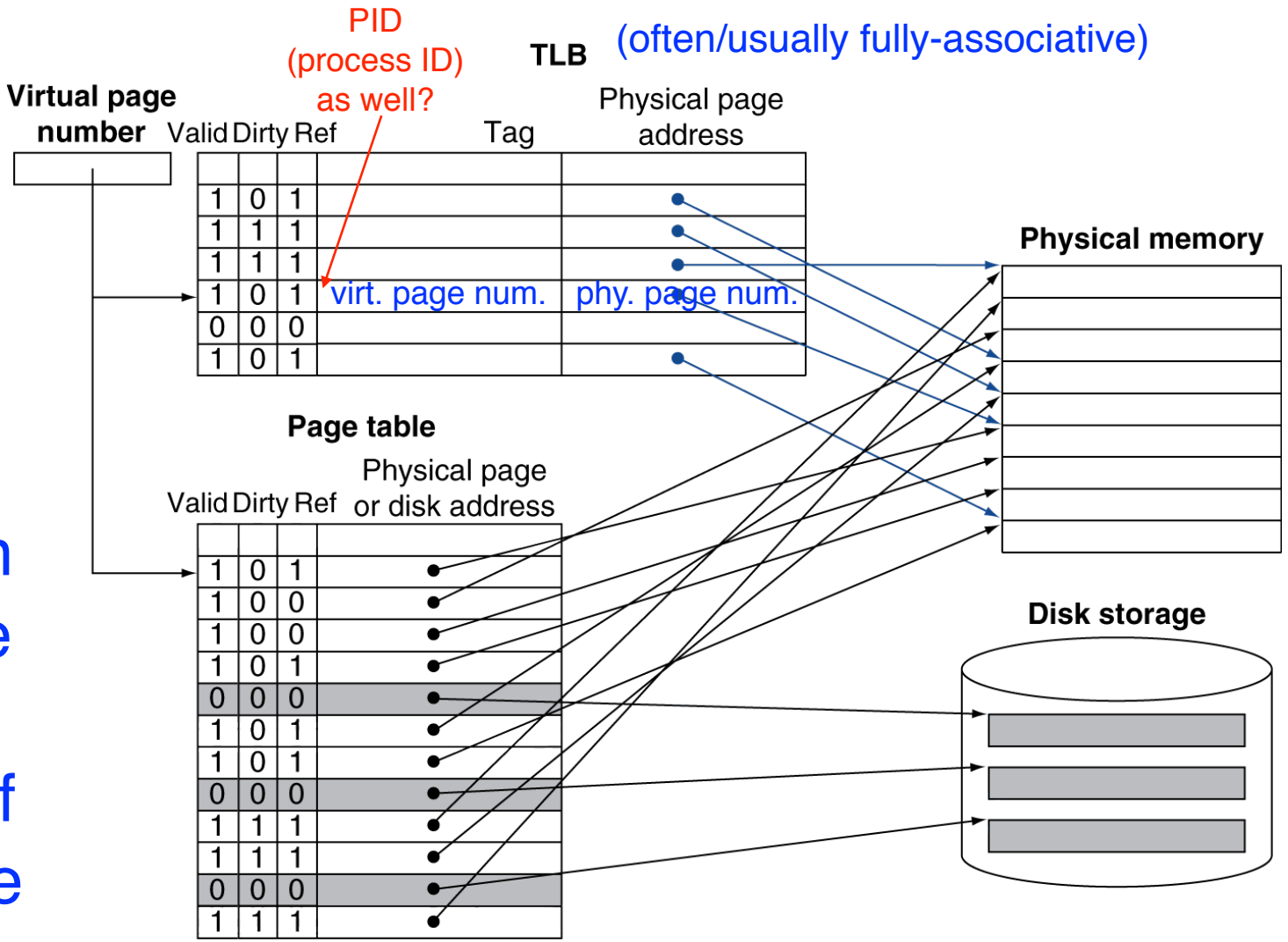
In this example:

- Page size: 4 KBytes
- Virtual Address Space: 128 KBytes
  => 32 virtual pages per process
- Physical Address Space: 1 MByte
  => 256 physical pages

Replacement and Writes

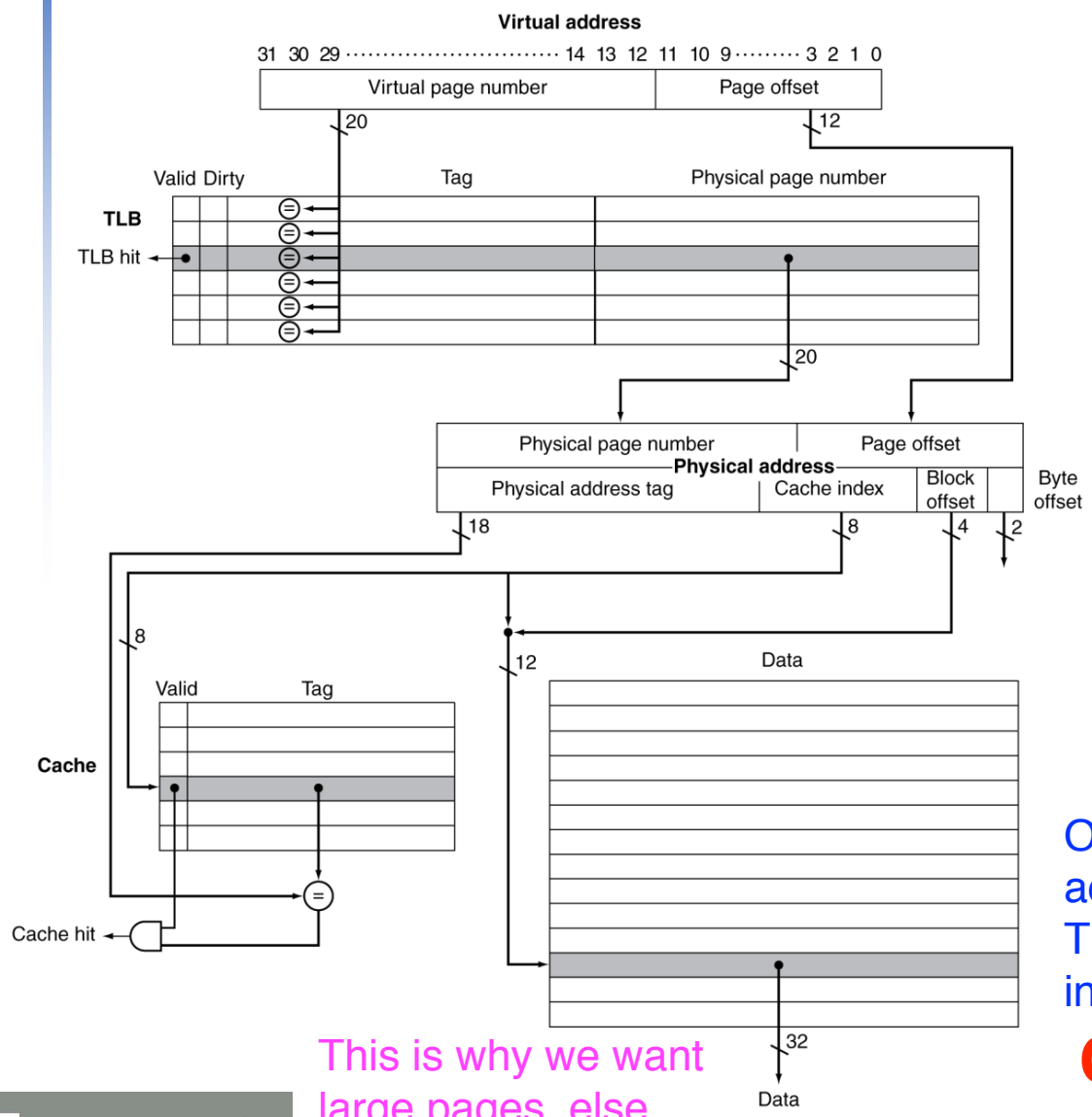
- To reduce page fault rate, prefer least-recently used (LRU) replacement
  - Reference bit (aka use bit) in PTE set to 1 on access to page
  - Periodically cleared to 0 by OS
  - A page with reference bit = 0 has not been used recently
- Disk writes take millions of cycles
  - Block at once, not individual locations
  - Write through is impractical
  - Use write-back
  - Dirty bit in PTE set when page is written

