Πανεπιστήμιο Δυτικής Μακεδονίας
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Συστήματα Παράλληλης & Κατανεμημένης Επεξεργασίας

Ενότητα 10: Υποκλέπτοντα πρωτόκολλα. 2-state,3-state,4-state (MESI, dragon) cache coherent protocols. Συμφωνία μνημών L1-L2.

Δρ. Μηνάς Δασυγένης
mdasyg@ieee.org
Εργαστήριο Ρομποτικής, Ενσωματωμένων και Ολοκληρωμένων Συστημάτων
http://arch.ece.uowm.gr/mdasyg

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σκοπός της Ενότητας

- Η παρουσίαση βασικών πρωτοκόλλων επίτευξης συμφωνίας μνημών cache.
- Η ανάλυση του πρωτοκόλλου MESI.

Ποιο μοντέλο συνέπειας ακολουθούμε;

- Στους παράλληλους επεξεργαστές μας ενδιαφέρει κυρίως η **συνέπεια της μνήμης** του επεξεργαστή.
- Μια ανάγνωση που έπεται μιας εγγραφής από τον ίδιο επεξεργαστή, θα πρέπει να επιστρέψει τα αποτελέσματα της τελευταίας εγγραφής από αυτόν τον επεξεργαστή (ή κάποιας άλλης εγγραφής από παράλληλο πυρήνα).
- Μια εγγραφή που έπεται μια ανάγνωση από τον ίδιο επεξεργαστή, θα πρέπει να μην επηρεάζει το αποτέλεσμα της ανάγνωσης (να μην εκτελεστεί πρώτα η ανάγνωση δηλαδή).

Συνέπεια μνήμης & Αμοιβαίος αποκλεισμός

- Στα παράλληλα συστήματα υπάρχουν 2 **ανεξάρτητες** απαιτήσεις για τη μνήμη:
  - **Συνέπεια μνήμης**, δηλαδή ο κάθε επεξεργαστής βλέπει τα ίδια δεδομένα σε κάθε θέση της μνήμης (υλοποιείται με hardware {π.χ. writethrough}).
  - **Κλείδωμα θέσης μνήμης**, ώστε στα κρίσιμα τμήματα μόνο μια διεργασία να ενημερώνει την κοινή περιοχή μνήμης (υλοποιείται με software {π.χ. mutex} με υποστήριξη από hardware {π.χ. ατομικές λειτουργίες}).
  - Αν δεν υπάρχει σωστή αντιμετώπιση και των 2, τότε θα ανακύψουν προβλήματα και ασυνεπή αποτελέσματα.

Υποκλέπτοντα Πρωτόκολλα (snooping protocols)

Υποκλέπτοντα πρωτόκολλα συνοχής κρυφής μνήμης

- Τα πολυ-πύρηνα συστήματα που έχουν κρυφές μνήμες συνδεδεμένες στον ίδιο φυσικό δίαυλο χρησιμοποιούν τα “υποκλέπτοντα πρωτόκολλα συνοχής”.
- Είναι ιδιαίτερα δημοφιλή, γιατί δεν έχουν τα μειονεκτήματα του ευρετηρίου, όπως μοναδικό σημείο συμφόρησης, αυξημένη κυκλοφορία στο δίαυλο και προβλήματα κλιμάκωσης.
- **“Coherence is maintained by having all cache controllers “snoop” on the bus and monitor the transactions”.**

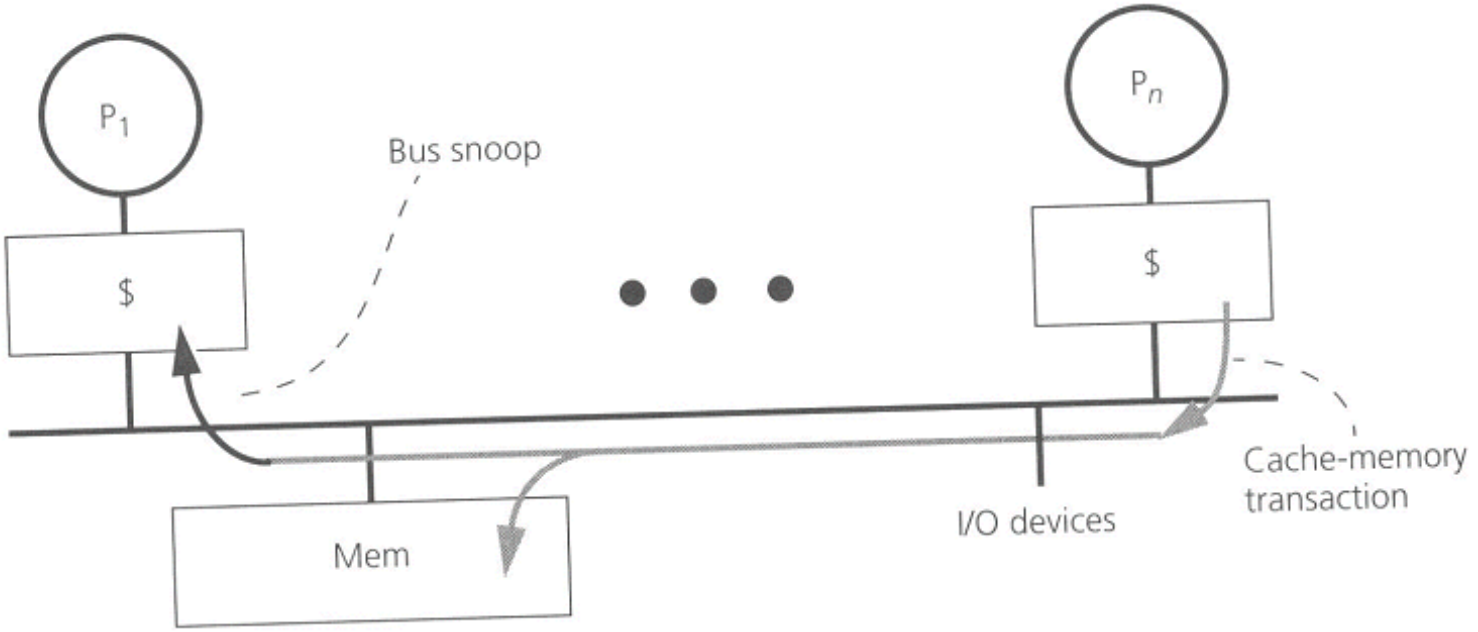
Συνέπεια και γραμμή cache

- Επειδή οι κρυφές μνήμες χρησιμοποιούν γραμμές cache, η συνέπεια στα snooping protocols, διατηρείται σε επίπεδο cache block.

Βασικές ιδιότητες υποκλεπτόντων πρωτοκόλλων

- Όλες οι συναλλαγές στο δίαυλο είναι εμφανείς στις άλλες cache.
- Είναι εμφανείς με την ίδια σειρά (που εμφανίστηκαν στο δίαυλο).
- Πρέπει να εμφανίζονται όλες οι “απαραίτητες” συναλλαγές στο δίαυλο.
- Όλοι οι ελεγκτές cache πρέπει να ενεργούν κατάλληλα όταν βλέπουν τις συναλλαγές.

