Προηγμένες αρχιτεκτονικές υπολογιστών και προγραμματισμός παραλλήλων συστημάτων

07. Πολυεπεξεργαστές

Πολυεπεξεργαστές

- Απλοί στη σχεδίαση και στον προγραμματισμό.
- Απλό δίκτυο (bus).
- Δε μπορούμε να έχουμε πολλούς επεξεργαστές (συνήθως $N < 16$).
- Το bus αποτελεί “bottleneck” καθώς όλη η επικοινωνία διεξάγεται μέσα από αυτό.

Σχεδίαση hardware

- Απαραίτητη η χρήση cache. Μειώνει την κυκλοφορία μεταξύ επεξεργαστών και μνήμης.
- Πρόβλημα: διαχείριση και συγχρονισμός πολλαπλών cache.
  - Συνέπεια πολλαπλών cache.
  - (cache coherence problem).

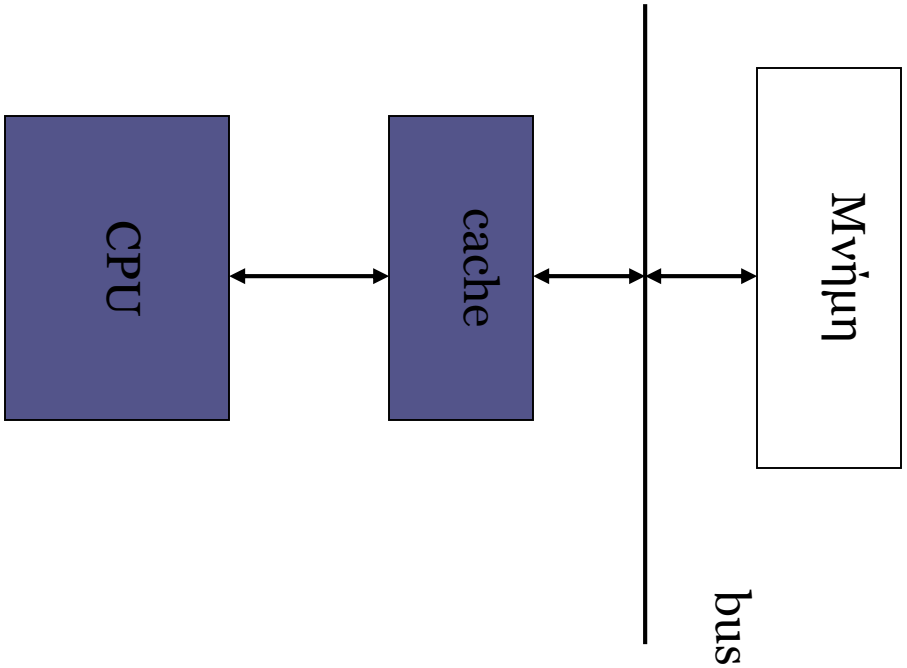
Το πρόβλημα της συνέπειας πολλαπλών cache

- Πολλαπλές κόπιες των ιδίων δεδομένων σε διαφορετικές cache και στην κύρια μνήμη
- Πώς συγχρονίζονται οι cache μεταξύ τους;
- Πώς και πότε ενημερώνεται η κύρια μνήμη;
- Πώς ξέρουμε ότι αυτό που διαβάσαμε από την cache είναι σωστό;

Η cache στον μονο-επεξεργαστή

- Cache = κρυφή, γρήγορη μνήμη αλλά μικρή σε μέγεθος. Συνήθως δε χωράει όλο το πρόγραμμα και όλα τα δεδομένα.
- Ο επεξεργαστής συμβουλεύεται την cache πρώτα και αν δεν βρει εκεί το δεδομένο τότε η cache καλεί την κύρια μνήμη.