Fast Translation Using a TLB



TLB = Translation Look-aside Buffer (a cache of Page-Table entries)

TLB and Cache Interaction



- If cache tag uses physical address
  - Need to translate before cache lookup
- Alternative: use virtual address tag
  - Complications due to aliasing
    - Different virtual addresses for shared physical address

Often we want: physical addr. cache, and TLB access in parallel with tag read from cache. This requires cache index to be fully contained in page offset bits, which means:

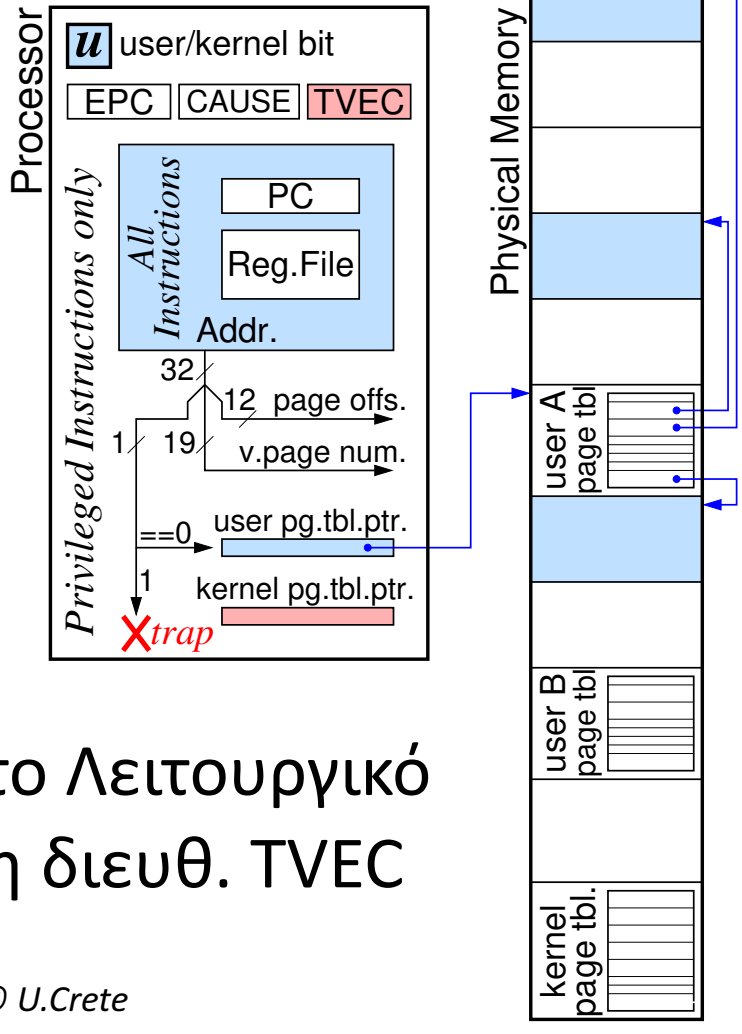
Cache Way Size ≤ Page Size

This is why we want large pages, else forced to increase associativity of L1

Εκτέλεση Διεργασίας Α, σε User Mode

Σε *User mode* *απαγορεύονται*:

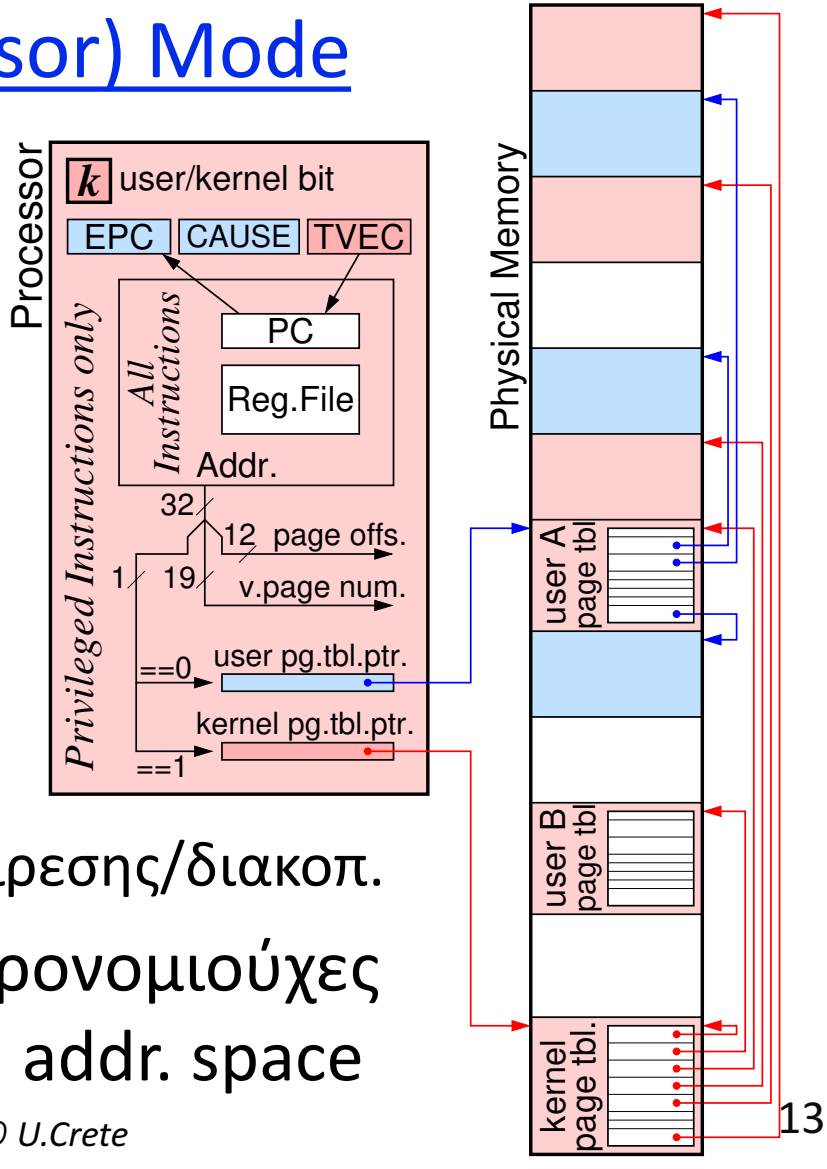
- Εκτέλεση «Προνομιούχων» εντολών (privileged instruct.)
- Προσπέλαση με MS bit εικονικής διεύθυνσης == 1
- Απόπειρα παραβίασης, ή page fault, ή ecall επιφέρει:



⇒ *Exception/trap*: είσοδος στο Λειτουργικό (kernel/supervisor mode) στη διευθ. TVEC

