Πολυπύρηνοι Επεξεργαστές (Multicore Proc.), Κοινόχρηστη Μνήμη (Shared Memory), Συνοχή Κρυφών Μνημών (Cache Coherence)

14α (§14.1-3) – 13-15 Μαΐου 2024 – Μανόλης Κατεβαίνης

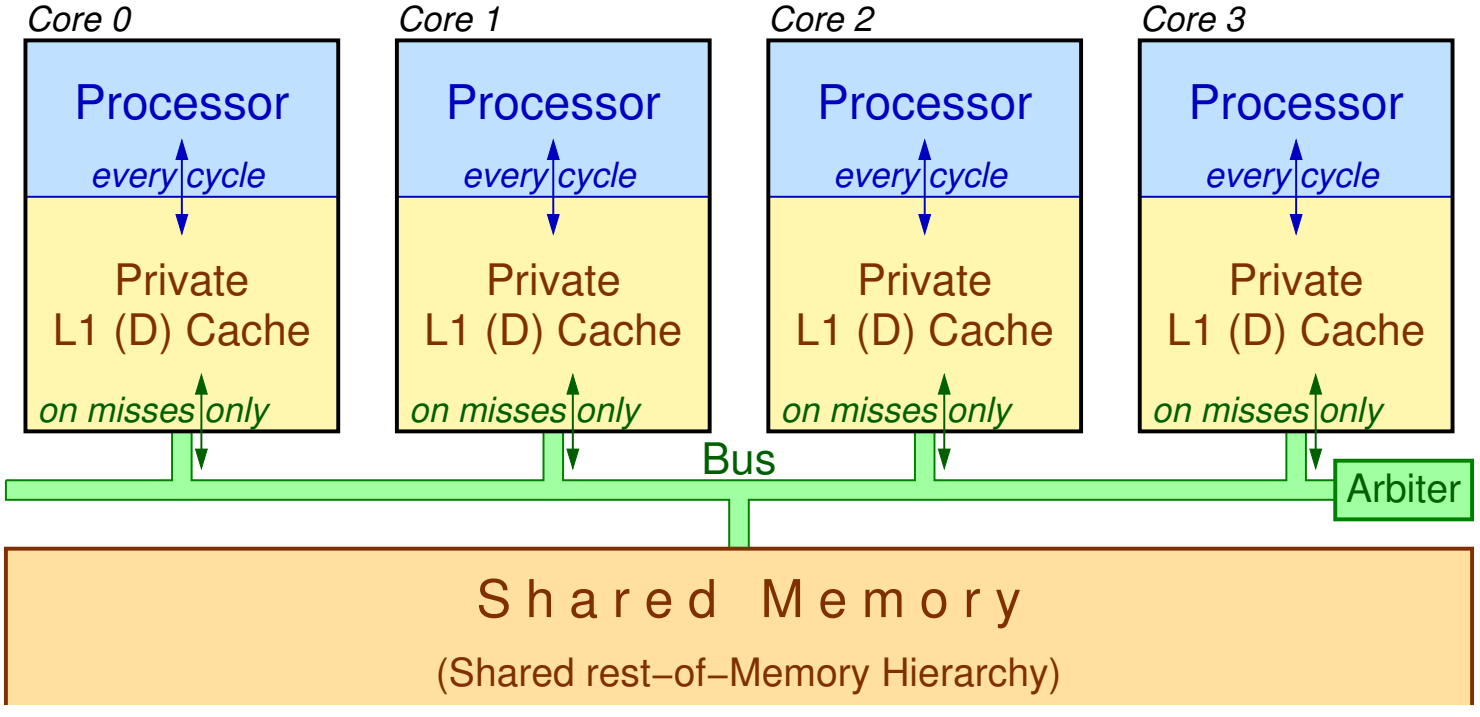
Πολυεπεξεργαστές (Multiprocessors)

- Πολλαπλοί Επεξεργαστές, για αξιοποίηση Παραλληλισμού
- είτε Ανεξάρτητες Διεργασίες (Processes) – το απλούστερο
- ή Παράλληλα Προγράμματα – Task Parallelism
  - είτε συνεργαζόμενες (επικοινωνούσες) Διεργασίες
  - ή πολλαπλά Νήματα (Threads) ελέγχου μέσα στην ίδια διεργασία
  - *"Thread-Level Parallelism* (TLP)", σε αντιδιαστολή προς "*Instruction-Level Parallelism* (ILP)" σε pipelines (esp. out-of-order)
- Συνεργασία (Επικοινωνία) μέσω:
  - είτε Κοινόχρηστης Μνήμης (Shared Memory)
  - ή Ανταλλαγής Μηνυμάτων (Message Passing)