Λειτουργία της cache (1)



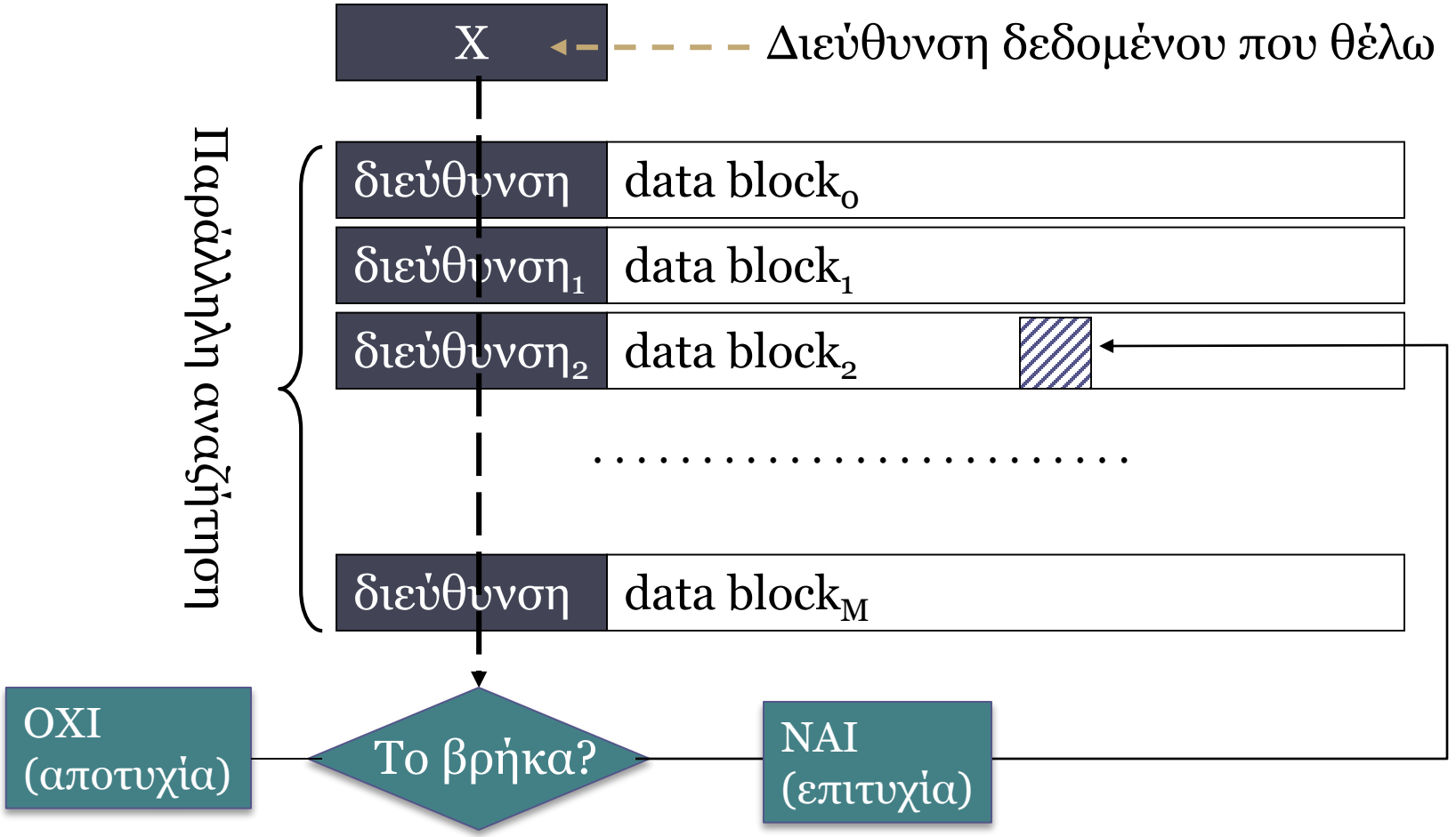
Λειτουργία της cache (2)

- Η cache φυλάσσει ζεύγη εγγραφών του τύπου <διεύθυνση, μπλοκ δεδομένων>. Το κάθε μπλοκ μπορεί να είναι 128 ή 256 ή 512 bytes, κλπ.