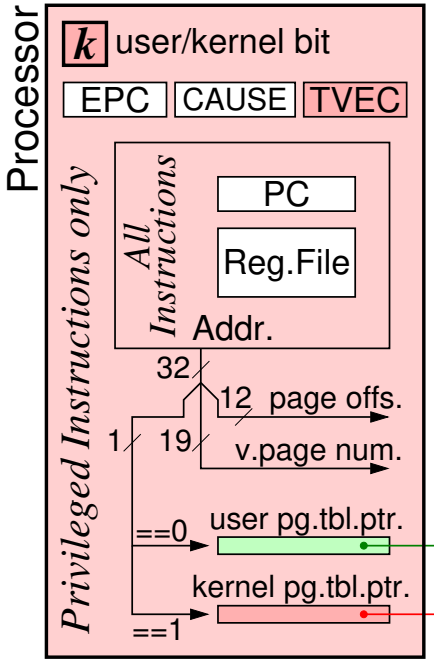
Είσοδος σε Kernel (Supervisor) Mode

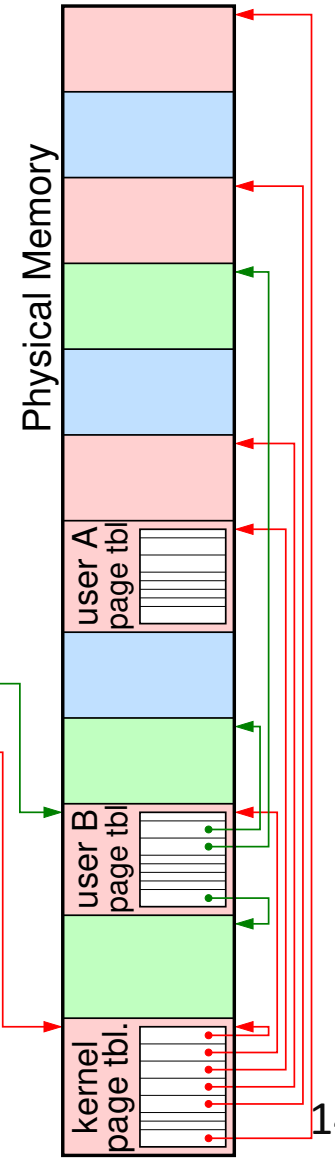
- Αποθήκευση του PC της διακοπείσας εντολής στον καταχ. EPC (exception PC)
- νέος PC = TVEC (trap vector)
  - τον έχει ορίσει το Λειτουργικό
  - απαγορεύεται είσοδος αλλού!
- set *Kernel mode* (& save prev. mode)
- CAUSE reg. ← κωδικός αιτίας εξαίρεσης/διακοπ.
- Σε *kernel mode* επιτρέπονται προνομιούχες εντολές και προσπελ. σε kernel addr. space



Προετοιμασία αλλαγής σε Διεργασία Β

- Όταν τρέχει το Λειτουργικό (σε kernel mode, με MS addr. bit == 1), βλέπει και τον χώρο διευθύνσεων του προηγούμ. ή επόμενου χρήστη, προκειμένου να παίρνει ή δίνει ορίσματα, δεδομένα, κλπ.

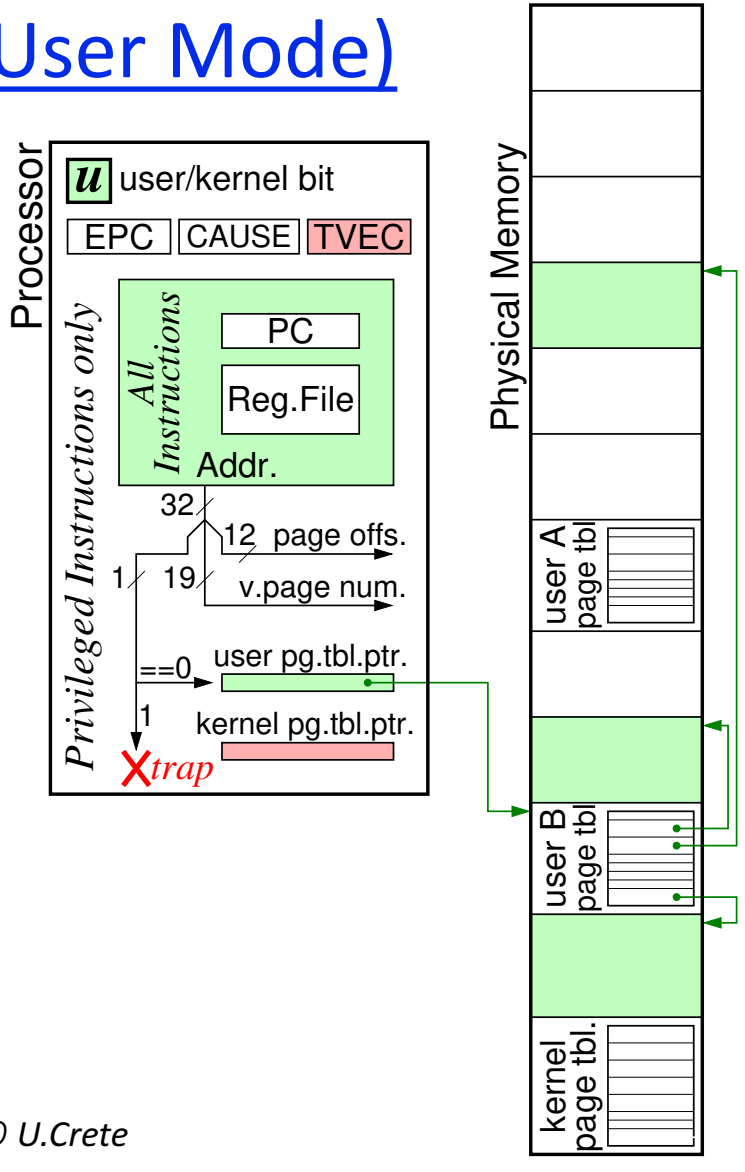




- Στο σχήμα, το Λειτουργικό ετοιμάζεται να αλλάξει στην διεργασία χρ. Β:
  - αλλαγή του User Page Table Pointer
  - επαναφορά καταχωρητών του χρήστη Β

Επιστροφή σε Διεργασία Β (User Mode)

- Set User mode
- Επαναφορά του PC σε ό,τι είχε σωθεί στον EPC όταν είχε διακοπεί η διεργασία Β
- Μόνον οι σελίδες που περιέχονται στον page table της διεργασίας Β είναι τώρα ορατές από τον επεξεργαστή



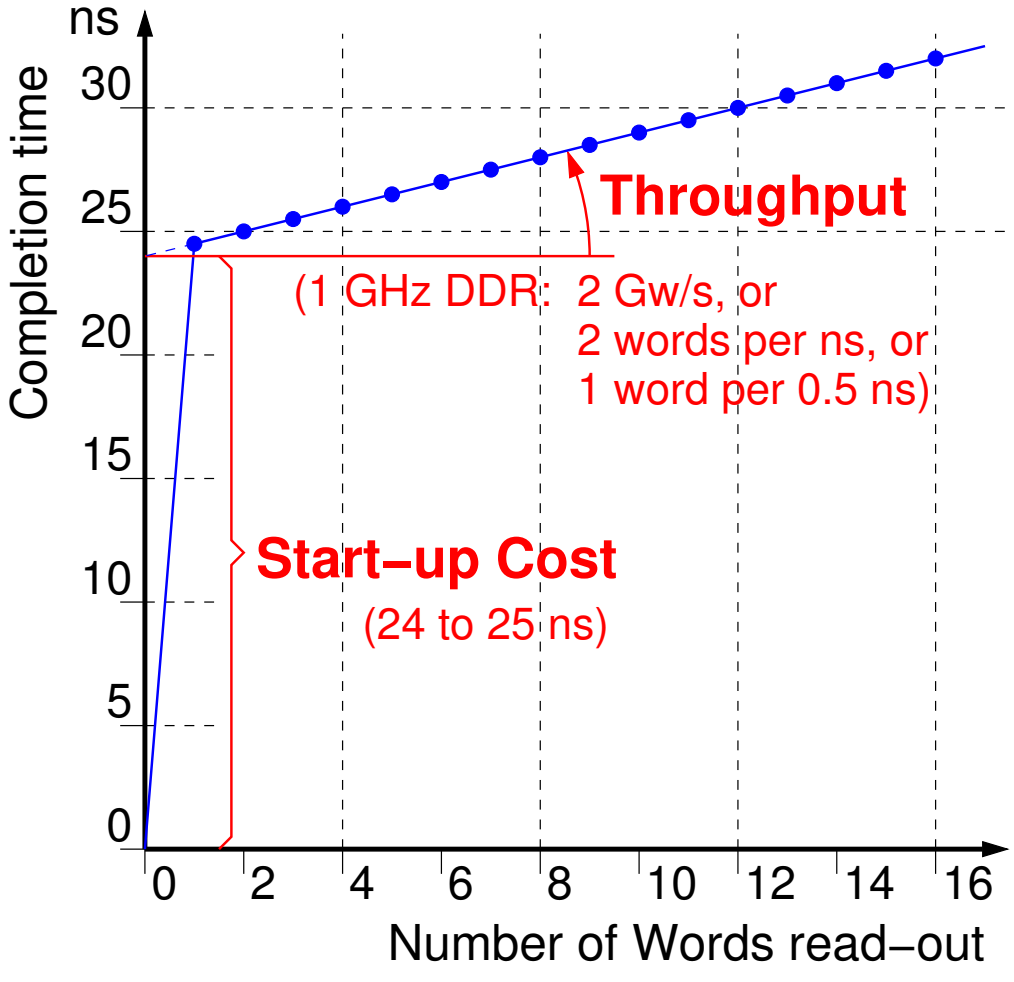
Διακοπές (Interrupts) και Εξαιρέσεις (Exceptions)