Παράδειγμα snooping protocol



- L1 write through (1980 SMP implementation) **Write-invalidate** or **write-update.**

Πως λειτουργεί το bus-transaction;

- Υπάρχουν 3 φάσεις:
  - Arbitration (καθορισμός/κρίση).
  - command/address (εντολή διεύθυνση).
  - Data (δεδομένα).
- Αρχικά η μια συσκευή ζητά τον έλεγχο του διαύλου.
- Ο διαιτητής (arbiter) το δίνει.
- Η συναλλαγή συνεχίζει.
- Όλες οι άλλες συσκευές πρέπει να παρακολουθούν το δίαυλο.

Τι γνωρίζετε για τις καταστάσεις στα πρωτόκολλα συνέπειας;

- Κάθε γραμμή cache σε ένα πρωτόκολλο συνέπειας έχει μια κατάσταση συνδεδεμένη με αυτό.
- Οι πιο κοινές καταστάσεις είναι:
  - Invalid/valid (άκυρο ή όχι)
  - clean/dirty (τροποποιημένο ή όχι)
- Μια αλλαγή στη γραμμή cache μπορεί να συμβεί είτε αν γίνει πρόσβαση σε αυτό άμεσα, ή αν μεταδοθεί στο δίαυλο κάποια πληροφορία σχετική με αυτό.
- Όλα τα δεδομένα έχουν μια κατάσταση. Ακόμη και αυτά που δε βρίσκονται στη cache θεωρούνται ότι είναι invalid.

Η πιο απλή cache

- Η πιο απλή cache **για συστήματα με ένα επεξεργαστή** είναι η write-through, write-no-allocate η οποία έχει ανάγκη μόνο για 1 bit (valid/invalid).
- Όταν έρθουν τα δεδομένα σε μια αστοχία ανάγνωσης τότε σημειώνονται ως valid.
- Σε εγγραφή ενημερώνεται αμέσως και η κρυφή μνήμη και η μνήμη RAM.
- **Μια πιο σύνθετη cache** είναι η write-back cache η οποία έχει ακόμη ένα bit το dirty/clean που υποδηλώνει ότι τα δεδομένα στην cache δεν έχουν μεταφερθεί στη μνήμη.

Πιο σύνθετες cache

- Οι πιο σύνθετες cache έχουν πολλαπλές καταστάσεις.
- Σε κάθε cache η ίδια διεύθυνση μπορεί να είναι σε διαφορετικές καταστάσεις.
- Οπότε μιλάμε για ένα διάνυσμα **p** διαστάσεων, όπου **p** είναι ο αριθμός των κρυφών μνημών.
- Σε ένα πρωτόκολλο snooping η κατάσταση μιας γραμμής αλλάζει είτε από τον επεξεργαστή είτε από το δίαυλο.

Ποια είναι τα τμήματα ενός πρωτοκόλλου snooping

- Αποτελείται:
  - Από τις προκαθορισμένες καταστάσεις.
  - Από το διάγραμμα μετάβασης καταστάσεων οι οποίες οφείλονται είτε σε αιτήματα του επεξεργαστή είτε σε μηνύματα από τον κοινό δίαυλο με τις άλλες κρυφές μνήμες.
  - Τις ενέργειες που συνδέονται με κάθε αλλαγή κατάστασης, σύμφωνα με το διάγραμμα καταστάσεων.

Τι σημαίνει atomic bus;

- Atomic bus, σημαίνει ότι όλες οι φάσεις αποστολής μιας συναλλαγής στο δίαυλο (διαιτησία, διεύθυνση, δεδομένα) μπορούν να συμβούν χωρίς να διακοπούν από μια άλλη συσκευή που θέλει να αποκτήσει και αυτή έλεγχο του διαύλου.
- Όλες οι επικοινωνίες του διαύλου μπαίνουν σε μια σειρά, η οποία ονομάζεται bus order.

Πότε μια κρυφή μνήμη είναι owner μιας γραμμής;

- Μια κρυφή μνήμη write-back, θεωρείται ότι είναι owner αν έχει μια τροποποιημένη γραμμή (dirty bit) στην cache την οποία θα πρέπει να τοποθετήσει στο δίαυλο σε περίπτωση που ζητηθεί από κάποια άλλη κρυφή μνήμη.
- Η cache θα πρέπει να έχει αποκλειστικότητα της γραμμής (exclusivity).
- Αν δεν έχει exclusivity τότε θα πρέπει να ζητήσει αυτήν την κατάσταση από τις άλλες κρυφές μνήμες. Ακόμη και να έχει σε έγκυρη κατάσταση τη γραμμή απαιτείται μια συναλλαγή με το δίαυλο για αυτό και ονομάζεται write-miss. Αν είναι σε modified κατάσταση τότε δεν απαιτείται write-miss.

Ποια είναι τα πιο σημαντικά υποκλέπτοντα πρωτόκολλα;

- Πρωτόκολλο ακύρωσης εγγραφής.
  - Ακυρώνει άλλα αντίγραφα σε άλλες cache με την εγγραφή.
- Πρωτόκολλο ενημέρωσης εγγραφής
  - Όλες οι κρυφές μνήμες παρακολουθούν το δίαυλο για εγγραφές.
- Έχει επικρατήσει το πρωτόκολλο ακύρωσης εγγραφής.

Γιατί έχει επικρατήσει το πρωτόκολλο ακύρωσης;

- Πολλαπλές εγγραφές στην ίδια λέξη χωρίς αναγνώσεις απαιτούν πολλαπλές κοινοποιήσεις ενημέρωσης, αλλά μόνο μια ακύρωση σε ένα πρωτόκολλο ακύρωσης.
- Το πρωτόκολλο ακύρωσης ενεργεί σε μπλοκ κρυφής μνήμης, ενώ της ενημέρωσης σε λέξεις ή bytes. Αν τροποποιούνται πολλαπλές λέξεις στην ίδια γραμμή cache απαιτούνται πολλαπλές ενημερώσεις αλλά μόνο μια ακύρωση.
- Στο πρωτόκολλο ακύρωσης υπάρχει μια καθυστέρηση σε αστοχία ανάγνωσης επειδή πρέπει να μεταφερθεί η γραμμή, ενώ στο πρωτόκολλο ενημέρωσης η αλλαγή έχει ήδη μεταφερθεί.
- Η ακύρωσης απαιτεί λιγότερο εύρος ζώνης.