Πολυπύρηνοι Επεξεργαστές (Multicore Processors)

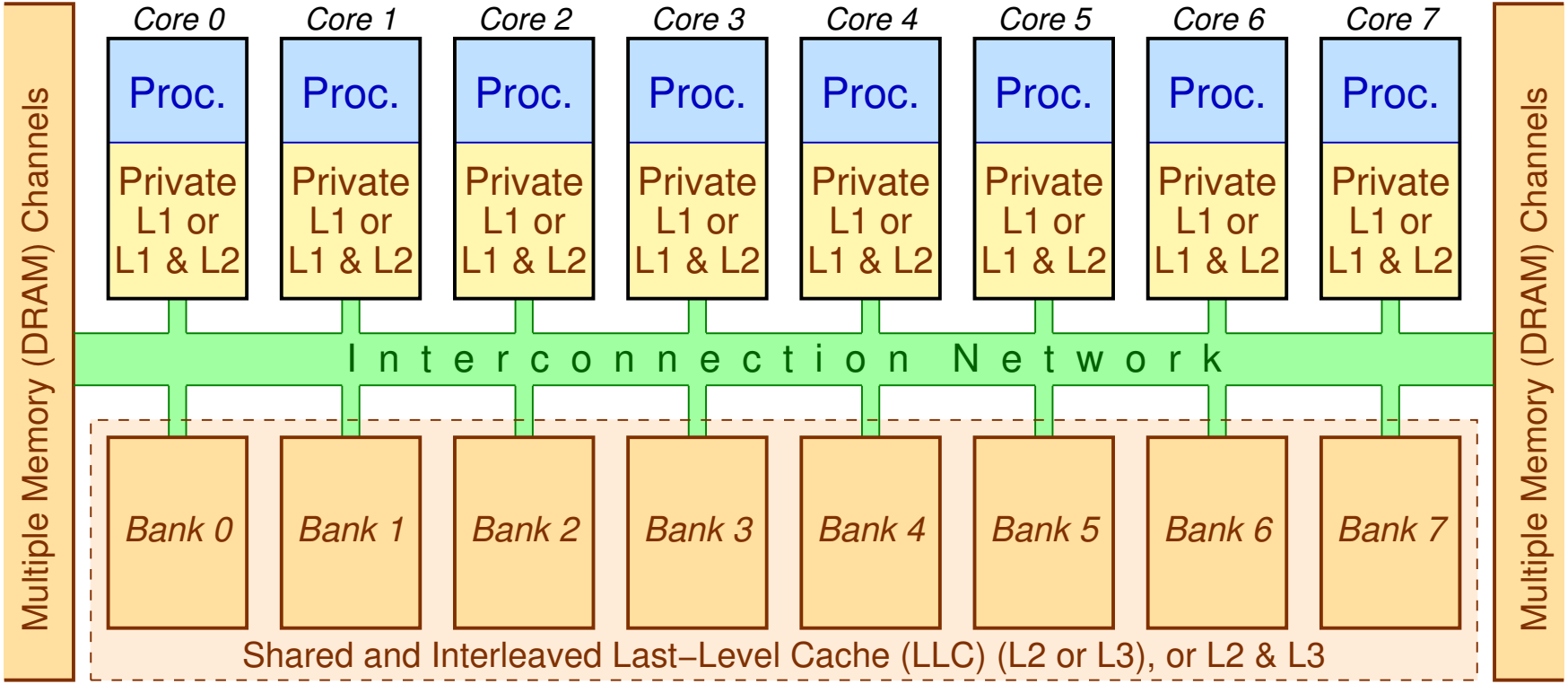
- Δύο ή περισσότεροι επεξεργαστές στο ίδιο chip
  - "*Multicore*": συνήθως 2 έως 8 (γρήγοροι) επεξεργαστές/chip
  - "*Manycore*": συνήθως δεκάδες (απλούστεροι) επεξεργ./chip
- Ορολογία: *IP Core* = Intellectual Property Core
  - Οι κατασκευαστές των System-on-Chip (SoC) αγοράζουν (license) διάφορα IP cores από διάφορους σχεδιαστές, και τα συνενώνουν
  - Ομοίως και οι multicore processors περιέχουν πολλαπλά IP cores
- Επικοινωνία (Συνεργασία):
  - Ένα ή λίγα multicores: σχεδόν πάντα μέσω Κοινόχρηστης Μνήμης
  - Μεγάλα manycores ή πολλά multicores: συχνά μέσω Μηνυμάτων

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)