- Βασίζεται στην αρχή της τοπικότητας των κλήσεων:

« Αν η εντολή k προσπέλασε τη θέση μνήμης X τότε η εντολή k+1 κατά πάσα πιθανότητα θα προσπελάσει κάποια θέση γειτονική στο X.»

Λειτουργία της cache (3)



Λειτουργία της cache (4)

- **Συχνότητα Επιτυχίας *h* (hit rate):** Εξαρτάται από το μέγεθος της cache, από το μέγεθος του block, από τον αλγόριθμο αντικατάστασης, κ. ο. κ. Συνήθως $h > 99\%$.
- **Συχνότητα αποτυχίας (miss-rate):** $m = 100 - h$.
- **Μέγεθος cache:** Συνήθως 0,5 - 32Mb.
- **Μέγεθος block:** Συνήθως > 32 bytes.
- **Οργάνωση cache:** (direct-mapped, fully associative, set associative).

Λειτουργία της cache (5)

- Η λειτουργία της cache μπορεί να αντιμετωπίσει 4 δυνατές περιπτώσεις:
  1. Επιτυχία ανάγνωσης (Read-hit).
  2. Αποτυχία ανάγνωσης (Read miss).
  3. Επιτυχία εγγραφής (Write-hit).
  4. Αποτυχία εγγραφής (Write-miss).

Read-hit/Read-miss

- **Read-hit:** Ο επεξεργαστής ζητάει να διαβάσει το δεδομένο *D* και αυτό βρίσκεται στην cache.
- Τότε απλά ο επεξεργαστής διαβάζει το *D* από την cache και δεν απασχολεί καθόλου την κύρια μνήμη
- **Read-miss:** Ο επεξεργαστής ζητάει να διαβάσει το δεδομένο *D* και αυτό δεν βρίσκεται στην cache.
- Η cache φέρνει το block *B* που περιέχει το *D* από τη μνήμη και ο επεξεργαστής διαβάζει το *D* από την cache. Ελπίζουμε ότι αμέσως επόμενες κλήσεις του επεξεργαστή θα εξυπηρετηθούν από το ίδιο block *B* οπότε δε θα χρειαστεί να προσπελάσουμε τη μνήμη.

Write-hit

- **Write-hit:** Ο επεξεργαστής ζητάει να γράψει το δεδομένο *D* και αυτό βρίσκεται στην cache.
- Τότε η cache έχει 2 επιλογές:
  1. Να γράψει ο επεξεργαστής το *D* μόνο στην cache και όχι στη μνήμη (μέθοδος write-back). Πρόβλημα συγχρονισμού cache-μνήμης. Απαραίτητη η σημείωση ότι το block *B*, που περιέχει το *D*, είναι πλέον «βρώμικο». Σημειώνεται θέτοντας: `dirty_bit = 1`.
  2. Να γράψει ο επεξεργαστής το *D* και στην cache και στη μνήμη (μέθοδος write-through). Κανένα πρόβλημα συγχρονισμού cache-μνήμης.

Write-back vs. Write-through

- Write-back ταχύτερο από write-through.
- Το write-through καταργεί τις ωφέλειες της cache στις εγγραφές, αφού σε κάθε εγγραφή καλείται η κύρια μνήμη.
- Ωστόσο, οι εγγραφές είναι σπανιότερες από τις αναγνώσεις (20% προς 80%).
- Στο write-through η cache και η μνήμη είναι πάντα σε συμφωνία.
- Το write-back έχει κυριαρχήσει.