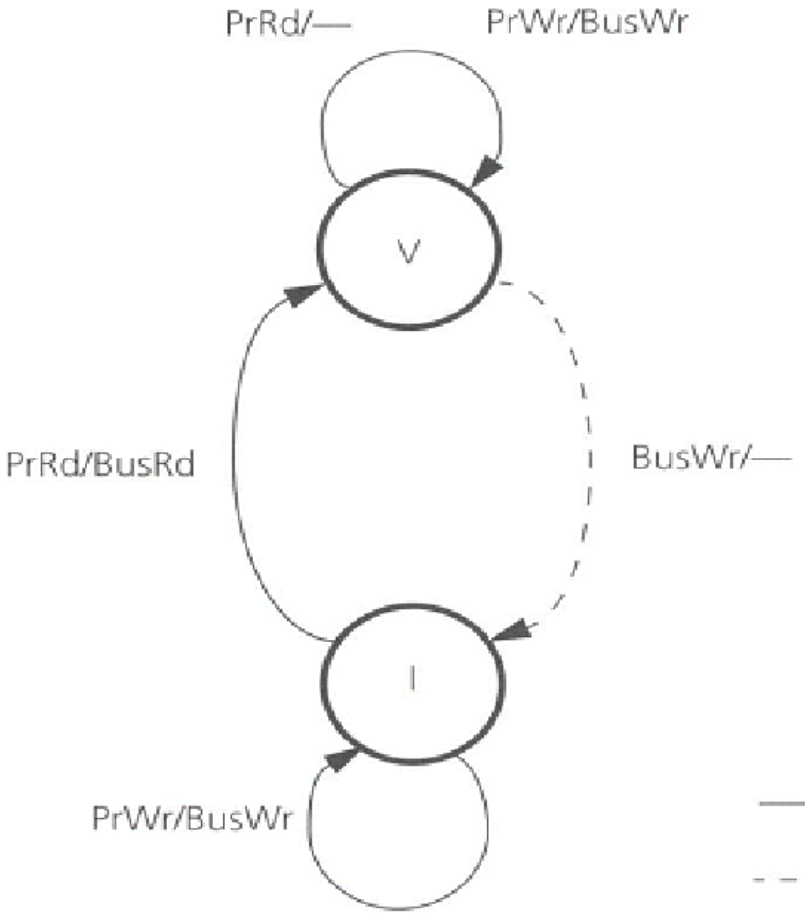
Υλοποίηση snooping write-invalidate (1/2)

- Η κάθε cache υποκλέπτει τα δεδομένα που στέλνονται από τις υπόλοιπες cache.
- Αν ένας άλλος επεξεργαστής ζητήσει δεδομένα για εγγραφή, τότε όσοι έχουν τη συγκεκριμένη διεύθυνση την ακυρώνουν.
- Απαιτείται σειριοποίηση των εγγραφών, ιδιαίτερα αν υπάρχει ανταγωνισμός για εγγραφή στην ίδια περιοχή μνήμης.
- Ο πρώτος επεξεργαστής που θα αποκτήσει πρόσβαση στο δίαυλο αναγκάζει τον άλλο επεξεργαστή να ακυρώσει το αντίγραφό του.

Υλοποίηση snooping write-invalidate (2/2)

- Αν έχουμε εγγραφή στο δίαυλο, τότε αν μια cache έχει τροποποιήσει τα δεδομένα, τα γράφει στη μνήμη.
- Μπορεί να χρησιμοποιηθεί και το bit εγκυρότητας για να ακυρωθεί μια γραμμή.
- Μπορεί να προστεθεί και άλλο bit για να έχουμε και καταστάσεις, όπως αποκλειστική πρόσβασης (ιδιοκτήτης).
- Χρησιμοποιείται ένας ελεγκτής πεπερασμένων καταστάσεων για τις πιθανές μεταβάσεις.

Ένα snoopy protocol write-through με 2 καταστάσεις



A/B σημαίνει αν παρατηρηθεί το A τότε θα γίνει το B.

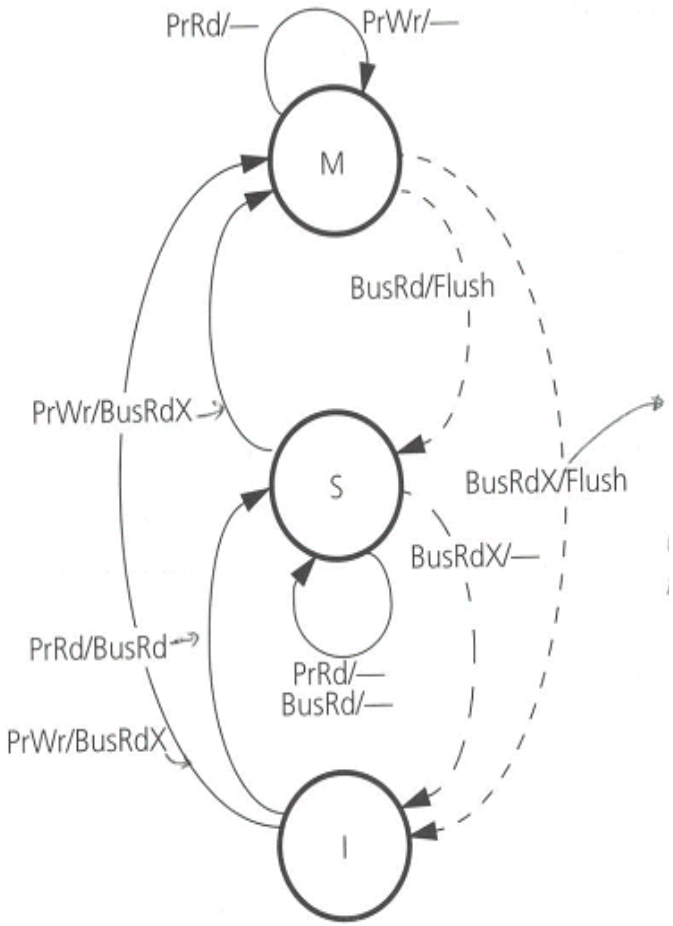
- PrRd Processor Read
- PrWr Processor Write
- BusRd Bus Read
- BusWr Bus Write

Processor-initiated transactions

Bus-snooper-initiated transactions

Το πρωτόκολλο MSI

Ένα πρωτόκολλο 3 καταστάσεων MSI (silicon graphics 4d)



- Χρησιμοποιεί write-back cache.
- States: Modified, Shared, Invalid:
  - Modified: exclusive, only this cache is valid.
  - Shared: memory in sycn, may or may not be shared.
  - Read-exclusive απαιτείται αν θέλουμε να τροποποιήσουμε μια γραμμή cache.
  - BusRd: Bus Read.
  - BusRdX: Bus Read Exclusive.
  - BusWB: Bus Write back.
- Οι μεταβάσεις με - - - οφείλονται στο δίαυλο.
- Η διαφορά με το write-update είναι ότι δε δημιουργείται επικοινωνία για κάθε εγγραφή. Μόνο όταν υπάρχει λόγος.
- Αρχικά ήταν η μετάβαση από M->I (BusRd).

Σχεδιαστικές αποφάσεις που επηρεάζουν τις επιδόσεις

- Στο προηγούμενο πρωτόκολλο, αρχικά είχαμε (δε φαίνεται στο γράφο) μετάβαση από το M->I για BusRd με συνέπεια να δοθεί πλήρης (exclusive) πρόσβαση στον επεξεργαστή που το ζητούσε.
- Αυτό συνέβαινε γιατί συνήθως ένας επεξεργαστής διαβάζει κάτι για να το τροποποιήσει.
- Στα σύγχρονα συστήματα επιλέχθηκε M->S, επειδή τις περισσότερες φορές βασιζόμαστε στην κοινή χρήση. Όμως, σε περίπτωση εγγραφής τότε απαιτείται ένα read exclusive που σημαίνει επιπρόσθετη καθυστέρηση.