Περισσότεροι Πυρήνες, Μεγαλύτερη Παροχή

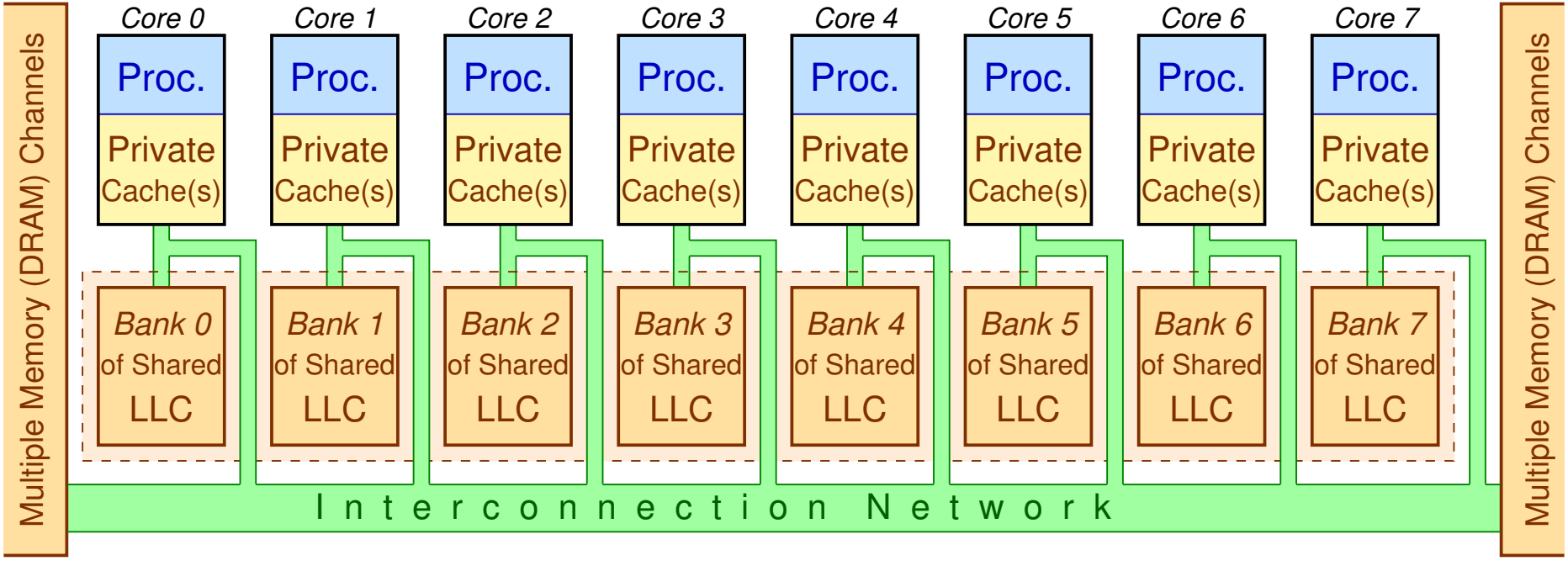


- Διαφύλλωση Κρυφών Μνημών για μεγαλύτερη παροχή (throughput)
- Διασπορά δεδομένων στις Banks: ανά cache-line ή ανά σελίδα;

Uniform Mem. Access (UMA) or Non-Uniform (NUMA)?

