ΚΕΦΑΛΑΙΟ 17

ΠΑΡΑΛΛΗΛΗ ΕΠΕΞΕΡΓΑΣΙΑ

17.7 Υπολογισμοί Διανυσμάτων

Προσεγγίσεις Σχετικά με τους Υπολογισμούς Διανυσμάτων

Ο Μηχανισμός Επεξεργασίας Διανυσμάτων της Μηχανής IBM 3090

17.8 Προτεινόμενη Μελέτη και Δικτυακοί Τόποι

17.9 Όροι-κλειδιά, Ερωτήσεις Επανάληψης και Προβλήματα

ΒΑΣΙΚΑ ΣΗΜΕΙΑ

- Ένας κλασικός τρόπος αύξησης της απόδοσης ενός συστήματος, είναι η χρήση πολαπλών επεξεργαστών, οι οποίοι έχουν τη δυνατότητα να εκτελούν προγράμματα παράλληλα, ώστε να υποστηρίζουν ένα δεδομένο φορτίο εργασίας. Οι δύο πιο κοινές μορφές οργάνωσης πολλαπλών επεξεργαστών είναι οι **συμμετρικοί πολυεπεξεργαστές** (Symmetric Multiprocessors-SMPs) και οι συστάδες (Clusters). Προσφάτως, έχουν εισαχθεί στο εμπόριο. τα συστήματα **μη ομοιόμορφης προσπέλασης μνήμης** (Nonuniform Memory Access-NUMA).

- Ένα σύστημα SMP αποτελείται από πολλούς παρόμοιους επεξεργαστές ευρισκόμενους στον ίδιο υπολογιστή, οι οποίοι διασυνδέονται με ένα δίαυλο, ή με μία διάταξη μεταγωγέων. Το πιο σημαντικό πρόβλημα αυτού του συστήματος, είναι εκείνο της συνοχής της κρυφής μνήμης. Κάθε επεξεργαστής διαθέτει τη δική του κρυφή μνήμη, επομένως, είναι δυνατόν μία γραμμή δεδομένων να βρίσκεται σε περισσότερες από μία γραμμές της κρυφής μνήμης. Αν μία τέτοια γραμμή τροποποιηθεί σε μία κρυφή μνήμη, τότε τόσο η κύρια μνήμη όσο και οι άλλες κρυφές μνήμες έχουν μη έκδοση της γραμμής. Τα πρωτόκολλα συνοχής της κρυφής μνήμης είναι σχεδιασμένα για να αντιμετωπίζουν αυτό το πρόβλημα.

- Όταν περισσότεροι του ενός επεξεργαστές υλοποιούνται σε ένα chip, η διάταξη αναφέρεται ως **πολυεπεξεργασία σε επίπεδο chip** (Chip Multiprocessing). Μία σχετική τεχνική σχεδίασης, είναι να δημιουργήσουμε αντίγραφα ορισμένων στοιχείων ενός απλού επεξεργαστή, έτσι ώστε ο επεξεργαστής να είναι σε θέση να εκτελεί πολλαπλά νήματα ταυτόχρονα. Αυτός ο επεξεργαστής είναι γνωστός και ως **επεξεργαστής πολλαπλής νημάτωσης** (Multithreaded Processor).

- Η **συστάδα** είναι ένα σύνολο από διασυνδεδεμένους, αυτόνομους υπολογιστές, οι οποίοι εργάζονται μαζί ως ενοποιημένος υπολογιστικός πόρος, ο οποίος δημιουργεί την ψευδαίσθηση ότι είναι μία μηχανή. Ο όρος *αυτόνομος υπολογιστής*, δείχνει ακριβώς ότι κάθε υπολογιστικό σύστημα έχει τη δυνατότητα να εκτελεί από μόνο του, ανεξάρτητα της συστάδας.

- Ένα σύστημα NUMA αποτελεί έναν πολυεπεξεργαστή με κοινόχρηστη μνήμη, όπου ο χρόνος προσπέλασης μίας λέξης της μνήμης από έναν επεξεργαστή ποικίλει, αναλόγως με τη θέση αυτής της λέξης.

- Ένας ειδικός τύπος παράλληλης μορφής οργάνωσης είναι η διανυσματική διάταξη, η οποία είναι προσαρμοσμένη στην επεξεργασία διανυσμάτων ή πινάκων δεδομένων.

Παραδοσιακά, ο υπολογιστής εξεταζόταν ως ακολουθιακή μηχανή. Οι περισσότερες γλώσσες προγραμματισμού απαιτούν από τον προγραμματιστή να υλοποιεί αλγορίθμους, με τη μορφή ακολουθιών εντολών. Οι επεξεργαστές εκτελούν τα προγράμματα, εκτελώντας τις εντολές μηχανής με τη σειρά, μία κάθε φορά. Κάθε εντολή εκτελείται με μία σειρά λειτουργιών (ανάκληση εντολής, ανάκληση παραγόντων, εκτέλεση λειτουργίας, αποθήκευση αποτελεσμάτων).