Η προσθήκη μιας κατάστασης

- Στο πρωτόκολλο 3 καταστάσεων αν θέλουμε να τροποποιήσουμε μια γραμμή απαιτείται:
  - Busrd για να έρθει η γραμμή σε κατάσταση S.
  - BusRdX για να γίνει αποκλειστικής πρόσβασης σε M.
- Μπορεί να βελτιωθεί με την προσθήκη μιας νέας κατάστασης που ονομάζεται exclusive ή αποκλειστικής πρόσβασης (E).
- Είναι ανάμεσα στο shared και modified.
- Δεν απαιτούνται 2 transactions για να γίνει η εγγραφή.

Το πρωτόκολλο MESI

Συνέπεια Κρυφής Μνήμης με 4 καταστάσεις

- Για να υπάρχει συμφωνία μνημών cache σε ένα SMP σύστημα, χρησιμοποιείται ένα άλλο πρωτόκολλο συνήθως το MESI.
- Το πρωτόκολλο αυτό χρησιμοποιείται ευρέως από κάθε σύγχρονο SMP σύστημα.
- Στο MESI κάθε γραμμή CACHE προστίθενται δυο bit κατάστασης ανά ετικέτα.
- => Κωδικοποιούνται 4 καταστάσεις

Ποιες είναι οι καταστάσεις του MESI;

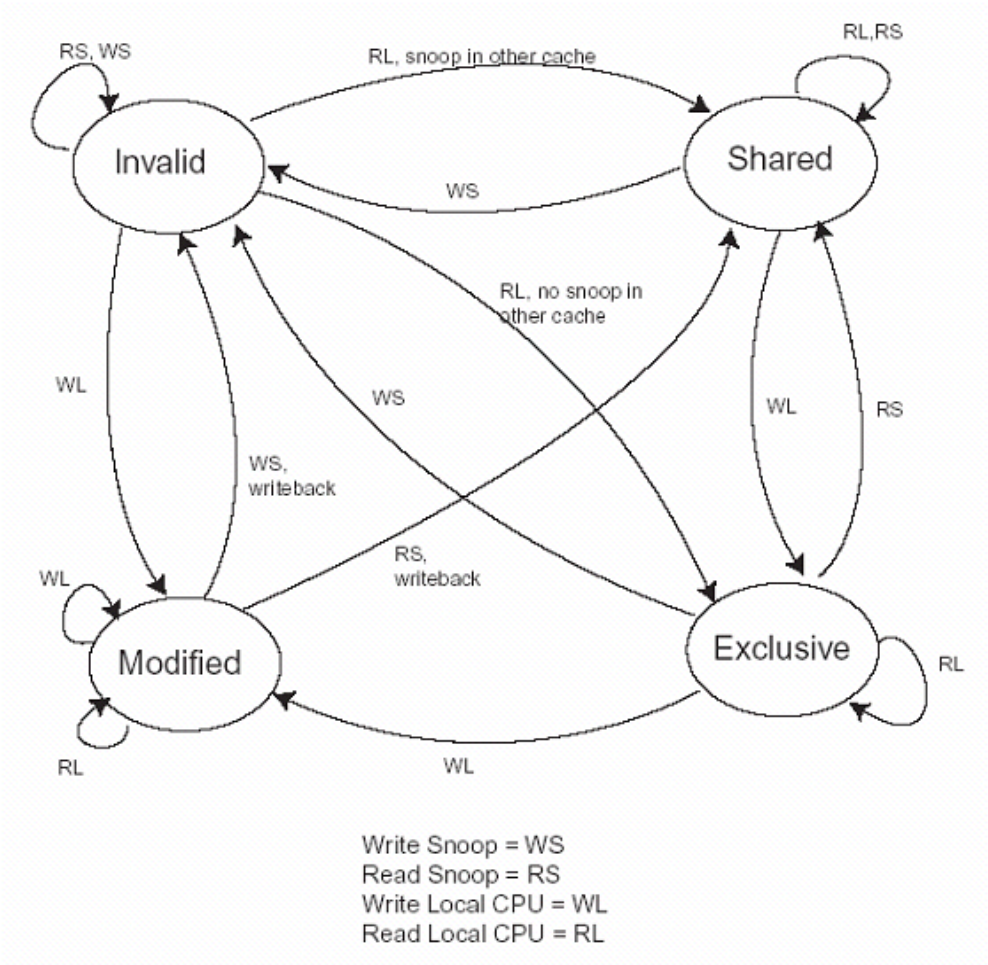
- **Modified (τροποποιημένη)**: Η γραμμή στη μνήμη cache έχει τροποποιηθεί (είναι διαφορετική από την κύρια μνήμη) και είναι διαθέσιμη μόνο σ' αυτή την μνήμη cache.
- **Exclusive (αποκλειστική)**: Η γραμμή στη μνήμη cache είναι ίδια με τη γραμμή στην κύρια μνήμη και δεν υπάρχει σε καμία άλλη μνήμη cache.
- **Shared (διαμοιραζόμενη)**: Η γραμμή στη μνήμη cache είναι ίδια με τη γραμμή στην κύρια μνήμη και μπορεί να υπάρχει και σε άλλη μνήμη cache.
- **Invalid (άκυρη)**: Η γραμμή στη μνήμη cache δεν περιέχει έγκυρα δεδομένα.