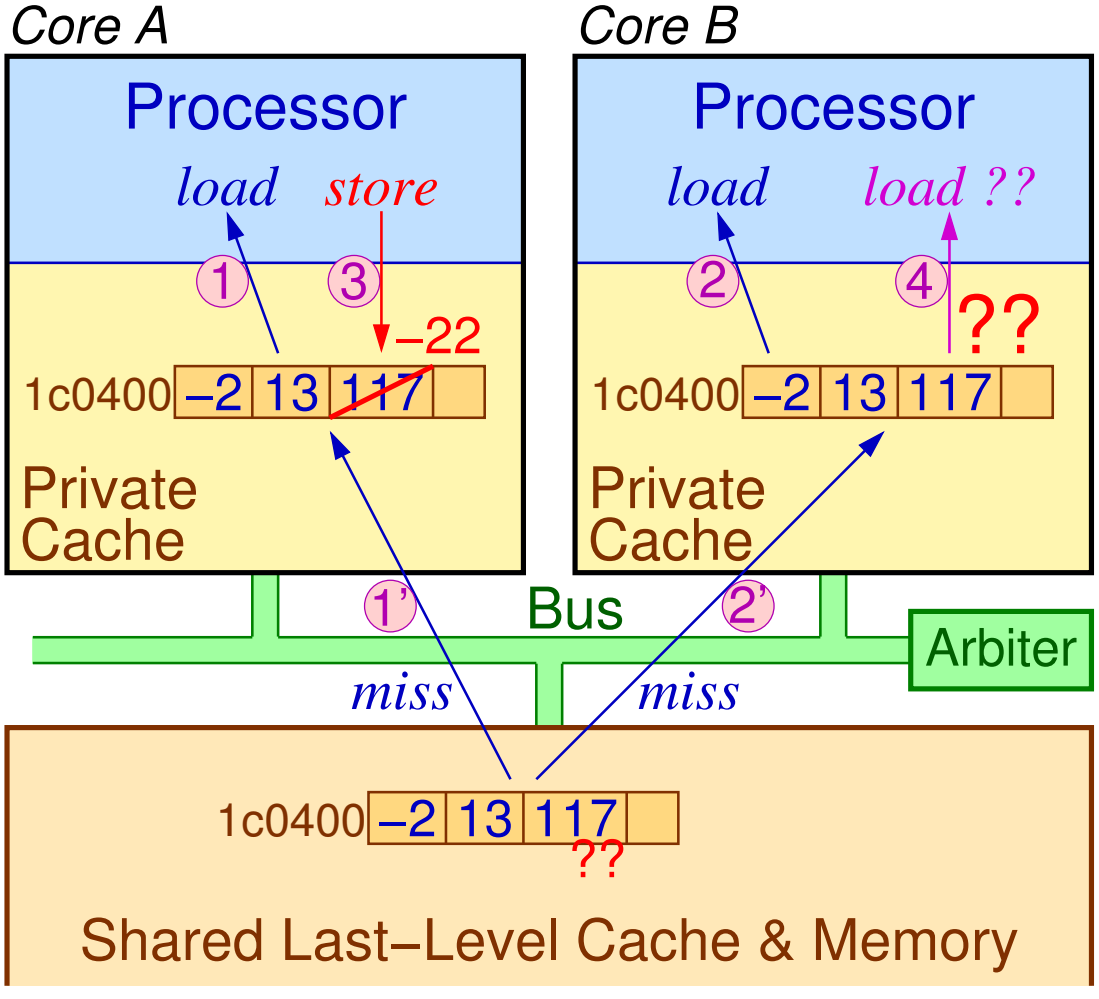
- *UMA* – Uniform Memory Access:
  - Ίδιος χρόνος προσπέλασης (εξυπηρέτησης αστοχίας) σε όλες τις θέσεις μνήμης, ανεξαρτήτως διεύθυνσης
  - Κοινόχρηστη(ες) κρυφή(ές) μνήμη(ες) & κεντρική μνήμη είτε ενιαίες ή ίσες αποστάσεις από όλα τα διαφυλλωμένα Banks
  - Πλεονέκτημα: Απλό στον προγραμματισμό
  - Μειονέκτημα: Μη κλιμακώσιμο για αύξον πλήθος επεξεργαστών
- *NUMA* – Non-Uniform Memory Access:
  - Χρόνος προσπέλασης ποικίλλει με απόσταση/διευθύνσεις μνήμης
  - Αξιοποιεί τοπικότητα, αλλά δυσκολεύει η βελτιστοποίηση
  - Κίνητρο για διασπορά διαφύλλωσης ανά σελίδα, αντί cache line

Non-Uniform Cache Access (NUCA) – NUMA too



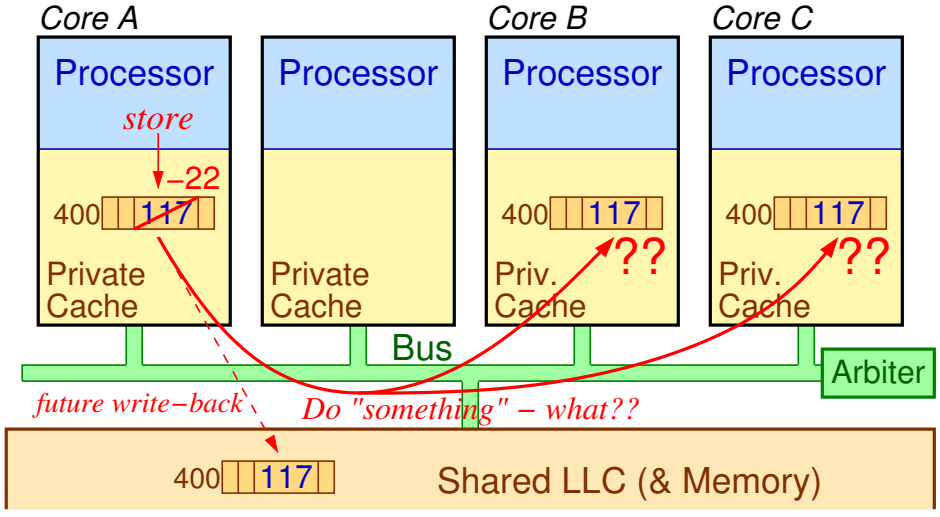
- Ο κάθε επεξεργαστής κοντύτερα σε ένα από τα Banks της κοινόχρηστης κρυφής μνήμης τελευταίου επιπέδου (Last-Level Cache – LLC)
- Ταχύτερη η πρόσβαση στο κοντυνό Bank απ' όσο στα υπόλοιπα