Αυτή η θεώρηση του υπολογιστή δεν ήταν ποτέ εντελώς πραγματική. Στο επίπεδο μικρολειτουργιών, παράγονται ταυτόχρονα, πολλαπλά σήματα ελέγχου. Η διασωλήνωση των εντολών, τουλάχιστον στο επίπεδο επικάλυψης των λειτουργιών ανάκλησης και εκτέλεσης, υπάρχει εδώ και πολύ καιρό. Τα παραπάνω, αποτελούν παραδείγματα εκτέλεσης παράλληλων λειτουργιών. Αυτή η προσέγγιση ακολουθείται περαιτέρω στη πολυβαθμωτή οργάνωση, η οποία εκμεταλλεύεται τον παραλληλισμό σε επίπεδο εντολής. Με μία υπερβαθμωτή μηχανή, υπάρχουν πολλαπλές μονάδες εκτέλεσης εντός ενός επεξεργαστή και αυτές οι μονάδες πιθανόν να εκτελούν παράλληλα πολλαπλές εντολές του ίδιου προγράμματος.

Καθώς έχει εξελιχθεί η τεχνολογία των υπολογιστών και έχει μειωθεί το κόστος του υλικού, οι σχεδιαστές αναζητούν ολοένα και περισσότερες ευκαιρίες παραλληλισμού, συνήθως για να βελτιώσουν την απόδοση και, σε ορισμένες περιπτώσεις, για να αυξήσουν τη διαθεσιμότητα. Μετά από μία γενική θεώρηση, αυτό το κεφάλαιο εξετάζει ορισμένες από τις πιο σημαντικές προσεγγίσεις της παράλληλης οργάνωσης. Αρχικά, εξετάζονται οι συμμετρικοί πολυεπεξεργαστές (SMPs), οι οποίοι αποτελούν ένα από τα πρώτα και ακόμη και τώρα, πιο κοινά παραδείγματα παράλληλης οργάνωσης. Στην οργάνωση των συμμετρικών πολυεπεξεργαστών, πολλοί επεξεργαστές μοιράζονται μία κοινή μνήμη. Αυτή η μορφή οργάνωσης εγείρει το ζήτημα της συνοχής της κρυφής μνήμης, στο οποίο είναι αφιερωμένη μία ενότητα. Στη συνέχεια, περιγράφουμε τις συστάδες, οι οποίες αποτελούνται από πολλούς ανεξάρτητους επεξεργαστές, οι οποίοι είναι οργανωμένοι με τέτοιο τρόπο, ώστε να υπάρχει συνεργασία μεταξύ τους. Στη συνέχεια, το κεφάλαιο εξετάζει τους επεξεργαστές πολλαπλής νημάτωσης και τα chip πολυεπεξεργαστών. Οι συστάδες έχουν γίνει ιδιαίτερα δημοφιλείς για την υποστήριξη μεγάλων φορτίων εργασίας και βρίσκονται έξω από τη χωρητικότητα ενός απλού chip. Μία άλλη προσέγγιση της χρήσης πολλαπλών επεξεργαστών την οποία εξετάζουμε, είναι εκείνη των μηχανών μη ομοιόμορφης προσπέλασης μνήμης (NUMA). Η προσέγγιση αυτή είναι σχετικά νέα και δεν έχει καθιερωθεί στην αγορά, όμως συχνά θεωρείται ως εναλλακτική των SMPs και των συστάδων. Τέλος, το κεφάλαιο εξετάζει τις προσεγγίσεις της οργάνωσης του υλικού, οι οποίες αφορούν τους υπολογισμούς διανυσμάτων. Αυτές οι προσεγγίσεις βελτιστοποιούν την αριθμητική και λογική μονάδα, σε σχέση με την επεξεργασία διανυσμάτων ή πινάκων αριθμών κινητής υποδιαστολής και είναι συνηθισμένες σε κατηγορίες συστημάτων γνωστών ως *υπερεπεξεργαστές*.

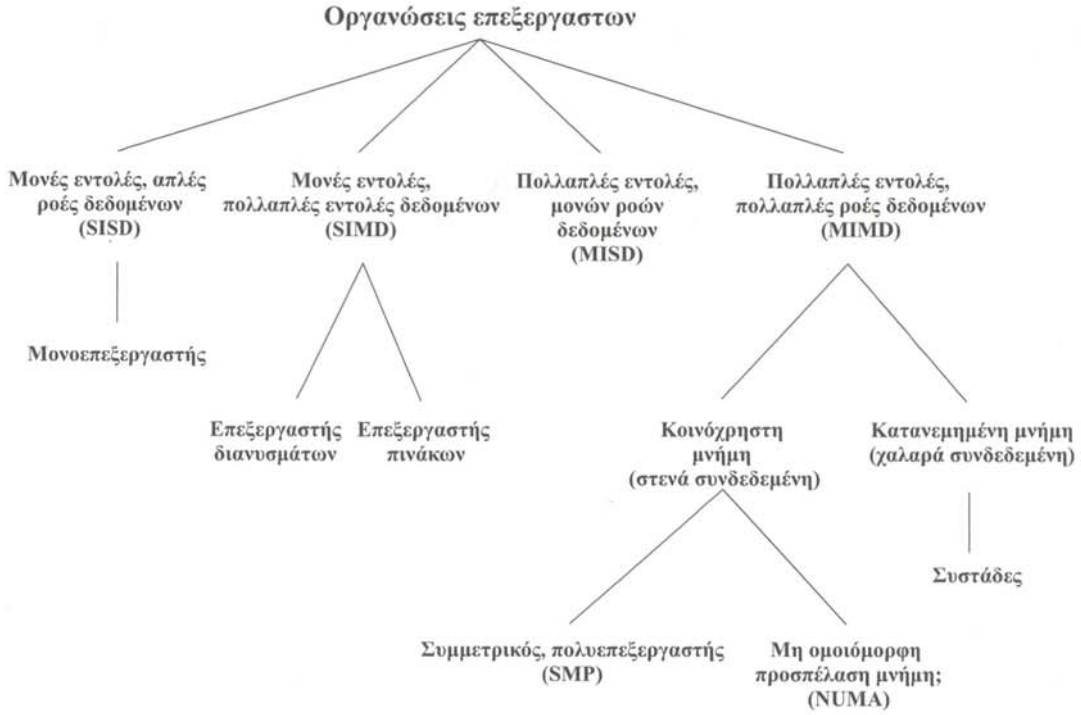
17.1 ΜΟΡΦΕΣ ΟΡΓΑΝΩΣΗΣ ΠΟΛΛΑΠΛΩΝ ΕΠΕΞΕΡΓΑΣΤΩΝ