Καταστάσεις γραμμών μνήμης cache MESI

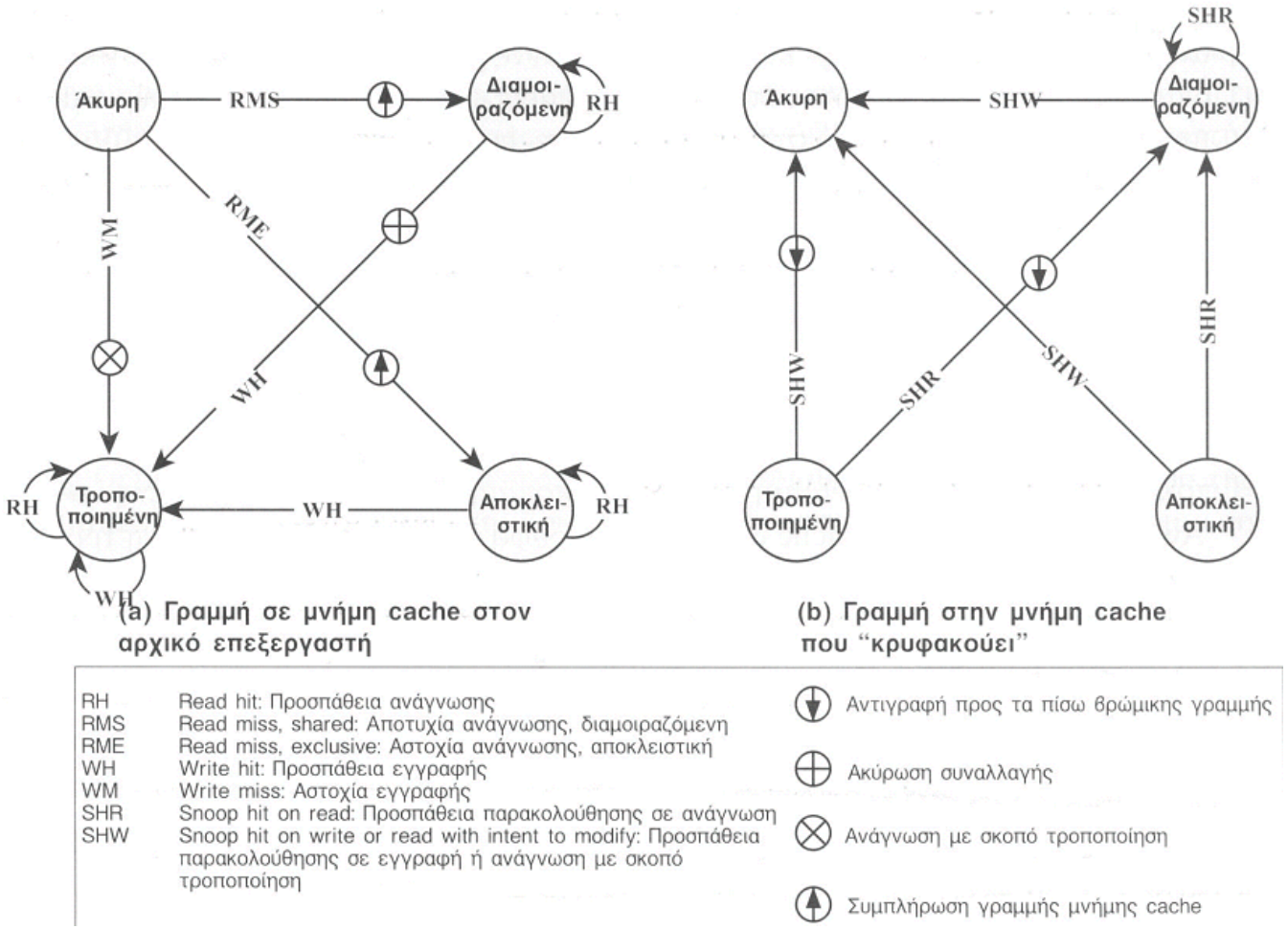
	M Τροποιημένη	E Αποκλειστική	S Διαμοιραζόμε	I Ακυρη
Ισχύει αυτή η γραμμή μνήμης cache ;	Ναι	Ναι	Ναι	Οχι
Το αντίγραφο μνήμης είναι ...	Μη ενημερωμένο	Εγκυρο	Εγκυρο	-
Υπάρχουν αντίγραφα σε άλλες μνήμες cache ;	Οχι	Οχι	Ισως	Ισως
Η εγγραφή σ' αυτή την γραμμή	Δεν πηγαίνει σε αρτηρία	Δεν πηγαίνει σε αρτηρία	Πηγαίνει σε αρτηρία και ενημερώνει μνήμη cache	Πηγαίνει κατευθείαν σε αρτηρία

- Κάθε γραμμή cache έχει τα δικά της bit κατάστασης.
- Κάθε γραμμή cache βρίσκεται σε μια κατάσταση.

Διάγραμμα Μεταβάσεων MESI (1/2)



Διάγραμμα Μεταβάσεων MESI (2/2)



Περιγράψτε τι γίνεται στο MESI κατά την αστοχία ανάγνωσης

- Όταν η local cache αστοχήσει, τότε εισάγεται η διεύθυνση της κύριας μνήμης στο κοινό δίαυλο.
  - Αν μια άλλη cache έχει καθαρό αντίγραφο σε EXCLUSIVE κατάσταση τότε εκείνος ο επεξεργαστής αλλάζει τη γραμμή σε SHARED, και ο local τη διαβάζει από τη μνήμη και ορίζεται σε SHARED.
  - Ομοίως αν παραπάνω από 1 caches έχουν τη γραμμή τότε παραμένει η SHARED κατάσταση, και η local τη διαβάζει από τη μνήμη και ορίζεται σε SHARED.
  - Αν κάποια άλλη cache έχει MODIFIED, τότε διαθέτει αυτή τη γραμμή στην αιτούσα και αλλάζει από MODIFIED σε SHARED, όπως και η local.
  - Αν κανείς δεν έχει αντίγραφο, τότε διαβάζεται από τη RAM και αλλάζει από INVALID σε EXCLUSIVE.

Περιγράψτε τι γίνεται στο MESI κατά την επιτυχία ανάγνωσης

- Αν υπάρχει στη cache η συγκεκριμένη διεύθυνση τότε η cache δίνει στον επεξεργαστή τα στοιχεία που χρειάζεται και δεν υπάρχει αλλαγή κατάστασης.
  - Αν είναι EXCLUSIVE παραμένει EXCLUSIVE.
  - Αν είναι MODIFIED παραμένει MODIFIED.
  - Αν είναι SHARED παραμένει SHARED.
- Τι γίνεται με την κατάσταση INVALID;

Περιγράψτε τι συμβαίνει στο MESI κατά την αστοχία εγγραφής (1/2)

- Ο επεξεργαστής που κάνει write miss, εκδίδει σήμα “ανάγνωση με σκοπό την τροποποίηση” (read-with-intent-to-modify, RWITM).
- 1ο σενάριο: Καμία μνήμη cache δεν έχει τροποποιημένο αντίγραφο.
- Δεν απαντάει κάνεις άλλος επεξεργαστής.
- Ο επεξεργαστής προχωράει σε ανάγνωση της γραμμής από RAM και τροποποίηση. Αλλάζει από INVALID σε MODIFIED.
- Αν υπάρχει αντίγραφο σε περισσότερες cache τότε αλλάζουν την κατάσταση από SHARED/EXCLUSIVE σε INVALID.

Περιγράψτε τι συμβαίνει στο MESI κατά την αστοχία εγγραφής (2/2)

- 2ο σενάριο: Κάποια cache έχει τροποποιημένο αντίγραφο.
  - Απαντάει η cache που έχει MODIFIED και κάνει εγγραφή στη RAM. Αλλάζει η κατάσταση από MODIFIED σε INVALID.
  - Ο αρχικός επεξεργαστής επαναλαμβάνει RWITM, και αν δεν απαντήσει κανείς, διαβάζει τη λέξη από τη RAM. Αλλάζει η κατάσταση από INVALID σε MODIFIED.

Περιγράψτε τι γίνεται στο MESI κατά το write hit

- Write hit συμβαίνει σε γραμμή που είναι στη cache και είναι modified, exclusive, shared. Διακρίνονται λοιπόν οι παρακάτω περιπτώσεις:
  - **Exclusive**, η γραμμή είναι αποκλειστική οπότε γίνεται η τροποποίηση και αλλάζει η κατάσταση σε MODIFIED.
  - **Modified**, η γραμμή είναι αποκλειστική οπότε γίνεται η τροποποίηση και παραμένει η κατάσταση σε MODIFIED.
  - **Shared**, ο επεξεργαστής στέλνει ειδικό σήμα για τις άλλες cache. Αλλάζουν από Shared σε Invalid στις άλλες κρυφές μνήμες. Η αρχική cache αλλάζει σε MODIFIED.

Γιατί υπάρχουν οι καταστάσεις shared/exclusive;

- Όπως είδαμε υπάρχει η κατάσταση SHARED και η EXCLUSIVE.
- Αυτό είναι βελτιστοποίηση και βοηθάει στην κατάργηση της ανάγκης μιας συναλλαγής διαύλου κατά την εγγραφή σε ένα τέτοιο μπλοκ.