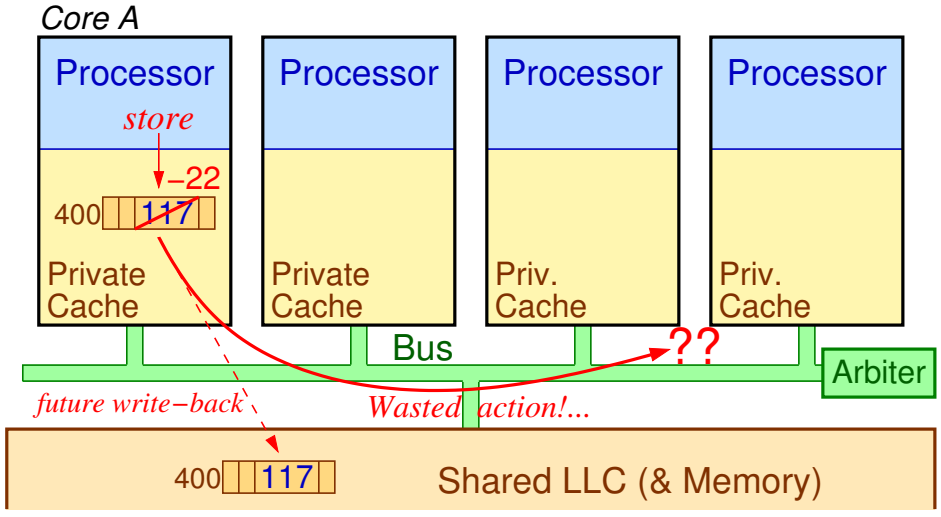
Το πρόβλημα της Συνοχής (Coherence) Κρυφών Μν.



- 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;