Οργάνωση Παράλληλης Επεξεργασίας

Η ταξινόμηση την οποία εισήγαγε ο Flynn [FLYN72], εξακολουθεί να αποτελεί τον πιο συνηθισμένο τρόπο κατηγοριοποίησης των συστημάτων που έχουν τη δυνατότητα παράλληλης επεξεργασίας. Ο Flynn πρότεινε τις παρακάτω κατηγορίες υπολογιστικών συστημάτων:

- **Ροή μονής εντολής, μονών δεδομένων (Single Instruction, Single Data, SISD):** Ένας επεξεργαστής εκτελεί μία εντολή, η οποία λειτουργεί πάνω σε δεδομένα τα οποία είναι αποθηκευμένα σε μία μνήμη. Οι μονοεπεξεργαστές ανήκουν σε αυτή την κατηγορία.
- **Ροή μονής εντολής, πολλαπλών δεδομένων (Single Instruction, Multiple Data, SIMD):** Μία μονή εντολή μηχανής ελέγχει την ταυτόχρονη εκτέλεση ενός πλήθους στοιχείων επεξεργασίας. Κάθε στοιχείο επεξεργασίας διαθέτει μία συσχετιζόμενη μνήμη δεδομένων, ώστε κάθε εντολή να εκτελείται από διαφορετικούς επεξεργαστές, πάνω σε ένα διαφορετικό σύνολο δεδομένων. Οι επεξεργαστές διανυσμάτων και πινάκων ανήκουν σε αυτή την κατηγορία και περιγράφονται στην Ενότητα 18.7.
- **Ροή πολλαπλής εντολής εντολής, μονών δεδομένων (Multiple Instruction, Single Data, SISD):** Μία ακολουθία δεδομένων μεταφέρεται σε ένα σύνολο επεξεργαστών, καθένας εκ των οποίων εκτελεί μία διαφορετική ακολουθία εντολών. Αυτή η δομή δεν υλοποιείται εμπορικά.
- **Ροή πολλαπλής εντολής, πολλαπλών δεδομένων (Multiple Instruction, Multiple Data, MISD):** Ένα σύνολο από επεξεργαστές, οι οποίοι εκτελούν ταυτόχρονα διαφορετικές ακολουθίες εντολών σε διαφορετικά σύνολα δεδομένων. Σε αυτή την κατηγορία, ανήκουν οι συμμετρικοί πολυεπεξεργαστές, οι συστάδες, και τα συστήματα μη ομοιόμορφης προσπέλασης μνήμης.

Στην οργάνωση MIMD, οι επεξεργαστές είναι γενικού σκοπού. Καθένας από αυτούς, έχει τη δυνατότητα να επεξεργαστεί όλες τις εντολές οι οποίες είναι αναγκαίες ώστε να εκτελεστεί ο απαραίτητος



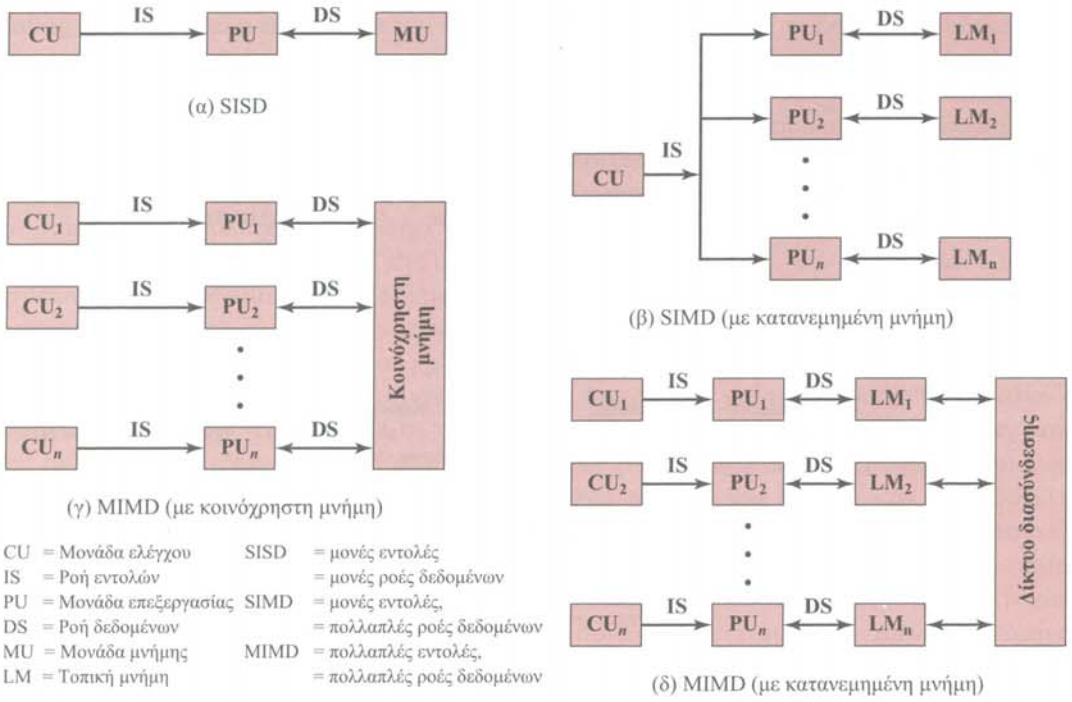
Σχήμα 17.1: Ταξινόμηση αρχιτεκτονικών παράλληλης επεξεργασίας