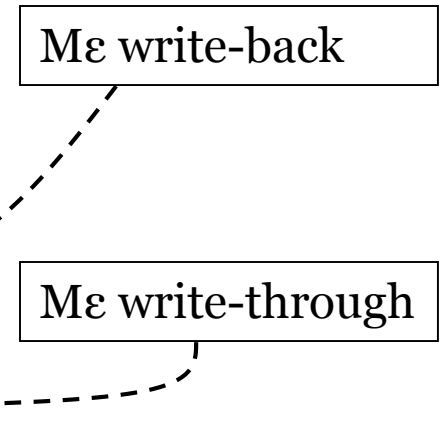
Write-miss

- **Write-miss:** Ο επεξεργαστής ζητάει να γράψει το δεδομένο *D* και αυτό δεν βρίσκεται στην cache.
- Τότε η cache έχει 2 επιλογές:
  1. Να φέρει το block *B*, που περιέχει το *D* από τη μνήμη στην cache και ο επεξεργαστής να γράψει στην cache και όχι στη μνήμη (μέθοδος write-allocate). Συνδυάζεται με write-back.
  2. Να γράψει ο επεξεργαστής το *D* μόνο στη μνήμη (μέθοδος no-write-allocate). Συνδυάζεται με write-through.

Συνέπεια πολλαπλών cache

- Απαραίτητος ο συγχρονισμός των δεδομένων όταν πολλές κόπιες του ιδίου δεδομένου φυλάσσονται σε διαφορετικές cache και στη μνήμη.

Χρόνος	Συμβάν	cache A	cache B	Μνήμη *D*
0				*D*=1
1	*A* read *D*	*D*=1		*D*=1
2	*B* read *D*	*D*=1	*D*=1	*D*=1
3	*A* write *D*=2	*D*=2	*D*=1	*D*=1
3*	*A* write *D*=2	*D*=2	*D*=1	*D*=2



Συνέπεια συστήματος μνήμης

- **Ορισμός -** ένα σύστημα μνήμης καλείται συνεπές αν:
- Το σύστημα μνήμης διατηρεί την σειρά των εντολών του προγράμματος.
- Οι επεξεργαστές επικοινωνούν γράφοντας σε θέσεις μνήμης οι οποίες διαβάζονται κατόπιν από άλλους επεξεργαστές.
- Διαφορετικές εγγραφές στην ίδια θέση μνήμης εκτελούνται σειριακά και αυτή η σειρά φαίνεται η ίδια από όλους τους επεξεργαστές.

Πρωτόκολλα συνέπειας

- Μέθοδος κρυφοκοιτάγματος (snooping).
  - Απαιτεί την ύπαρξη bus – εφαρμόζεται μόνο σε πολυ-επεξεργαστές
- Μέθοδος βασισμένη σε κατάλογο (directory-based).
  - Λειτουργεί σε οποιοδήποτε δίκτυο – εφαρμόζεται σε πολυ-επεξεργαστές και σε πολυ-υπολογιστές με λογικά κοινή μνήμη.

Το πρωτόκολλο snooping (1)

- Βασική φιλοσοφία: όλες οι cache κρυφοκοιτάνε τις άλλες cache όταν κάνουν κάποια δραστηριότητα που μεταβάλει τα δεδομένα (εντολή write) είτε ζητούν δεδομένα από την κύρια μνήμη.
- Μέσο κρυφοκοιτάγματος: το bus.
- Κάθε «επικίνδυνη» δραστηριότητα δημοσιοποιείται στο bus.

Το πρωτόκολλο snooping (2)

- Υποθέτουμε χρήση write-back σε όλες τις cache.
- Ο επεξεργαστής *i* γράφει το *D*. Είτε με write-miss είτε με write-hit πρέπει να ενημερωθούν οι άλλες cache. Δύο επιλογές:
  1. Οι άλλες cache ακυρώνουν τη δική τους κόπια του block *B* που περιέχει το *D* (write-invalidate).
  2. Οι άλλες cache ενημερώνονται με τη νέα τιμή του *D* (write-update).