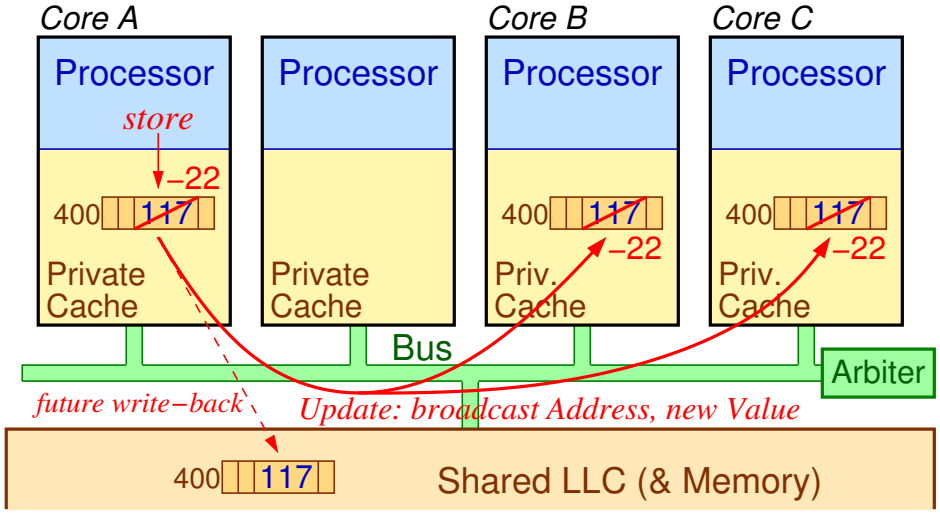


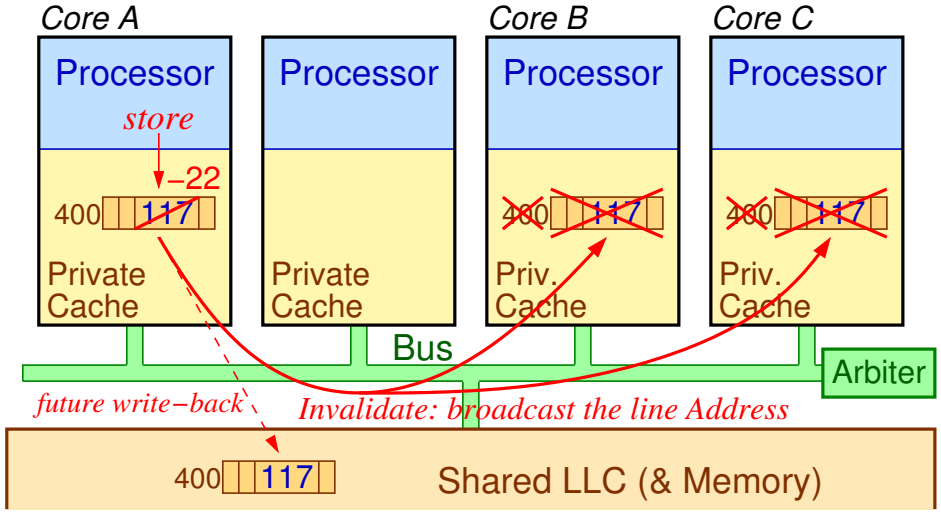


- Εάν υπάρχει η πληροφορία ότι η cache line 400 είναι "*Shared*" ⇒
- ή να ενημερώσουμε
  ή να ακυρώσουμε
  τα υπόλοιπα αντίτυπά της

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

Write-Update versus Write-Invalidate





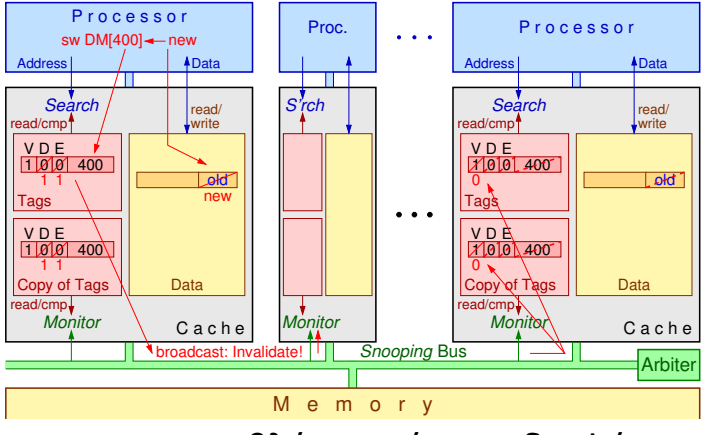
- Write to Shared: **Update** all others
- Όπως το write-through, μεταφέρει λέξη-λέξη, αντί ολόκληρη γραμμή
- Δεν το προτιμάμε: συχνά η χρήση νέων data από άλλους επεξ. αργεί

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

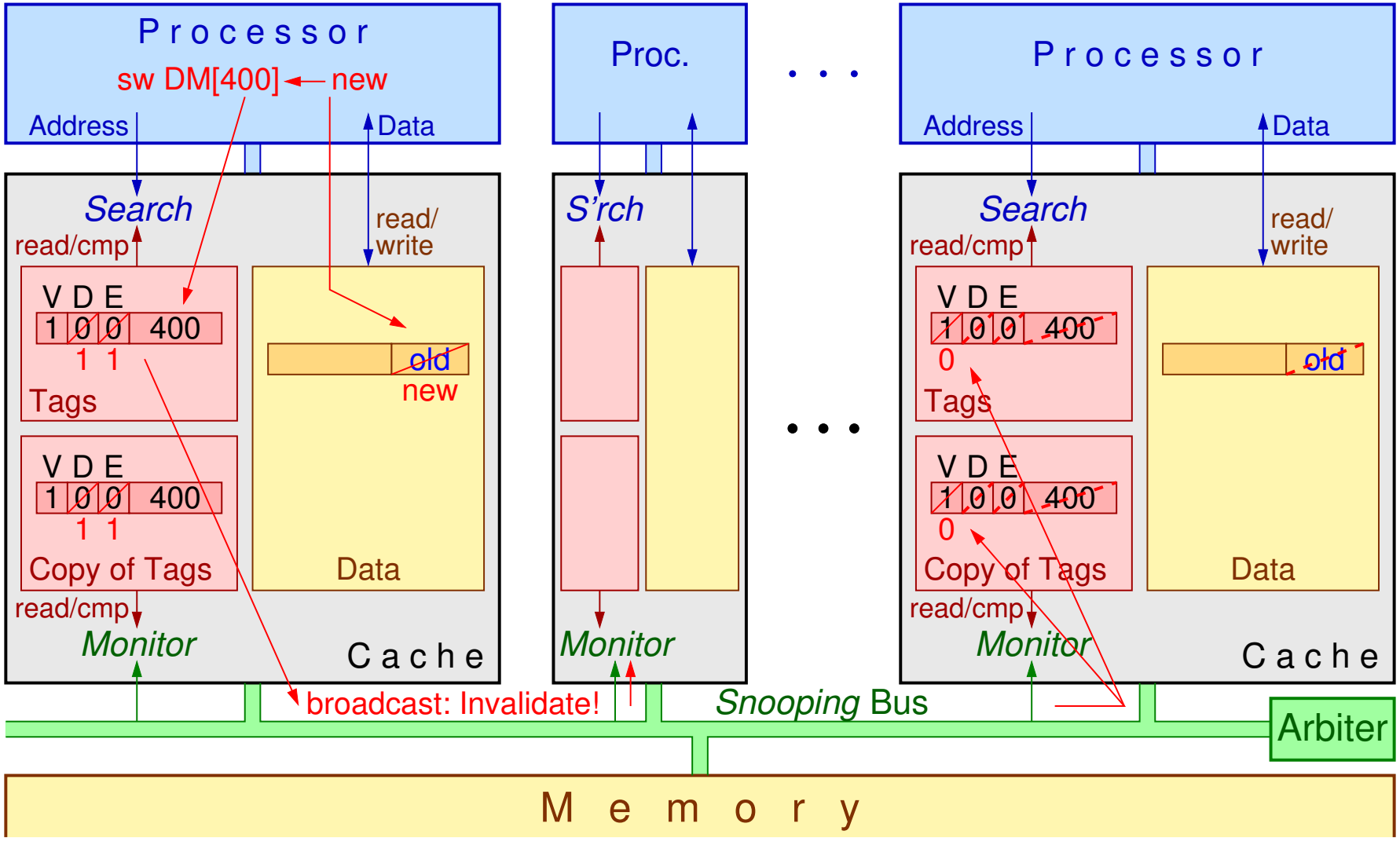
Snooping Coherence – Συνοχή μέσω Παρακολούθησης