μετασχηματισμός των δεδομένων. Μία οργάνωση MIMD είναι δυνατόν να υποδιαιρεθεί περαιτέρω, με βάση τον τρόπο επικοινωνίας των επεξεργαστών (Σχήμα 17.1). Αν οι επεξεργαστές μοιράζονται μία κοινή μνήμη, τότε κάθε επεξεργαστής προσπελαύνει τα προγράμματα και τα δεδομένα, τα οποία αποθηκεύονται σε αυτή, και οι επεξεργαστές επικοινωνούν μεταξύ τους μέσω αυτής της μνήμης. Η πιο συνηθισμένη μορφή αυτού του συστήματος, είναι γνωστή και ως **συμμετρικός πολυεπεξεργαστής (SMP)**, και εξετάζεται στην Ενότητα 17.2. Σε ένα συμμετρικό πολυεπεξεργαστή, οι επεξεργαστές μοιράζονται μία μνήμη ή ένα σύνολο μνημών, με τη βοήθεια ενός κοινόχρηστου διαύλου ή ενός μέσου διασύνδεσης. Ένα χαρακτηριστικό για το οποίο διακρίνεται το σύστημα, είναι ότι ο χρόνος προσπέλασης κάθε περιοχής της μνήμης είναι περίπου ίδιος για κάθε επεξεργαστή. Μία πιο πρόσφατη εξέλιξη είναι η οργάνωση **μη ομοιόμορφης προσπέλασης μνήμης** (NUMA), η οποία περιγράφεται στην Ενότητα 17.5. Όπως δείχνει και το όνομα, ο χρόνος προσπέλασης σε διαφορετικές περιοχές της μνήμης ποικίλει μεταξύ των επεξεργαστών μίας μηχανής NUMA.

Ένα σύνολο από ανεξάρτητους μονοεπεξεργαστές ή συμμετρικούς πολυεπεξεργαστές, είναι δυνατόν να διασυνδεθούν σχηματίζοντας μία **συστάδα**. Η επικοινωνία ανάμεσα στους υπολογιστές γίνεται είτε μέσω σταθερών μονοπατιών, είτε μέσω κάποιας υπηρεσίας δικτύου.

Παράλληλες Οργανώσεις

Το Σχήμα 17.2 απεικονίζει τη γενική οργάνωση της ταξινόμησης την οποία συναντάμε στο Σχήμα 17.1. Το Σχήμα 17.2α απεικονίζει τη δομή μίας οργάνωσης SISD. Υπάρχει μία μορφή μονάδας ελέγχου (Control Unit-CU), η οποία παρέχει μία ροή εντολών (Instruction Stream-IS) σε μία μονάδα επεξεργασίας (Processing Unit-PU). Η μονάδα επεξεργασίας λειτουργεί πάνω σε μία απλή ροή δεδομένων (Data Stream-DS) τα οποία βρίσκονται σε μία μονάδα μνήμης (Memory Unit-MU). Σε μία οργάνωση SIMD, εξακολουθεί να υπάρχει μία μονάδα ελέγχου, η οποία τώρα τροφοδοτεί μία μονή ροή εντολών σε πολλαπλές μονάδες επεξεργασίας. Κάθε μονάδα επεξεργασίας διαθέτει τη δική της μνήμη (Σχήμα 17.2β), ή είναι δυνατόν να υπάρχει μία κοινή μνήμη. Τέλος, σε μία οργάνωση MIMD, υπάρχουν



Σχήμα 17.2: Εναλλακτικές οργανώσεις υπολογιστών

πολλαπλές μονάδες ελέγχου, κάθε μία από τις οποίες τροφοδοτεί μία ξεχωριστή ροή εντολών στη δική της μονάδα επεξεργασίας. Η MIMD πιθανόν να είναι ένας πολυεπεξεργαστής κοινής μνήμης (Σχήμα 17.2γ) ή ένας πολυ-υπολογιστής κατανεμημένης μνήμης (Σχήμα 17.2δ).