- *Interrupt*: γεγονός υψηλότερης προτεραιότητας (π.χ. περιφερειακή συσκευή) έχει ανάγκη να χρησιμοποιήσει τον επεξεργαστή, και γι' αυτό διακόπτει την διεργασία χαμηλότερης προτεραιότητας που τρέχει εκεί τώρα
- *Exception*: κάτι πήγε στραβά στην τρέχουσα διεργασία, και η εκτέλεσή της δεν μπορεί να συνεχιστεί ως έχει, αλλά πρέπει να διακοπεί, είτε τελεσίδικα, είτε προσωρινά μέχρι διόρθωση και επανεκκίνηση
- Παρόμοιος μηχανισμός και γιά τις δύο περιπτώσεις, ανεξαρτήτως αιτίας

Διεύθυνση Εισόδου, Vectored Interrupts

- *TVEC* (Trap Vector (ή "Supervison Trap Vector" – STVEC)): Διεύθυνση εισόδου στο Λειτουργικό, όχι υπό τον έλεγχο του χρήστη (ώστε να μη μπορεί αυτός να παρακάμπτει τους ελέγχους ασφαλείας), π.χ. από ειδικό καταχωρητή που το Λειτουργικό Σύσ. θέτει από πριν
- *CAUSE* Register (ή "Supervisor Exception Cause" – SCAUSE): κωδικοποιημένη πληροφορία γιά το ποιά η αιτία της διακοπής ή εξαίρεσης
- *Vectored Interrupts* – εναλλακτικός τρόπος συνδυασμού των δύο: διαφορετική διεύθυνση εισόδου γιά την κάθε διαφορετική αιτία
  - κάτι σαν Switch Statement βάσει αιτίας γιά είσοδο στο Λειτουργικό
  - πλεονέκτημα: γλιτώνουμε να το κάνουμε σε software
  - μειονέκτημα: κοινός κώδικας στην αρχή, σε όλες τις περιπτώσεις, γιά σώσιμο καταχωρητών κλπ. προετοιμασίες, κι *ύστερα* το Switch Stmt. βάσει αιτίας

Κόστος Εκκίνησης – Παροχή (Οικονομία Κλίμακας)



Παράδειγμα DRAM:

- σημαντικό κόστος εκκίνησης
- πολύ μικρό κόστος για γειτονικές λέξεις από ίδια γραμμή

Παράδειγμα Pipeline:

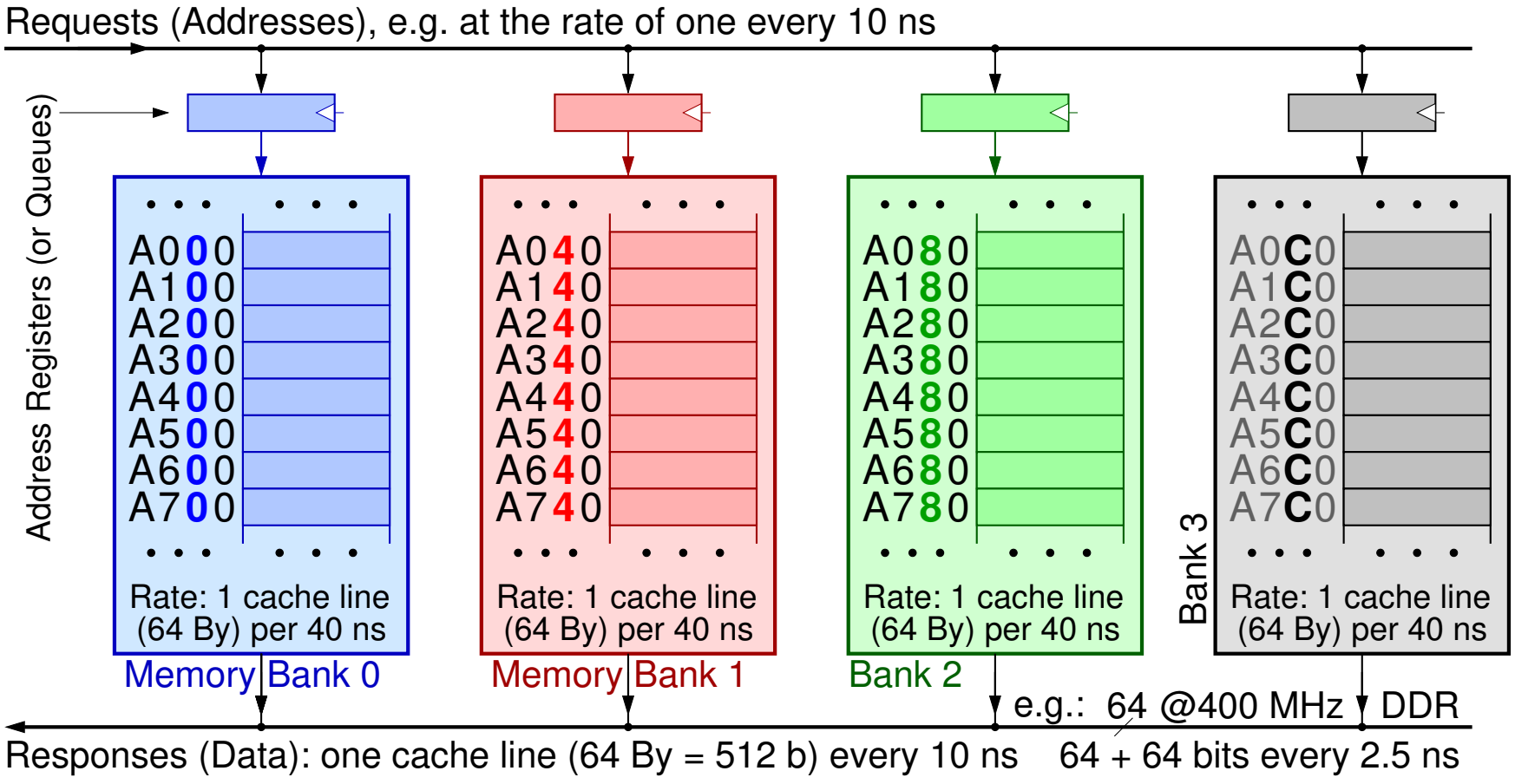
- κόστος εκκίνησης = 4
- Παροχή (ρυθμός ολοκλήρωσης) = 1 εντολή/κύκλο ρολ.