Write-invalidate vs. Write-update

- Πλεονεκτήματα write-invalidate:
  - Η ακύρωση του block *B* που περιέχει το *D* είναι γρηγορότερη από την ενημέρωση με τη σωστή τιμή.
  - Πολλαπλές εγγραφές είτε στο *D* είτε στο block *B* ακυρώνουν το block μια μόνο φορά.
- Πλεονεκτήματα write-update:
  - Οι cache είναι πάντα ενημερωμένες με τη σωστή τιμή.
- Το write-invalidate είναι ταχύτερο και το προτιμάμε σε σχέση με το write-update.

Το πρωτόκολλο snooping (3)

- **Διαχείριση αποτυχίας ανάγνωσης.**
- Ο επεξεργαστής *i* ζητάει να διαβάσει το *D* αλλά δεν το βρίσκει στην τοπική του cache. Η cache *i* βγάζει αίτηση ανάγνωσης στο bus με αποδέκτες (α) τις άλλες cache που κρυφοκοιτάνε και (β) την κύρια μνήμη.
- Επειδή οι cache είναι ταχύτερες από την μνήμη, αν κάποια από αυτές έχει το *D* τότε το δίνει σταματώντας την προσπέλαση στη μνήμη.

Υλοποίηση snooping (1)

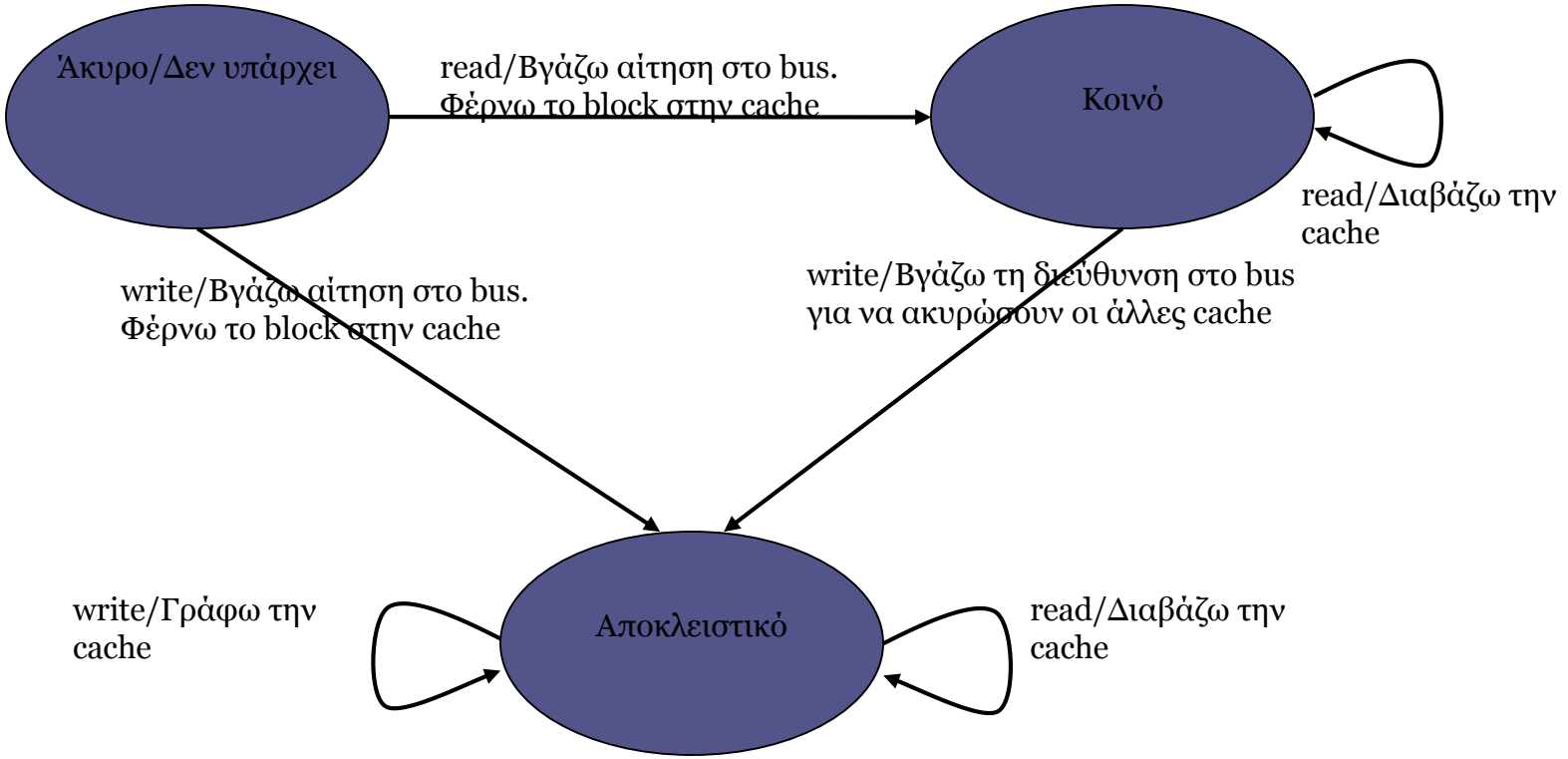
- Μηχανές Πεπερασμένων Καταστάσεων (Finite State Machines) = Γράφοι όπου η μετάβαση από ένα κόμβο (κατάσταση) σε έναν άλλο προκαλείται από συγκεκριμένη διέγερση (input) και προκαλεί συγκεκριμένες δράσεις (actions).
- Κάθε block βρίσκεται σε μια από τρεις καταστάσεις:
  1. Άκυρο/Δεν υπάρχει.
  2. Κοινό = Πιθανώς να υπάρχει και σε άλλη cache.
  3. Αποκλειστικό = Σίγουρα υπάρχει μόνο σε μένα.