- *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


Processor
sw DM[400] ← new
Address
Data
Search
read/cmp
read/
write
V D E
1 0 0 400
1 1
Tags
old
new
V D E
1 0 0 400
1 1
Copy of Tags
Data
read/cmp
Monitor
Cache
Proc.
S'rch
Monitor
Processor
Address
Data
Search
read/cmp
read/
write
V D E
1 0 0 400
0
Tags
old
V D E
1 0 0 400
0
Copy of Tags
Data
read/cmp
Monitor
Cache
broadcast: Invalidate!
Snooping Bus
Arbiter
Memory

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

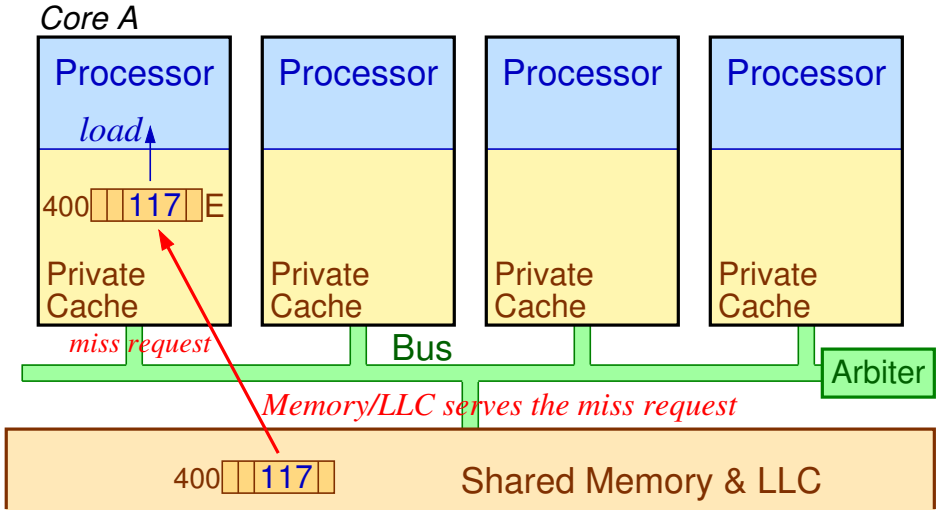
- Επέκταση των Valid-Dirty (per line) bits: τι ξέρω γιά την γραμμή αυτή

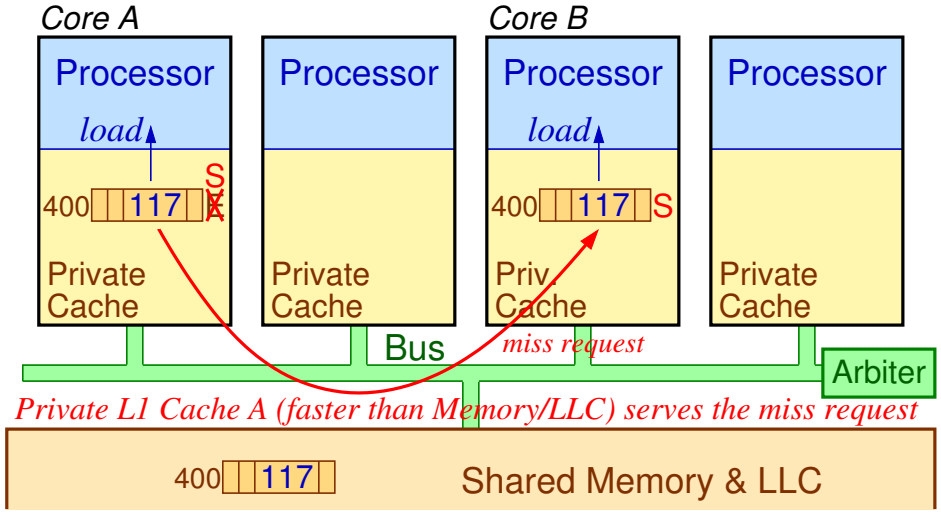
"*MSI*": το απλούστερο πρωτόκολλο snooping coher.