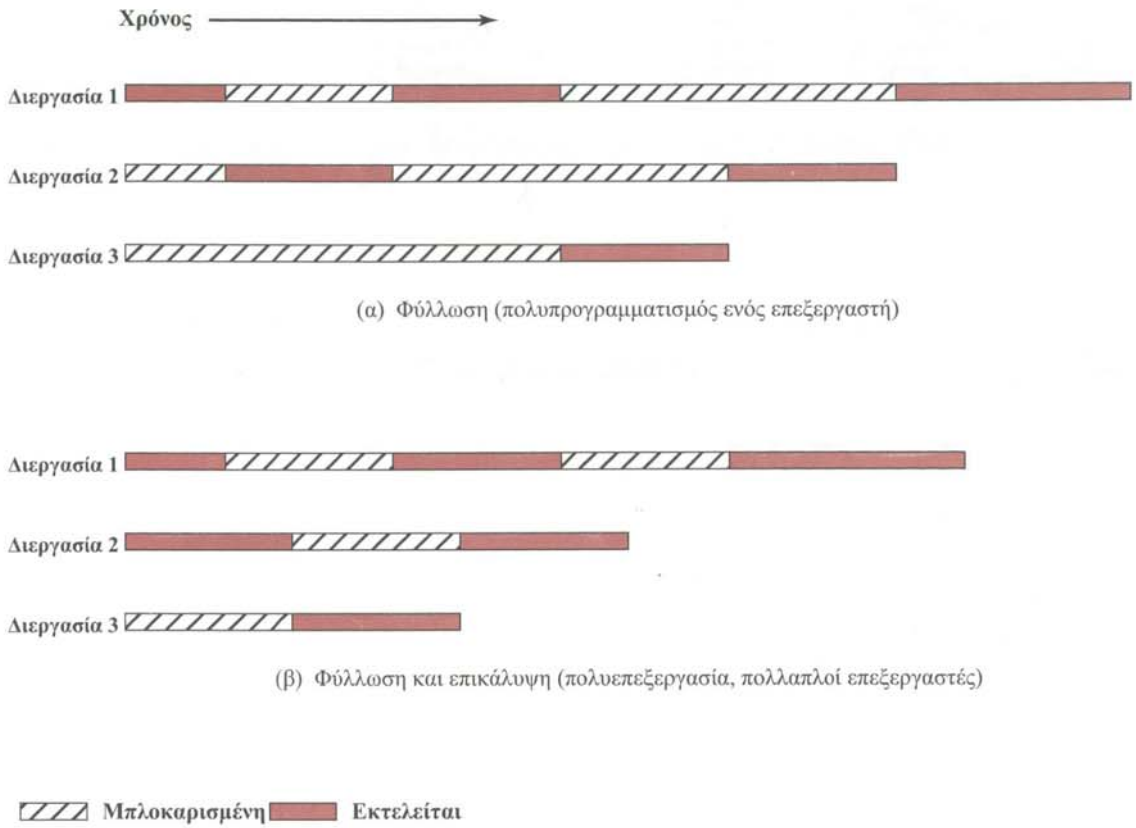
Τα ζητήματα σχεδίασης τα οποία σχετίζονται με τους συμμετρικούς πολυεπεξεργαστές, τις συστάδες, και τους υπολογιστές μη ομοιόμορφης προσπέλασης της μνήμης είναι πολύπλοκα και εμπεριέχουν ζητήματα τα οποία σχετίζονται με τη φυσική οργάνωση, τις δομές διασύνδεσης, την επικοινωνία ανάμεσα στους επεξεργαστές, τη σχεδίαση του λειτουργικού συστήματος, αλλά και τεχνικές υλοποίησης του λογισμικού εφαρμογών. Στο σημείο αυτό, το ενδιαφέρον μας είναι κυρίως η οργάνωση, παρά το γεγονός ότι εξετάζουμε επιδερμικά τα ζητήματα σχεδίασης του λειτουργικού συστήματος.

17.2 ΣΥΜΜΕΤΡΙΚΟΙ ΠΟΛΥΕΠΕΞΕΡΓΑΣΤΕΣ

Μέχρι πολύ πρόσφατα, ουσιαστικά όλοι οι προσωπικοί υπολογιστές ενός χρήστη και οι περισσότεροι σταθμοί εργασίας περιήχαν ένα μικροεπεξεργαστή γενικού σκοπού. Καθώς αυξανόταν οι απαιτήσεις σχετικά με την απόδοση και το κόστος των μικροεπεξεργαστών συνέχισε να πέφτει, οι κατασκευαστές εισήγαγαν συστήματα με οργάνωση συμμετρικών πολυεπεξεργαστών. Ο όρος *συμμετρικός πολυεπεξεργαστής* αναφέρεται σε μία αρχιτεκτονική του υλικού του υπολογιστή, αλλά και στη συμπεριφορά του λειτουργικού συστήματος η οποία αντανακλά αυτή την αρχιτεκτονική. Ένας συμμετρικός πολυεπεξεργαστής ορίζεται ως ένα αυτόνομο υπολογιστικό σύστημα με τα ακόλουθα χαρακτηριστικά:

1. Υπάρχουν δύο ή περισσότεροι παρόμοιοι επεξεργαστές με συγκρίσιμες δυνατότητες.

2. Αυτοί οι επεξεργαστές μοιράζονται την ίδια κοινή μνήμη και τις μονάδες Ε/Ε και συνδέονται μεταξύ τους με ένα δίαυλο ή κάποια άλλη εσωτερική τεχνική διασύνδεσης, έτσι ώστε ο χρόνος προσπέλασης της μνήμης να είναι περίπου παρόμοιος για κάθε επεξεργαστή.