Το πρωτόκολλο dragon

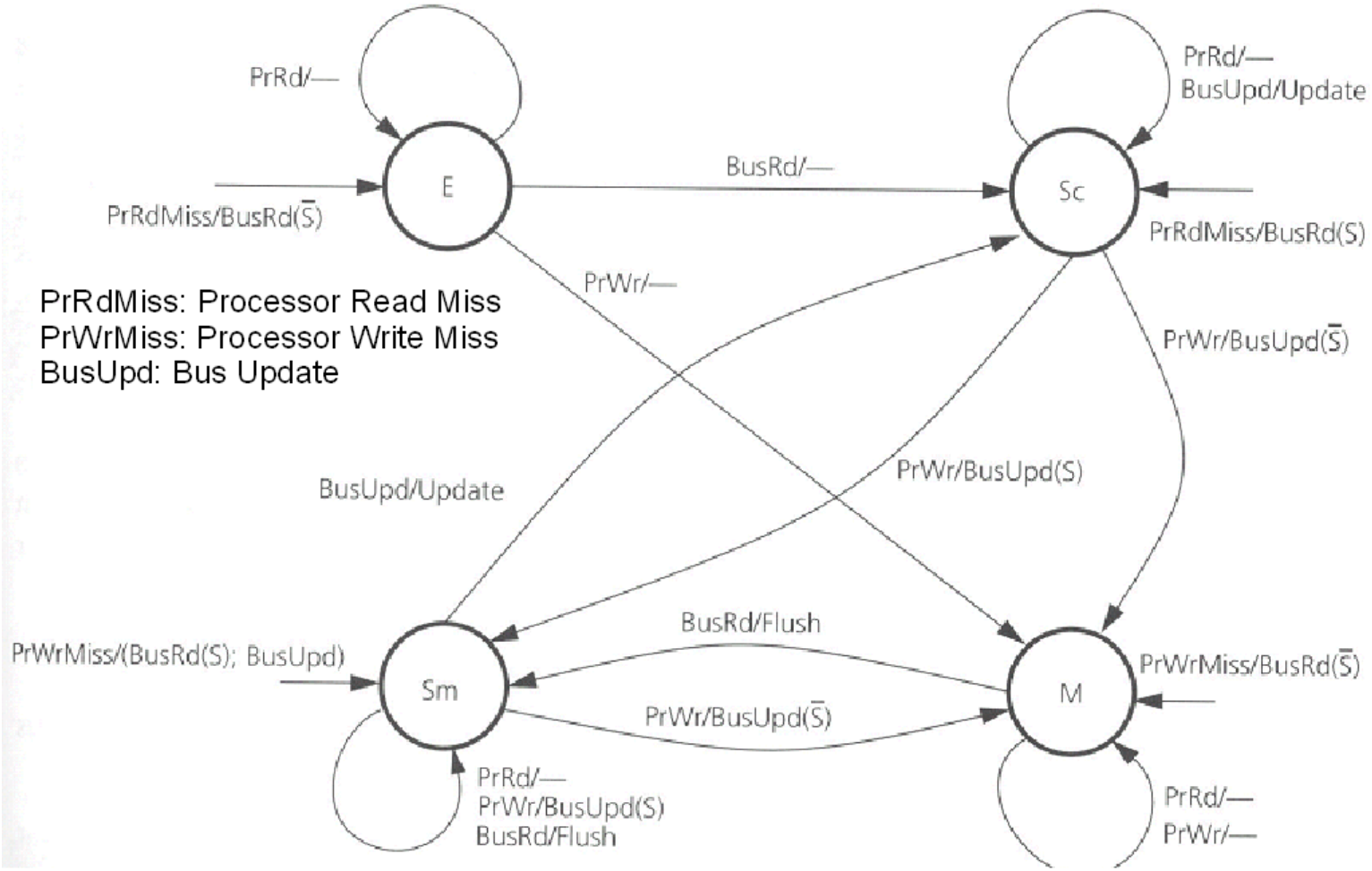
Το πρωτόκολλο 4 καταστάσεων συνέπειας κρυφής μνήμης dragon (1/3)

- Χρησιμοποιείται σε Sun Sparcserver.
- 4 καταστάσεις:
  - Exclusive-clean (E):
    - Μόνο μια cache έχει αντίγραφο και δεν έχει τροποποιηθεί.
  - Shared-clean (Sc):
    - Πολλαπλές cache έχουν αντίγραφα. Η μνήμη ίσως είναι ή δεν είναι συνεπής.
  - Shared-modified (Sm):
    - Δυο ή παραπάνω κρυφές μνήμες έχουν αυτή τη γραμμή και είναι τροποποιημένη. Η κυρίως μνήμη δεν είναι συνεπής για αυτή τη γραμμή. Μόνο μια cache θα είναι σε Sm. Οι άλλες θα είναι Sc
  - Modified (M):
    - Αποκλειστική πρόσβαση και τροποποιημένο.

Το πρωτόκολλο 4 καταστάσεων συνέπειας κρυφής μνήμης dragon (2/3)

- Δεν υπάρχει κατάσταση I (κατάσταση invalid).
- Είναι update-based πρωτόκολλο.
- Είναι πάντα up-to-date οι γραμμές που είναι στη cache.
- Υπάρχει όμως και ένα valid bit που χρησιμοποιείται μόνο για την αρχή.

Το πρωτόκολλο 4 καταστάσεων συνέπειας κρυφής μνήμης dragon (3/3)



Σχεδιαστικές αποφάσεις για το dragon

- Μπορεί να παραληφθεί η Sm κατάσταση.
- Υπάρχει γιατί βασίζεται στο γεγονός ότι η SRAM είναι πιο γρήγορη από τη DRAM, οπότε δεν είναι σωστό να περιμένει μέχρι η DRAM να ενημερώσει την τιμή σε μια κατάσταση BusUpd.
- Άλλη απόφαση... αν είμαστε σε Sc τότε σε περίπτωση που γίνει replace να ενημερωθούν οι άλλες κρυφές μνήμες ώστε αν είναι μια τελευταία να μπει σε exclusive;
- Απαιτείται:
  - Write serialization, write completition detection, write atomicity, atomic bus.

Κόστος συνέπειας μνήμης.. (1/2)

- Υπάρχουν περιπτώσεις που θέλουμε να αποφύγουμε το αυξημένο κόστος της διατήρησης της συνέπειας.
- Για παράδειγμα, όταν δημιουργούνται δεδομένα τα οποία δε θέλουμε να τα αποθηκεύσουμε στην κρυφή μνήμη, γιατί γνωρίζουμε ότι δεν πρόκειται να χρησιμοποιηθούν σύντομα.
- Σύμφωνα με το μοντέλο συνέπειας, όταν δημιουργούμε δεδομένα και θέλουμε να τα αποθηκεύσουμε στην κρυφή μνήμη, επιλέγεται μια κρυφή γραμμή, αποβάλλονται τα δεδομένα της γραμμής, και τοποθετούνται τα δεδομένα (allocate on write).

Κόστος συνέπειας μνήμης.. (2/2)

- Όμως μπορεί να εκ διώχνονται δεδομένα τα οποία έχουν μεγάλο βαθμό επαναχρησιμοποίησης, από δεδομένα που γνωρίζουμε ότι δε θα επαναχρησιμοποιηθούν. Για παράδειγμα: παραγωγή αποτελεσμάτων για μεγάλους πίνακες (array initialization) στην αρχή της εφαρμογής, όπου καθώς γεμίζει ο πίνακας, απομακρύνονται οι πρώτες γραμμές. Επίσης, συνήθως εμφανίζεται το φαινόμενο να απομακρύνονται οι γραμμές της κρυφής μνήμης, χωρίς να έχουν επαναχρησιμοποιηθεί ως αυτό το σημείο.