Τρεις μόνον δυνατές καταστάσεις για την κάθε γραμμή:

- ***M*** (Modified): όπως το κλασσικό Dirty, *αναγκαστικά* και exclusive
  - Το αποτέλεσμα δικού μου store, που αναγκάζει Invalidation υπολοίπων
  - Εάν άλλη cache μου ζητήσει αντίγραφο (μέσω miss της), κάνω write-back στη Μνήμη, οπότε το παίρνει και η άλλη cache από το Bus, και πάμε και οι δυο μας σε κατάσταση S (Shared)
- ***S*** (Shared): όπως και το κλασσικό Clean
  - Για απλότητα, *δεν* κρατάμε στοιχεία για άλλες caches, άρα θεωρούμε συντηρητικά ότι υπάρχουν, με υποχρεωτικά επίσης καθαρό αντίγραφο
  - Σε δικό μου store, *πρέπει υποχρεωτικά* να στείλω Invalidation στο Bus
- ***I*** (Invalid): όπως και το κλασσικό Invalid

Πληροφορία Exclusive versus Shared: πώς το ξέρω;





- Εάν σε miss απαντήσει η Μνήμη, τότε καμιά άλλη cache δεν το έχει
- Άρα ξέρω ότι είμαι ***E*** (Exclusive)
- Εάν σε E κάνω store, *δεν* χρειάζεται να στείλω Invalidation στο Bus:
- Πλεονέκτημα – Πρωτόκολλο "***MESI***"

- Σε miss άλλων, εάν εγώ το έχω, *πρέπει* να απαντήσω *πριν* τη Μνήμη
- Τότε και οι δύο πάμε σε κατάσταση ***S*** (Shared) – όπως και η S του MSI
- Μπορεί όλοι οι άλλοι να κάνουν evict & να μείνω μόνος – δεν το ξέρω

"*MOESI*": το πολυπλοκότερο πρωτόκολλο snooping

Και οι πέντε δυνατές καταστάσεις για την κάθε γραμμή:

- ***M*** (Modified): όπως το κλασικό Dirty, συν *ξέρω* ότι είμαι exclusive
  - Εάν άλλος ζητήσει αντίγραφο (miss), του το δίνω και πάω σε κατάσταση **O**wned, αλλά *δεν* κάνω write-back τώρα, άρα ταχύτερο από MSI/MESI
- ***O*** (Owned): Dirty ως προς τη μνήμη, *εγώ* έχω την υποχρέωση write-back· μπορεί και άλλοι να έχουν (ενημερωμένο) αντίγραφο σε κατ. ***S***
- ***E*** (Exclusive): clean & exclusive
- ***S*** (Shared): μπορεί να την έχουν και άλλοι, πάντως εγώ *δεν* έχω υποχρέωση write-back (είτε clean, ή άλλος θα κάνει το write-back)
- ***I*** (Invalid): όπως και το κλασσικό Invalid

Directory-based Coherence, για περισσότερες caches

- Σε snooping coherence το κοινόχρηστο μέσο (bus) είναι bottleneck
  - Broadcast όλων των εντολών: μόνον ένας στέλνει κάθε φορά, απασχολώντας τους πάντες, έστω και αν η εντολή αφορά λίγους μόνον
- *Directory-based Coherence* για να υπερπηδήσουμε το bottleneck:
  - Ένα κοινόχρηστο Directory κρατά πληροφορία, για κάθε γραμμή που υπάρχει σε ≥1 private cache, *ποιές* private caches έχουν αντίγραφο
  - Όταν πρέπει να κάνουμε Invalidate, ρωτάμε το Directory ποιοί έχουν αντίγραφο, και στέλνουμε την εντολή *μόνον* σε αυτούς (multicast)
  - Το Directory συχνά «δίπλα» στην κοινόχρηστη κρυφή μνήμη
  - Για να μη γίνει bottleneck το ίδιο το Directory, συχνά το κάνουμε Διαφυλλωμένο (Interleaved)