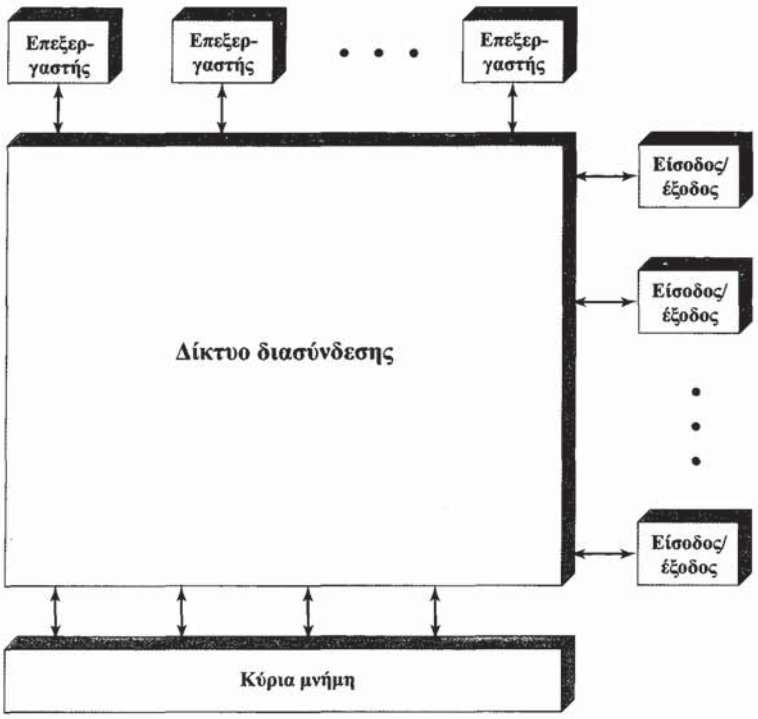
Σχήμα 17.3: Πολυπρογραμματισμός και πολυεπεξεργασία

3. Όλοι οι επεξεργαστές μοιράζονται την προσπέλαση σε μονάδες Ε/Ε, είτε μέσα από τα ίδια κανάλια, είτε μέσα από διαφορετικά κανάλια τα οποία παρέχουν μονοπάτια προς την ίδια μονάδα
4. Όλοι οι επεξεργαστές έχουν τη δυνατότητα να εκτελέσουν τις ίδιες λειτουργίες (για το λόγο αυτό υπάρχει ο όρος *συμμετρικοί*).
5. Το σύστημα ελέγχεται από ένα ολοκληρωμένο λειτουργικό σύστημα το οποίο παρέχει διασύνδεση ανάμεσα στους επεξεργαστές και τα προγράμματά τους, σε επίπεδο εργασίας, αρχείων, και στοιχείων δεδομένων.

Τα σημεία 1 ως 4 είναι αυταπόδεικτα. Το σημείο 5 τονίζει μία από τις αντιθέσεις που υπάρχουν, σε σύγκριση με τα λιγότερο συμπαγώς συνδεδεμένα συστήματα πολυεπεξεργασίας, όπως είναι οι συστάδες. Σε μία οργάνωση συστάδων, η φυσική μονάδα συναλλαγής είναι συνήθως ένα μήνυμα, ή ένα ολόκληρο αρχείο. Σε ένα σύστημα συμμετρικών πολυεπεξεργαστών, διαφορετικά στοιχεία δεδομένων είναι δυνατόν να συνιστούν το επίπεδο συναλλαγής και είναι δυνατόν να υπάρχει μεγάλος βαθμός συνεργασίας ανάμεσα στις διεργασίες.

Το λειτουργικό σύστημα ενός συστήματος συμμετρικών πολυεπεξεργαστών χρονοδρομολογεί τις διεργασίες ή νήματα (threads) και τις μοιράζει στους επεξεργαστές. Αυτή η μορφή οργάνωσης εμφανίζει ορισμένα πλεονεκτήματα σε σχέση με την οργάνωση ενός επεξεργαστή, τα οποία είναι τα εξής:

- **Απόδοση:** Αν η εργασία η οποία πρόκειται να υλοποιηθεί από έναν υπολογιστή είναι δυνατόν να οργανωθεί κατά τέτοιο τρόπο ώστε αυτή η υλοποίηση να είναι παράλληλη, τότε ένα σύστημα παράλληλων επεξεργαστών θα δώσει καλύτερα αποτελέσματα σε σύγκριση με ένα σύστημα ίδιου τύπου, το οποίο διαθέτει έναν επεξεργαστή (Σχήμα 17.3).