- Για αυτές τις περιπτώσεις, οι σύγχρονοι επεξεργαστές υποστηρίζουν τις εντολές "**Non temporal**", δηλαδή εντολές που δεν έχουν τοπικότητα και δεν πρέπει να επηρεάζουν την κρυφή μνήμη.

Σχεδιαστικές αποφάσεις

Η υλοποίηση είναι πιο σύνθετη

- Το παραπάνω πρωτόκολλα θεωρούν ότι οι πράξεις είναι αδιάσπαστες, δηλαδή μια πράξη μπορεί να εκτελεσθεί με τέτοιο τρόπο ώστε να μη μπορεί να υπάρχει κάποια άλλη πράξη ενδιάμεσα.
- Οι μη αδιάσπαστες ενέργειες εισάγουν την πιθανότητα αδιεξόδου στο πρωτόκολλο.

Τι ονομάζουμε αστοχία κτήσης ή αναβάθμισης

- Αστοχία κτήσης ή αναβάθμισης συμβαίνει όταν μια cache έχει σε SHARED κάποια δεδομένα, αλλά θέλει να γράψει σε αυτά, οπότε στέλνεται ένα σήμα εγγραφής στο δίαυλο, χωρίς όμως να μεταφερθούν δεδομένα. Αυτό έχει ως συνέπεια οι άλλες κρυφές μνήμες να αλλάξουν από SHARED σε INVALID για αυτή τη γραμμή, και για την ίδια cache από SHARED σε MODIFIED.
- Είναι διαφορετικό η αστοχία εγγραφής και η ευστοχία εγγραφής.

Που αλλού εμφανίζεται η συνέπεια κρυφών μνημών;

- Προβλήματα συνέπειας κρυφών μνημών εμφανίζονται και σε συστήματα ενός επεξεργαστή εξαιτίας των DMA και I/O.
- Το DMA μεταφέρει δεδομένα από τα περιφερειακά προς τη μνήμη και αντίστροφα.
- Θα πρέπει ο επεξεργαστής να ενημερώνεται για την αλλαγή αν το DMA έχει γράψει κάτι στη μνήμη.
- Θα πρέπει το DMA να μη χρησιμοποιεί άκυρα δεδομένα αν η cache είναι write-back.
- Επειδή το I/O δεν είναι τόσο συχνό είναι πιο εύκολο να αντιμετωπιστεί.
- Η πιο κοινή λύση για memory mapped I/O είναι να μαρκαριστούν οι συγκεκριμένες διευθύνσεις μνήμες ως uncacheable.

Τι γνωρίζετε για την τεχνική uncachable memory;

- Είναι μια διαδεδομένη τεχνική για να μη προκαλείται πρόβλημα συνέπειας, ιδιαίτερα σε I/O.
- Δε χρησιμοποιείται σε πολυπύρηνα συστήματα γιατί θα μειώνονταν η απόδοση, δεδομένου ότι υπάρχει έντονη πρόσβαση σε κοινά δεδομένα στις παράλληλες εφαρμογές.

Ποιες ιδιότητες έχει το μοντέλο συνέπειας που ακολουθούμε;

- **Write-propagation:**
  - Όλες οι εγγραφές γίνονται εμφανείς σε όλους τους επεξεργαστές.
- **Write serialization:**
  - Όλες οι εγγραφές σε μια διεύθυνση κοινής μνήμης εμφανίζονται με την ίδια σειρά σε όλες τις άλλες διεργασίες.
  - Παράδειγμα: Αν μια διεργασία βλέπει ότι πρώτα γράφτηκε w1 σε μια διεύθυνση μνήμης και μετά η w2, θα πρέπει όλες οι διεργασίες να βλέπουν αυτή τη σειρά. Δηλαδή, η τιμή που θα έχει η διεύθυνση μνήμης θα είναι w2.

Σχεδιαστικές Αποφάσεις!

- Σε περίπτωση BusRd ποιος θα δώσει τα δεδομένα; Μια άλλη γραμμή cache (αν τα έχει) ή η μνήμη.
- Αρχικά ήταν η cache (cache-to-cache sharing) με επιχείρημα ότι επειδή ήταν τεχνολογίας SRAM θα ήταν πιο γρήγορη από τη DRAM.
- Σε πολυεπεξεργαστές όμως με κατανεμημένες μνήμες και φυσικές απομακρυσμένες μνήμες είναι αποδοτικό.
- Όμως,
  - Ήταν πιο σύνθετη υλοποίηση.
  - Είναι ακριβή υλοποίηση.
  - Η κυρίως μνήμη θα πρέπει να περιμένει μήπως μια άλλη κρυφή μνήμη δώσει τα δεδομένα.

Συμφωνία κρυφών μνημών L1-L2

Συμφωνία μνημών L1-L2

- Το MESI λειτουργεί αποδοτικά όταν υπάρχει κοινή αρτηρία όπου συνδέονται όλες οι κρυφές μνήμες και έτσι μεταδίδονται μηνύματα άμεσα από μια cache στην άλλη ή μπορούν να εποπτεύουν την κοινή αρτηρία.
- Συνήθως αυτές οι μνήμες είναι L2 (ή L3 αν είναι ιδιωτικές).
- Οι L1 δε συνδέονται απευθείας με την αρτηρία και δε μπορεί να εμπλακεί σε κρυφό πρωτόκολλο.
- Όμως, χρειάζεται να διατηρηθεί η ακεραιότητα των δεδομένων σε όλες τις μνήμες cache.

Πως επιτυγχάνεται συμφωνία L1-L2 (1/2)

- Επεκτείνεται το MESI στις μνήμες L1.
- Κάθε γραμμή της κρυφής μνήμης L1 έχει τα bit ένδειξης κατάστασης.
- Η κατάσταση της γραμμής της L1, θα πρέπει να παρακολουθεί τη L2 (και αντίστροφά).
- Χρησιμοποιείται write-through L1 προς L2, ώστε οι τροποποιήσεις να πηγαίνουν άμεσα στην L2 και στις υπόλοιπες κρυφές μνήμες.
- Τα περιεχόμενα της L1 πρέπει πάντα να είναι υποσύνολο της L2 (η προσεταιριστικότητα της L2 να είναι ίση ή μεγαλύτερη της L1).

Πως επιτυγχάνεται συμφωνία L1-L2 (2/2)

- Μπορεί να χρησιμοποιηθεί και write-back L1, αλλά απαιτείται μια πολυπλοκότερη υλοποίηση.
- Οι αλλαγές στην L2 (bit κατάστασης) μεταφέρονται άμεσα στην L1.
- Υπάρχουν και άλλες προσεγγίσεις για διατήρηση της συμφωνίας.