⇒ Απόσβεση Κόστους μέσω αύξησης του μεγέθους block της εργασίας

Ομοίως και τα Δίκτυα και οι γρήγορες Αρτηρίες

- Παράδειγμα: Οπτική ίνα 2 km, 100 Gbits/s
  - Ταχ. φωτός (διηλεκτρ.) ≈ 200 Mm/s ⇒ t_d πρώτου bit ≈ 10 μs
  - Παροχή = 100 Gb/s ≈ 10 GBy/s = 10 By/ns = 1 By ανά 0.1 ns
    - απλοποιητική παραδοχή: 8b/10b line encoding (σήμερα: 64b/66b)
- Παράδειγμα: Αρτηρία σαν PCIe (x1), 20 cm, 25 Gb/s
  - t_d πρώτου bit @ 200 Mm/s ≈ 1 ns
  - Παροχή ≈ 2.5 GBy/s = 2.5 By/ns = 1 By / 0.4 ns (απλοπ.: old encoding)
  - Εάν Packet Header = 12 Bytes (διευθύνσεις, format, size, κλπ.) ⇒ t_d πρώτου data ("payload") Byte = 1 (xmit del.) + 4.8 (hdr) ≈ 6 ns
  - Εάν 64 By payload ⇒ 6 (startup cost) + 26 (payload) = 32 ns total
  - Εάν 4 KBy payload ⇒ 6 (startup) + 1640 (payload) ≈ 1.65 μs total

Παραλληλισμός Μνήμης: Διαφύλλωση (Interleaving)



Πολλαπλές μνήμες σε παράλληλη λειτουργία, διασπορά δεδομένων μεταξύ τους

I/O Instructions versus Memory-Mapped I/O

Έχουν υπάρξει στο παρελθόν ειδικές εντολές:

- *Input*: ανάγνωση πληροφορίας από εξωτερική πηγή
- *Output*: αποστολή πληροφ. σε εξωτερ. προορισμό

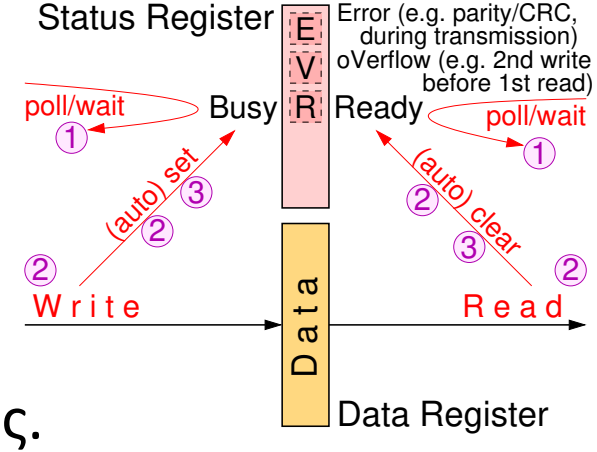
Σήμερα συνήθως: *Memory-Mapped Input/Output (I/O)*
(Απεικόνιση Μνήμης των Μονάδων Εισόδου/Εξόδου (Ε/Ε)):

- *Load*: ανάγνωση όπως από μνήμη, αλλά από εξωτ. πηγή
- *Store*: όπως εγγραφή σε μνήμη, αλλά σε εξωτερ. προορ.
- Διαφοροποίηση Μονάδων Ε/Ε από κανον. μνήμη βάσει Διεύθυνσης: τμήμα του χώρου (φυσικών) διευθύνσεων αφιερωμένο σε συσκευές επικοινωνίας, αντί μνήμης

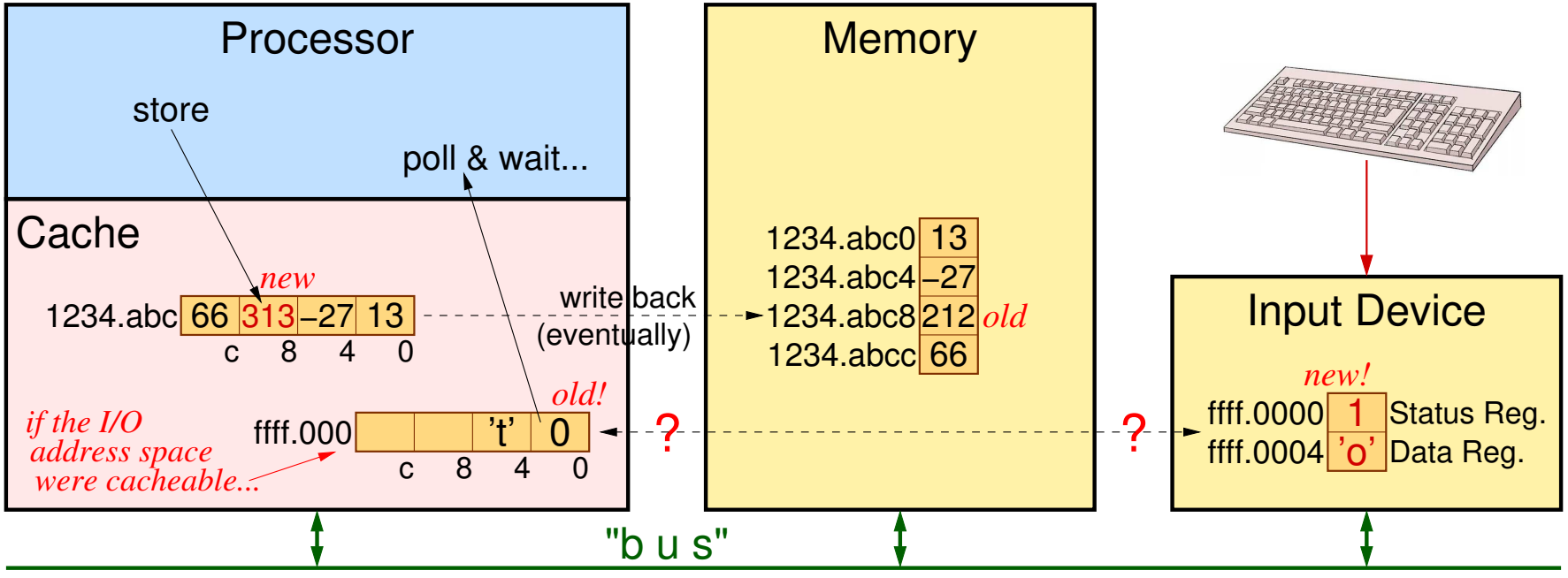
Προσβ. μέσω OS, για Προστασία!

Polling (Δειγματοληψία), Busy-Wait και εναλλακτικές

- Polling («δειγματοληψία», ή «περιόδευση»):
  - Ο τρόπος αναμονής για νέα δεδομένα εισόδου, ή για δυνατότητα επόμενης εξόδου, μέσω επανειλημμένης ανάγνωσης του Status Register – Ready/Busy bit, μέχρις αυτό να γίνει έτοιμο
- Busy-Wait:
  - Συνεχής ανάγνωσή του, που δεν αφήνει τίποτα άλλο χρήσιμο να γίνει, μέχρις ότου το Ready bit δείξει ετοιμότητα
  - Εναλλακτικά: το διαβάζουμε περιοδικά, ενώ ενδιάμεσα κάνουμε και άλλες εργασίες. Για εξασφάλιση ότι ποτέ δεν θα αργήσουμε υπερβολικά, π.χ.: σε κάθε interrupt από το real-time clock (κάθε ~10-20 ms)