Σχήμα 17.4: Γενικό διάγραμμα μπλοκ ενός επεξεργαστή με συμπαγή σύνδεση

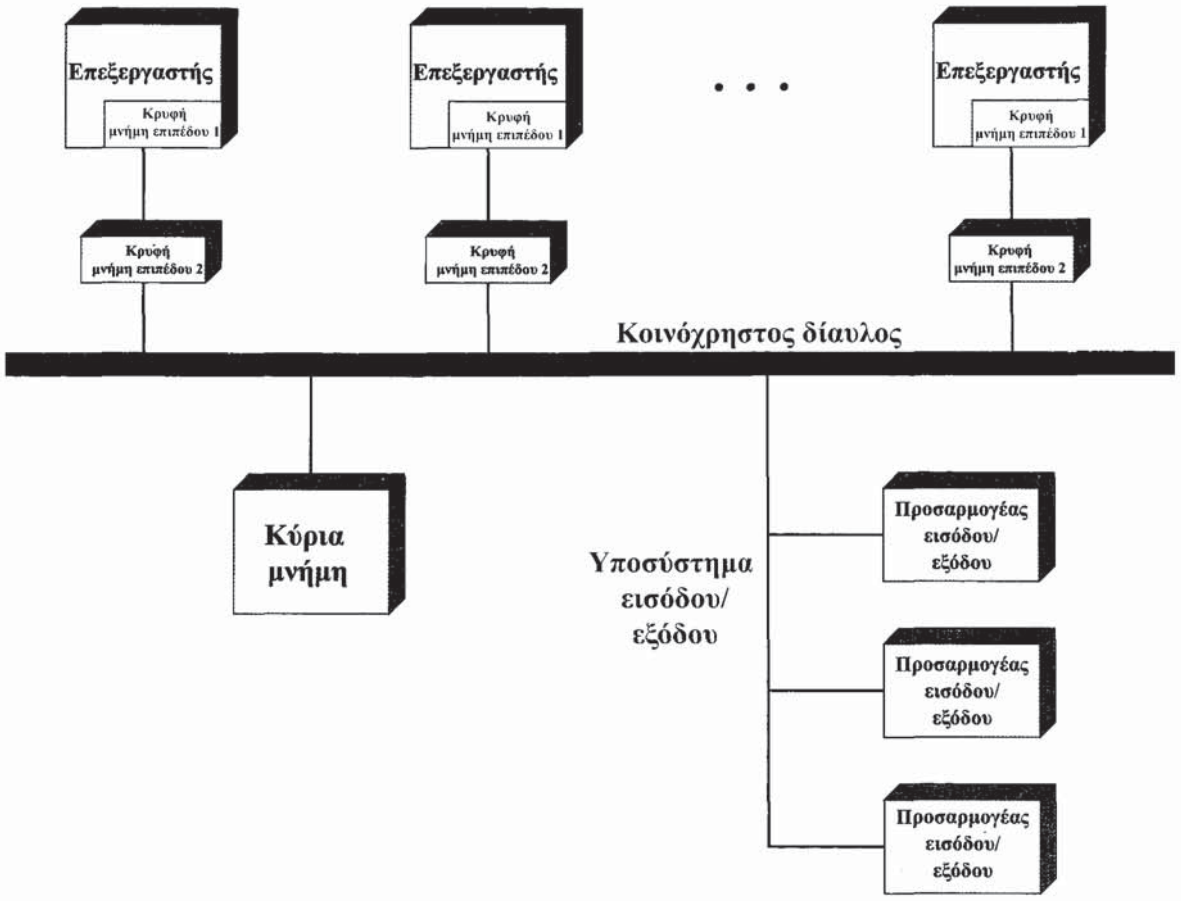
- **Διαθεσιμότητα:** Σε ένα συμμετρικό πολυεπεξεργαστή, επειδή όλοι οι επεξεργαστές έχουν τη δυνατότητα να εκτελέσουν τις ίδιες λειτουργίες, η βλάβη ενός επεξεργαστή δεν σταματά τη μηχανή. Αντίθετα, το σύστημα συνεχίζει να λειτουργεί με μειωμένη απόδοση.
- **Αυξημένη απόδοση:** Ο χρήστης έχει τη δυνατότητα να βελτιώσει την απόδοση ενός συστήματος προσθέτοντας έναν επιπλέον επεξεργαστή.
- **Κλιμάκωση:** Οι κατασκευαστές έχουν τη δυνατότητα να προσφέρουν μία ποικιλία προϊόντων με διαφορετικές τιμές και χαρακτηριστικά απόδοσης βάσει του πλήθους των επεξεργαστών του συστήματος.

Είναι σημαντικό να παρατηρήσει κανείς ότι, τα παραπάνω είναι εν δυνάμει ωφέλη και όχι εγγυημένα. Το λειτουργικό σύστημα θα πρέπει να παρέχει εργαλεία και λειτουργίες με τις οποίες θα εκμεταλλεύεται τον παραλληλισμό σε ένα σύστημα συμμετρικών πολυεπεξεργαστών.

Ένα ελκυστικό χαρακτηριστικό ενός συστήματος συμμετρικών πολυεπεξεργαστών είναι ότι η ύπαρξη των πολλαπλών επεξεργαστών δεν είναι ορατή στο χρήστη. Το λειτουργικό σύστημα αναλαμβάνει τη χρονοδρομολόγηση των διεργασιών ή νημάτων προς κάθε επεξεργαστή χωριστά και το συγχρονισμό ανάμεσα σε αυτούς τους επεξεργαστές.

Οργάνωση

Το Σχήμα 17.4 απεικονίζει σε γενικές γραμμές την οργάνωση ενός συστήματος πολυεπεξεργασίας. Υπάρχουν δύο ή περισσότεροι επεξεργαστές. Κάθε επεξεργαστής είναι αυτόνομος και συμπεριλαμβάνει μία μονάδα ελέγχου, μία αριθμητική και λογική μονάδα, καταχωρητές, και, τυπικά, ένα ή περισσότερα επίπεδα κρυφής μνήμης. Κάθε επεξεργαστής προσπελαύνει μία κοινόχρηστη μνήμη και τις μονάδες Ε/Ε μέσα από ένα μηχανισμό διασύνδεσης. Οι επεξεργαστές επικοινωνούν μεταξύ τους



Σχήμα 17.5: Οργάνωση συμμετρικού πολυεπεξεργαστή

μέσω της μνήμης (μηνύματα και πληροφορίες κατάστασης οι οποίες βρίσκονται σε κοινές περιοχές δεδομένων). Επίσης, υπάρχει δυνατότητα άμεσης ανταλλαγής σημάτων εκ μέρους των επεξεργαστών. Συχνά, η μνήμη είναι οργανωμένη κατά τέτοιο τρόπο ώστε να είναι εφικτές οι πολλαπλές ταυτόχρονες προσπελάσεις σε ξεχωριστά μπλοκ της μνήμης. Σε ορισμένες διατάξεις, κάθε επεξεργαστής πιθανόν να διαθέτει τη δική του ατομική κύρια μνήμη και κανάλια Ε/Ε, πέραν των κοινόχρηστων πόρων.