Στοιχεία Παράλληλου Υπολογισμού

Μοντέλα και Αρχιτεκτονική Μνήμης

- Τα μοντέλα παράλληλου προγραμματισμού **είναι ανεξάρτητα από την αρχιτεκτονική μνήμης**.
- Παράδειγμα:
  - Χρησιμοποιήθηκε το μοντέλο κοινής μνήμης σε ένα παράλληλο σύστημα κατανεμημένης μνήμης (Kendal Square Research). Η κατανεμημένη μνήμη εμφανίζονταν ως μια ενιαία μνήμη.
  - Χρησιμοποιήθηκε το μοντέλο κατανεμημένης μνήμης σε ένα παράλληλο σύστημα κοινής μνήμης (SGI Origin 2000). Η απόδοση ήταν καλύτερη με τη μετάδοση πληροφοριών με MPI παρά με τη CC-Numa.

Μοντέλο: Κοινή μνήμη (χωρίς νήματα)

- Κοινή ενιαία μνήμη.
- Ασύγχρονες εγγραφές και αναγνώσεις.
- Υποστηρίζονται σημαφόροι, κλειδώματα, κ.α.
- Δυσκολία στη διαχείριση της τοπικότητας, δηλαδή η διατήρηση των δεδομένων κοντά στη μνήμη που βρίσκεται στον επεξεργαστή που τα χρησιμοποιεί.
- Ευκολία υλοποίησης στα SMP συστήματα.
- Υλοποίηση με ειδικό hardware/software στα κατανεμημένα συστήματα.

Μοντέλο: Νήματα

- Κοινή μνήμη.
- Πολλαπλά νήματα ανά διεργασία.
- Συγχρονισμός με σημαφόρους, κλειδώματα, όριο.
- Δυο δημοφιλείς υλοποιήσεις:
  - Openmp
  - Posix Threads

Μοντέλο: Μεταβίβαση Μηνύματος

- Κάθε διεργασία έχει ιδιωτική μνήμη.
- Οι διεργασίες επικοινωνούν μεταξύ τους με μεταβίβαση μηνύματος.
- Η επικοινωνία απαιτεί ενέργειες και από το πομπό και το δέκτη.
- Η επικοινωνία γίνεται με τη κλήση κατάλληλων συναρτήσεων μιας βιβλιοθήκης.
- Ο προγραμματιστής ανακαλύπτει όλη την παραλληλία.
- Η δημοφιλής υλοποίηση είναι το MPI.

Μοντέλο: Παράλληλη Πρόσβαση Δεδομένων

- Υπάρχει ένα μεγάλο σετ δεδομένων εισόδου.
- Μια ομάδα από διεργασίες εκτελούν ενέργειες ανεξάρτητα πάνω σε αυτή την είσοδο.
- Όλες οι διεργασίες εκτελούν την ίδια συνάρτηση πάνω στα δεδομένα που επεξεργάζονται.
- Τα δεδομένα μπορεί να βρίσκονται είτε σε μια κοινή μνήμη είτε να βρίσκονται στην ιδιωτική τους μνήμη.
- Υλοποιούνται σε compilers οι οποίοι διαβάζουν το πως θα σταλούν τα δεδομένα και τοποθετούν τις κατάλληλες κλήσεις (π.χ. MPI) για να εξασφαλίσουν ότι ο κάθε επεξεργαστής έχει τα δεδομένα του προς επεξεργασία.

Μοντέλο: SPMD / MPMD

- High-End μοντέλα, τα οποία κατασκευάζονται από ένα ή περισσότερα μοντέλα από τα προηγούμενα.
- SPMD:
  - Single Program Multiple Data:
    - Όλες οι διεργασίες εκτελούν το ίδιο πρόγραμμα σε πολλαπλά δεδομένα. Υπάρχουν δομές if για να ρυθμίζουν τη ροή του προγράμματος, ώστε ο κάθε κόμβος να εκτελεί τις λειτουργίες του.
- MPMD:
  - Multiple Programs Multiple Data:
    - Πολλαπλά προγράμματα για πολλαπλά δεδομένα. Δε χρησιμοποιείται συχνά.

Ανάπτυξη Παράλληλων Εφαρμογών

- Η ανάπτυξη αλγορίθμων είναι μια κρίσιμη διαδικασία στην επίλυση προβλημάτων με χρήση υπολογιστών.
- **Σειριακός Αλγόριθμός**: Είναι μια αλληλουχία στοιχειωδών βημάτων που περιγράφουν τη λύση σε ένα συγκεκριμένο πρόβλημα χρησιμοποιώντας σειριακό υπολογιστή.
- **Παράλληλος Αλγόριθμος**: Είναι μια αλληλουχία στοιχειωδών βημάτων που περιγράφουν την λύση ένα συγκεκριμένο πρόβλημα χρησιμοποιώντας πολλούς επεξεργαστές.

Πως γίνεται η παραλληλοποίηση εφαρμογών;

- Η ανάπτυξη Παράλληλων Αλγορίθμων γίνεται συνήθως με την επεξεργασία ενός **υπάρχοντος σειριακού αλγόριθμου**. Η ανάπτυξη Παράλληλων Αλγορίθμων μπορεί να περιλαμβάνει κάποιο ή όλα από τα εξής σημεία:
  - Εντοπισμός των τμημάτων του κώδικα που μπορούν να εκτελεστούν.
  - Εντοπισμός των τμημάτων του κώδικα που μπορούν να εκτελεστούν παράλληλα.
  - Αντιστοίχιση των τμημάτων αυτών σε πολλαπλούς επεξεργαστές που εργάζονται παράλληλα.
  - Διανομή των δεδομένων εισόδου, εξόδου και ενδιάμεσων δεδομένων που σχετίζονται με το πρόγραμμα.
  - Συγχρονισμός των επεξεργαστών κατά τα διάφορα στάδια της παράλληλης εκτέλεσης.
  - Πρόσβαση στα δεδομένα από πολλαπλούς επεξεργαστές.

Κρίσιμα Τμήμα

- Τμήματα κώδικα που εκτελούνται παράλληλα ενδέχεται να χρησιμοποιούν “κρίσιμα τμήματα”.
- Το κρίσιμο τμήμα ενός προγράμματος, ονομάζεται η περιοχή του προγράμματος στην οποία γίνεται η ενημέρωση μιας κοινής περιοχής μνήμης από πολλαπλά νήματα.
- Ξεκινάει από την εντολή που διαβάζει την τρέχουσα τιμή της προς ενημέρωση περιοχή και τελειώνει αμέσως μετά την ενημέρωση αυτής της περιοχής.
- Απαιτείται αποκλειστική πρόσβαση σε αυτήν την περιοχή, με αμοιβαίο αποκλεισμό.

Κρίσιμα Τμήμα: Παράδειγμα

- Έστω υπάρχει το παρακάτω τμήμα κώδικα που εκτελείται ταυτόχρονα από πολλαπλές διεργασίες:

```
tempX=y;
tempX++;
y=tempX;
```

- Αν εκτελείται ταυτόχρονα από πολλαπλά νήματα, χωρίς αμοιβαίο αποκλεισμό, θα δημιουργηθούν προβλήματα.
- Πρέπει να υπάρχει αμοιβαίος αποκλεισμός, ώστε μόνο μια διεργασία να εισέρχεται μέσα στο τμήμα ως εξής:

```
mutex_lock(key)
tempX=y;
tempX++;
y=tempX;
mutex_unlock(key)
```

Τέλος Ενότητας