Ο χώρος διευθ. I/O πρέπει να είναι Non-Cacheable

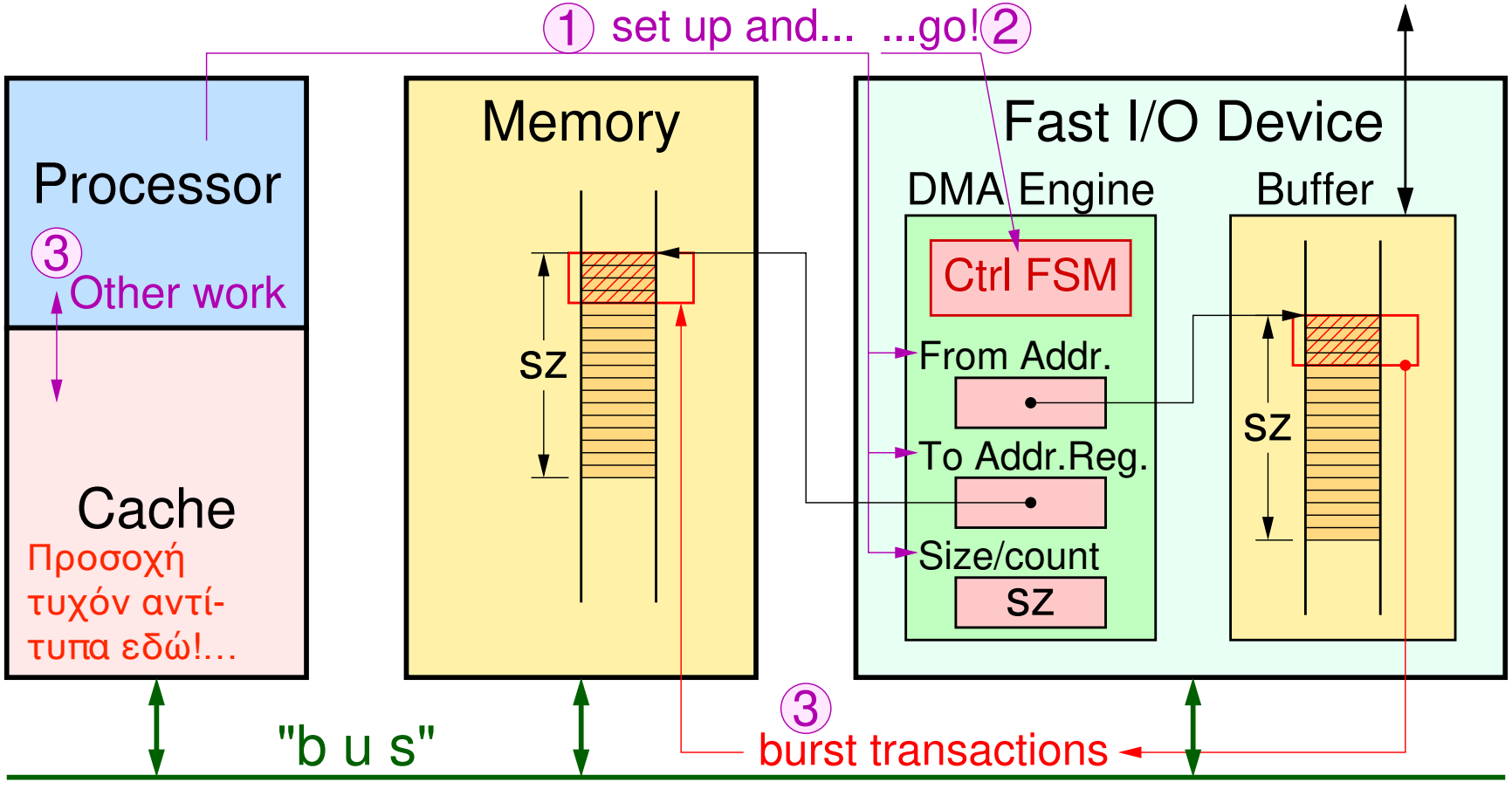


- Όποτε θέσεις «μνήμης» προσπελάζονται από πολλαπλούς παράγοντες (I/O ή πολύεπεξεργ.), τυχόν αντίγραφα σε κρυφές μνήμες απαιτούν ειδική φροντίδα
- Εδώ: Σελίδες non-cacheable (απαγορεύεται το caching). Αλλού: Συνοχή Κρ. Μν.
- Σημειώστε: το write-through είναι ημίμετρο – μόνο αποστολή, όχι ανάγνωση

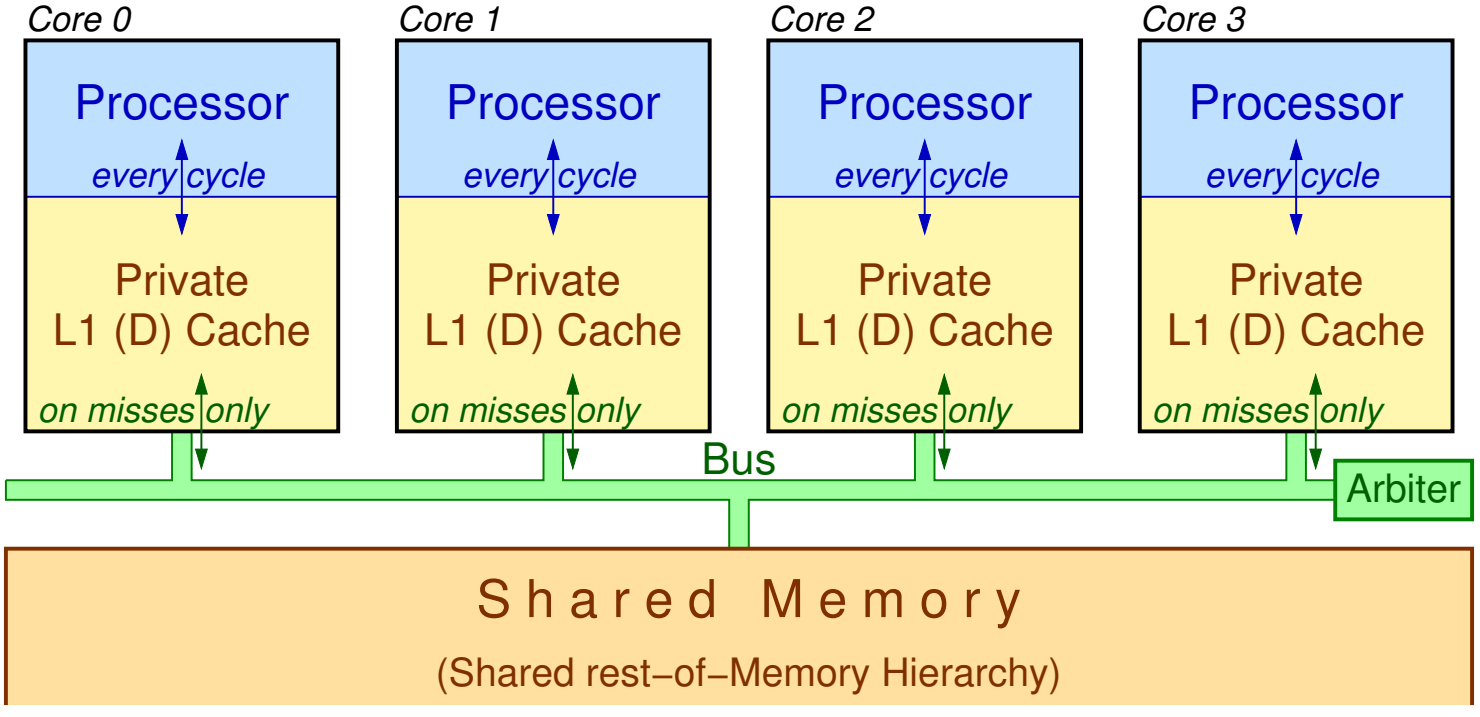
Interrupt-driven I/O (Ε/Ε βάσει διακοπών)

- Το Polling συχνά είναι ασύμφορα χρονοβόρο
  - ιδιαίτερα όταν ρυθμός άφιξης ποικίλλει ευρέως (απρόβλεπτοι χρόνοι)
  - ανάγνωση I/O Status reg. 10-100'δες κύκλους ρολογ. (non-cacheable!)
- I/O Interrupts: ο συνήθης εναλλακτικός μηχανισμός
  - Interrupt (Διακοπή) παρόμοια με Exception (Εξαίρεση) – §12.3
  - Ασυσχέτιστη με το πρόγραμμα που τρέχει όταν αυτή συμβαίνει
  - Δεν υπάρχει ανάγκη ακύρωσης καμιάς εντολής στην pipeline
  - Απλός μηχανισμός: ένα σύρμα ανά περιφερειακή συσκευή, το OR όλων τους προκαλεί την αποθήκευση & αλλαγή του PC κλπ.
  - Συχνά με προτεραιότητες ⇒ ενδεχόμενη συσσώρευση διακοπών χαμηλότερης προτεραιότητας ⇒ προσοχή στον καταχ. CAUSE

Direct Memory Access (DMA)