Υλοποίηση snooping (2)

- Κάθε cache εκτελεί δράσεις (actions) σύμφωνα με:
  1. τις εντολές του τοπικού της επεξεργαστή (CPU).
  2. τις δραστηριότητες που λαμβάνουν χώρα στο bus με πρωτοβουλία άλλων cache.
- Δύο Μηχανές Πεπερασμένων Καταστάσεων (ΜΠΚ).
- Μια για την τοπική CPU και μια για το BUS.

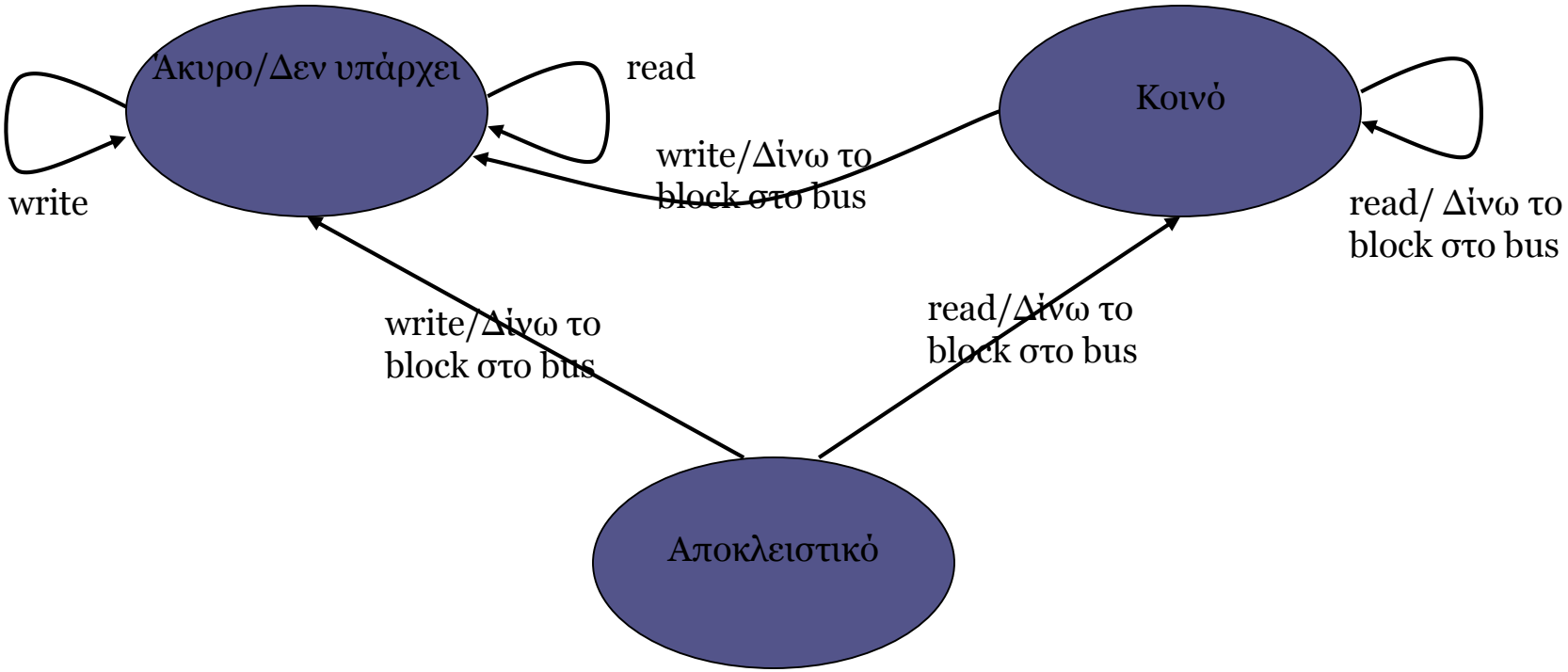
Υλοποίηση snooping (3)