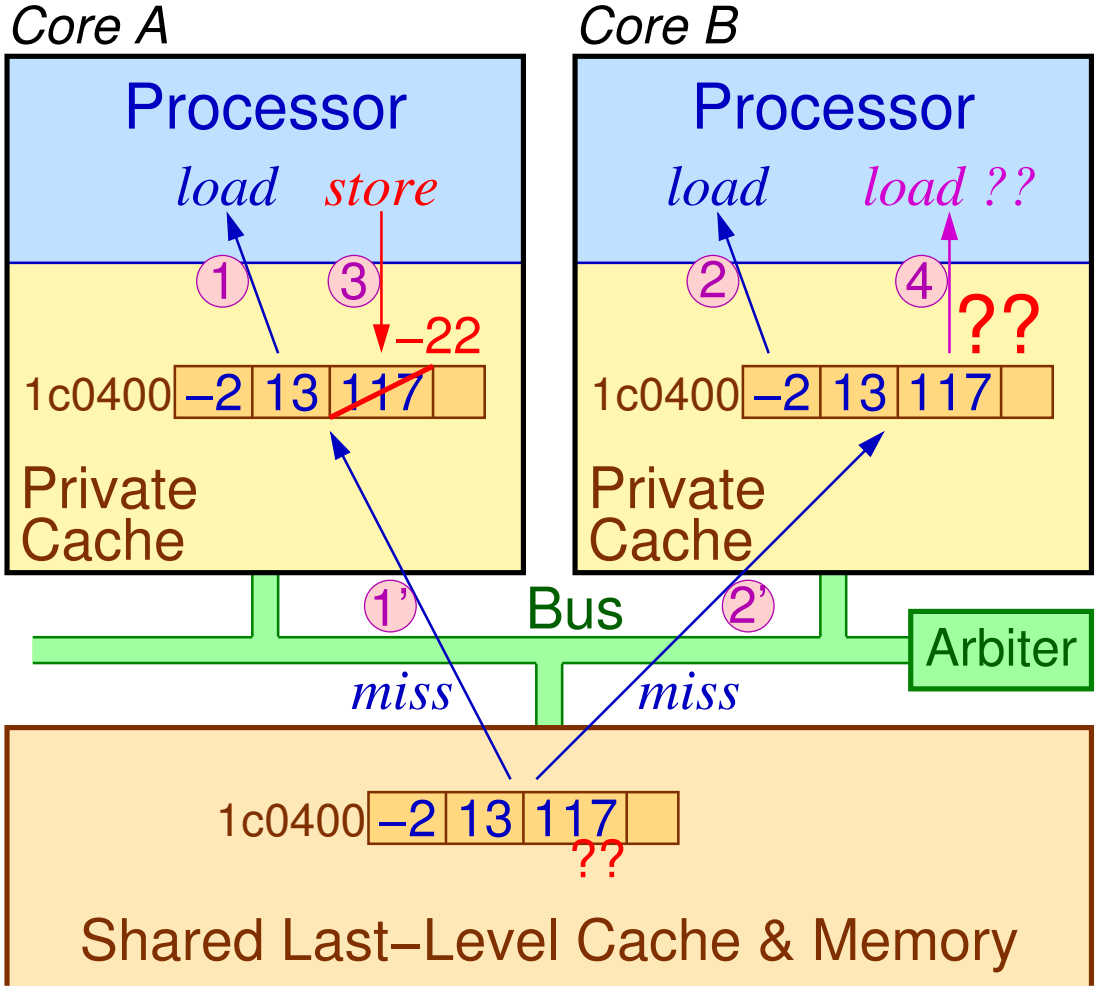


Απλό παράδειγμα Πολυπύρηνου Επεξεργαστή



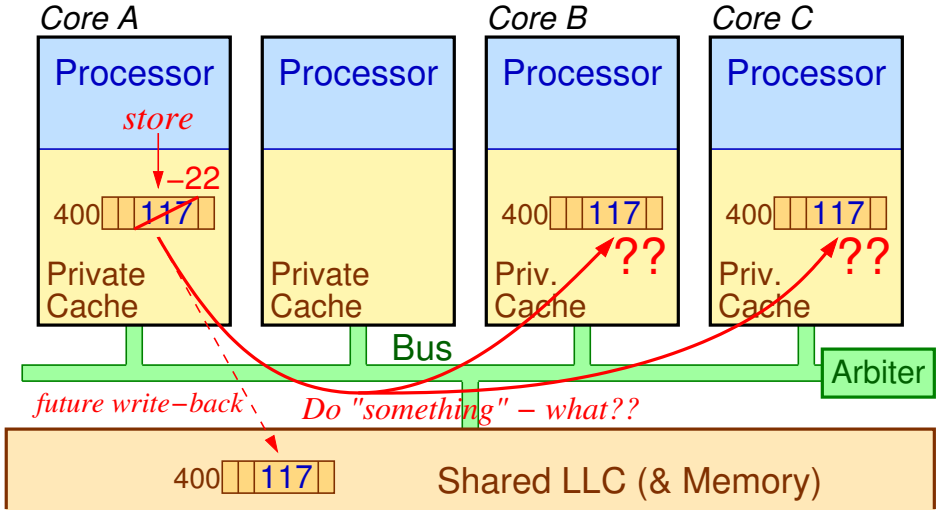
- Επικοινωνία/συνεργασία μέσω κοινόχρηστης μνήμης
  - για να αρκεί ένα κοινόχρηστο Bus για τις συνολικές αστοχίες: έως ≈8 επεξεργαστές (ή ≈16 εάν και οι L2 caches είναι private)

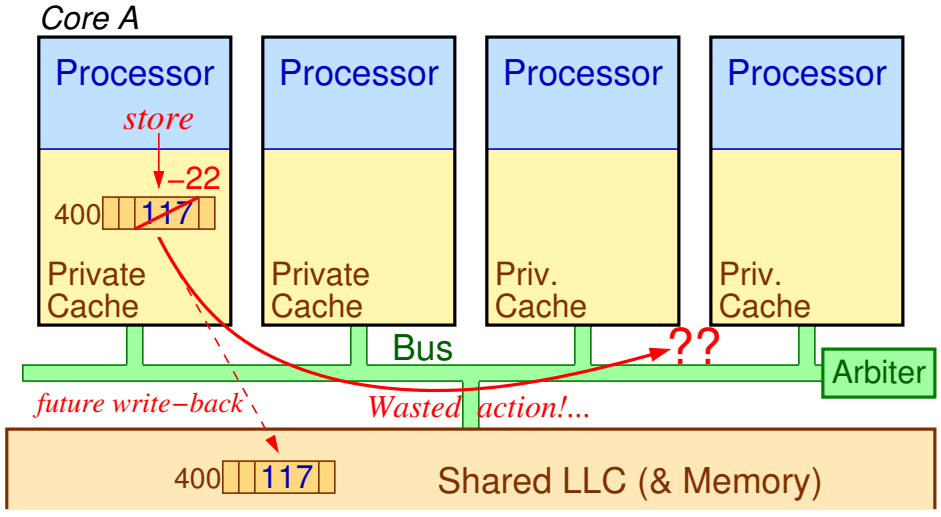
Το πρόβλημα της Συνοχής (Coherence) Κρυφών Μν.



- Πολλαπλά αντίτυπα της ίδιας πληροφορίας
- Όταν το ένα αλλάζει, τα άλλα τι κάνουν;
- Προς Μνήμη/LLC προ-ντίζει το write-back
- Προς άλλες Caches??
- Ξέρει ο Α ότι και άλλες caches έχουν αντίτυπα της γραμμής 1c0400 ?
- Να τα ενημερώσει ή να τα ακυρώσει;

Για κάθε cache line μου, ξέρω εάν την έχουν και άλλοι;

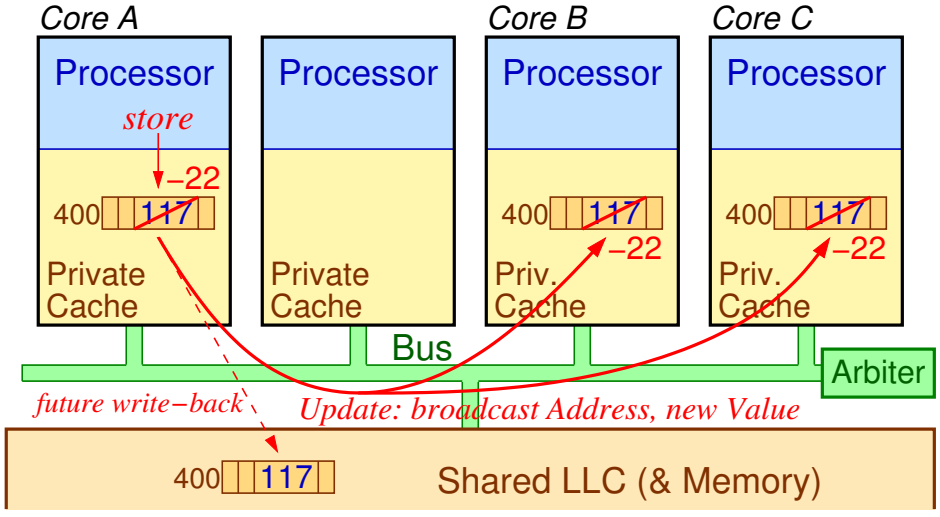


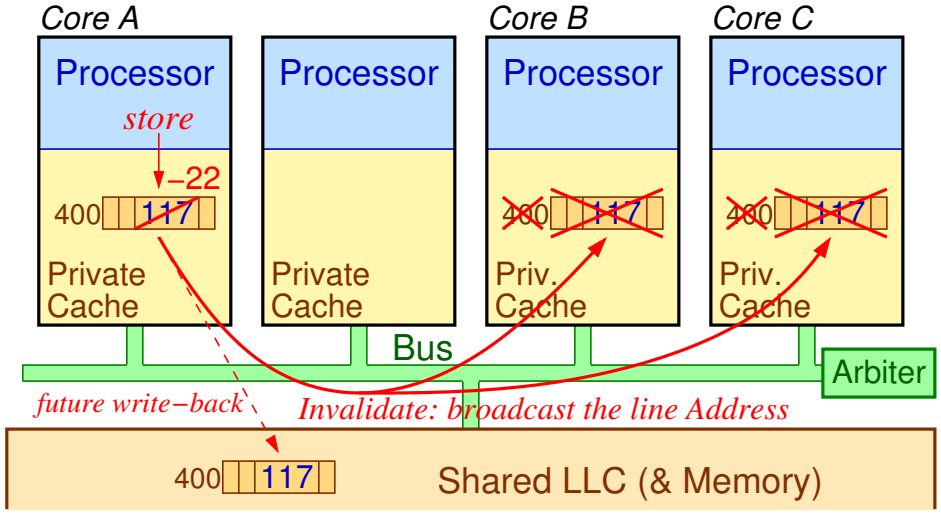


- Εάν υπάρχει η πληροφορία ότι η cache line 400 είναι "*Shared*" ⇒
- ή να ενημερώσουμε
  ή να ακυρώσουμε
  τα υπόλοιπα αντίτυπά της

- Εάν όμως ξέρουμε ότι έχουμε την cache line 400 "*Exclusive*" ⇒
- θα ήταν σπατάλη να απασχολούμε το Bus άνευ λόγου

Write-Update versus Write-Invalidate





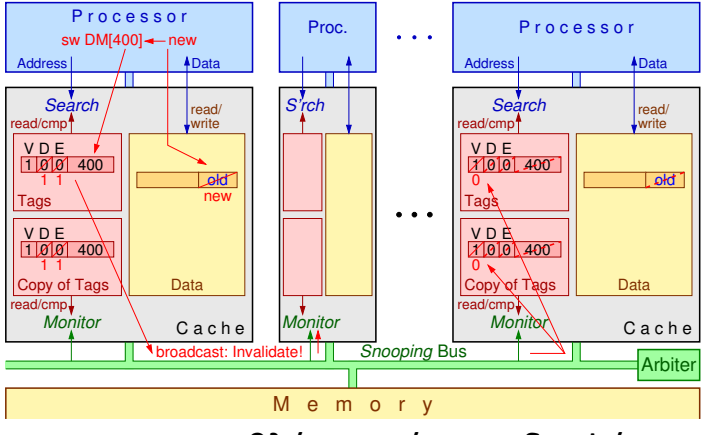
- Write to Shared: **Update** all others
- Όπως το write-through, μεταφέρει λέξη-λέξη, αντί ολόκληρη γραμμή
- Δεν το προτιμάμε: συχνά η χρήση νέων data από άλλους επεξ. αργεί

- Write to Shared: **Invalidate** others
- «Ας αποτελειώσω εγώ τη δουλειά μου τοπικά, και μετά όποιος άλλος τα θέλει ας τα ξαναζητήσει»
- Το συνηθισμένο πρωτόκολλο

Snooping Coherence – Συνοχή μέσω Παρακολούθησης

- *Snoop*: Παρακολουθώ, κρυφοκοιτάζω, κατασκοπεύω
- Κοινόχρηστο Bus, που όλες οι Κρυφές Μνήμες το παρακολουθούν, και όπου όλες διαλαλούν (broadcast) όλα όσα πρέπει οι άλλες να ξέρουν
  - Το απλούστερο πρωτόκολλο Συνοχής Κρυφών Μνημών
  - Αποδοτικό για έως περίπου 8 κρυφές μνήμες – πέραν τούτου η κίνηση γίνεται απαγορευτικά πολλή
  - Ο Διαιτητής (Arbiter) ορίζει την καθολική σειρά των γεγονότων (Memory Consistency – Συνέπεια)
  - Tags: δύο αντίγραφα ⟶ δίπορτη ανάγνωση

Snoopy, by Charles M. Schulz



→ βλέπε επόμενη διαφάνεια

Κατάσταση (State) κάθε Γραμμής: ορολογία "MOESI"

Guaranteed Exclusive

it is guaranteed that my copy is the only copy currently existing in any cache

Potentially Shared

other caches may have copies of this line (not known for sure)

Obliged to Write–back

my copy is up to date;
the memory version is outdated;
I am responsible to write back to memory, and also to provide this up to date version to other caches

Free to Evict

the memory or another cache have the up to date version, and I have a copy of that, which I am free to evict at any time

Invalid

nothing in this cache line

- Επέκταση των Valid-Dirty (per line) bits: τι ξέρω γιά την γραμμή αυτή

Dynamic Multiple Issue

- "Superscalar" processors
- CPU decides whether to issue 0, 1, 2, … each cycle (checks dependencies and)
  - Avoiding structural and data hazards
- Avoids the need for compiler scheduling
  - Though it may still help
  - Code semantics ensured by the CPU

Allows executables to run on newer processors, with same ISA but different pipeline, without needing to be recompiled

Dynamic Pipeline Scheduling

- Allow the CPU to execute instructions out of order to avoid stalls
  - But commit result to registers in order
- Example

```
ld   x31,20(x21)
add  x1,x31,x2
sub  x23,x23,x3
andi x5,x23,20
```

  - Can start sub while add is waiting for ld

Out-of-Order (ooo) Execution

In-Order Commit

(so as to flush results of mis-speculated instructions, and also allow precise exceptions)

Multithreading

One "thread of control" = one (traditional) sequential program.
Multiple threads = parallel program.

- Performing multiple threads of execution in parallel
  - Replicate registers, PC, etc.
  - Fast switching between threads
- Fine-grain multithreading
  - Switch threads after each cycle
  - Interleave instruction execution
  - If one thread stalls, others are executed
- Coarse-grain multithreading
  - Only switch on long stall (e.g., L2-cache miss)
  - Simplifies hardware, but doesn't hide short stalls (eg, data hazards)

mimic multiple cores, thus:

but Share the Functional Units and the Caches