- Μ.Π.Κ. λόγω τοπικής CPU:



Υλοποίηση snooping (4)

- Μ.Π.Κ. λόγω δραστηριότητας του BUS:



Πρωτόκολλο βασισμένο σε κατάλογο

- Το snooping απαιτεί bus για κρυφοκοίταγμα. Τι γίνεται αν έχουμε άλλου τύπου δίκτυο σε ένα πολύ-υπολογιστή με κοινή κατανεμημένη μνήμη;
- Λύση: μέθοδος directory-based
- Όμοιο με snooping. Τρεις καταστάσεις:
  1. Σε καμία cache (Uncached).
  2. Κοινό (Shared).
  3. Αποκλειστικό (Exclusive).

Πρωτόκολλο βασισμένο σε κατάλογο

- Ομοιότητα με snooping: Χρήση μηχανών πεπερασμένων καταστάσεων.
- Διαφορά με snooping: Για κάθε block στη μνήμη φυλάσσεται «κατάλογος» με τις cache που το έχουν.
- Κατάλογος = binary string (0=δεν το έχει, 1=το έχει). Πχ.

0	1	0	0	0	0	1	1	0	0	0

Πρωτόκολλο βασισμένο σε κατάλογο

- Όταν φορτώνεται κάποιο block, φορτώνεται μαζί και ο κατάλογός του. Έτσι ξέρουμε ανά πάσα στιγμή ποιος άλλος έχει το ίδιο block. Όταν γίνεται εγγραφή ή ανάγνωση ενημερώνονται οι άλλοι κάτοχοι των αντιγράφων.
- Προβλήματα:
  1. Χρονοβόρο.
  2. σπατάλη μνήμης με διαχειριστική πληροφορία.

Snooping vs. Κατάλογος

- Snooping γρηγορότερο από κατάλογο.
- Κατάλογος μόνη επιλογή όταν το δίκτυο δεν είναι bus.
- Συμπέρασμα:
  1. Όταν υπάρχει bus (πολυ-επεξεργαστής) χρησιμοποιούμε snooping.
  2. Όταν υπάρχει οποιοδήποτε άλλο δίκτυο (πολυ-υπολογιστής με κοινή κατανεμημένη μνήμη) χρησιμοποιούμε κατάλογο.