Η πιο κοινή μορφή οργάνωσης για προσωπικούς υπολογιστές, σταθμούς εργασίας, και εξυπηρέτες, είναι ο δίαυλος διαμοιραζόμενου χρόνου. Ο δίαυλος διαμοιραζόμενου χρόνου αποτελεί τον πιο απλό μηχανισμό κατασκευής ενός συστήματος πολυεπεξεργασίας (Σχήμα 17.5). Η δομή και οι διεπαφές είναι βασικά οι ίδιες με εκείνες ενός συστήματος το οποίο περιέχει έναν επεξεργαστή και χρησιμοποιεί μία διασύνδεση διαύλου. Ο δίαυλος αποτελείται από γραμμές ελέγχου, διευθύνσεων, και δεδομένων. Για να διευκολυνθούν οι μεταφορές άμεσης προσπέλασης μνήμης (DMA) από τους επεξεργαστές Ε/Ε, παρέχονται τα ακόλουθα γνωρίσματα:

- **Διευθυνσιοδότηση:** Θα πρέπει να υπάρχει δυνατότητα διάκρισης των μονάδων πάνω στο δίαυλο, ώστε να προσδιορίζουμε την αφετηρία και τον προορισμό των δεδομένων.

- **Διαιτησία:** Κάθε μονάδα Ε/Ε είναι δυνατόν να λειτουργήσει προσωρινά ως "αφέντης". Παρέχεται ένας μηχανισμός διαιτησίας ώστε να εξυπηρετούνται οι διαμαχόμενες αιτήσεις για τον έλεγχο του διαύλου, χρησιμοποιώντας μία τεχνική βασισμένη σε προτεραιότητες.

- **Διαμοιρασμός χρόνου:** Όταν μία μονάδα ελέγχει το δίαυλο, οι άλλες μονάδες είναι κλειδωμένες και θα πρέπει, αν αυτό είναι αναγκαίο, να αναστείλουν τη λειτουργία τους έως ότου λάβουν

εκ νέου πρόσβαση στο δίαυλο.

Αυτά τα χαρακτηριστικά ενός μονοεπεξεργαστή είναι άμεσα εφαρμόσιμα σε μία μορφή οργάνωσης συμμετρικών πολυεπεξεργαστών. Σε αυτή την περίπτωση, υπάρχουν πολλαπλοί επεξεργαστές και πολλαπλές μονάδες Ε/Ε που προσπαθούν να αποκτήσουν πρόσβαση σε μία ή περισσότερες μονάδες μνήμης μέσα από το δίαυλο.

Η οργάνωση του διαύλου έχει διάφορα ελκυστικά χαρακτηριστικά:

- **Απλότητα:** Πρόκειται για την πιο απλή προσέγγιση της οργάνωσης πολλαπλών επεξεργαστών. Στην περίπτωση αυτή, η φυσική διεπαφή και η λογική της διευθυνσιοδότησης, της διαιτησίας, και του διαμοιρασμού του χρόνου κάθε επεξεργαστή παραμένει ίδια, όπως σε ένα σύστημα απλού επεξεργαστή.

- **Ευελιξία:** Είναι γενικά εύκολο να επεκτείνουμε το σύστημα συνδέοντας περισσότερους επεξεργαστές στο δίαυλο.

- **Αξιοπιστία:** Ο δίαυλος είναι γενικά ένα παθητικό μέσο και η βλάβη οποιασδήποτε συνδεδεμένης συσκευής δεν θα πρέπει να προκαλεί βλάβη σε ολόκληρο το σύστημα.

Το βασικό μειονέκτημα της οργάνωσης του διαύλου, είναι η απόδοση. Όλες οι αναφορές στη μνήμη περνούν μέσα από τον κοινό δίαυλο. Επομένως, ο χρόνος του κύκλου διαύλου περιορίζει την ταχύτητα του συστήματος. Για να βελτιωθεί η απόδοση, είναι επιθυμητό ο κάθε επεξεργαστής να είναι εξοπλισμένος με μία κρυφή μνήμη. Αυτό το γεγονός θα προκαλέσει μεγάλη μείωση του πλήθους των προσπελάσεων του διαύλου. Τυπικά, οι συμμετρικοί πολυεπεξεργαστές έχουν δύο επίπεδα κρυφής μνήμης, όπου η κρυφή μνήμη Επιπέδου 1 είναι εσωτερική (στο chip όπου βρίσκεται ο επεξεργαστής) και η κρυφή μνήμη Επιπέδου 2 είναι είτε εσωτερική είτε εξωτερική. Ορισμένοι επεξεργαστές διαθέτουν και κρυφή μνήμη Επιπέδου 3.

Η χρησιμοποίηση της κρυφής μνήμης εισάγει νέα ζητήματα όσον αφορά τη σχεδίαση. Επειδή κάθε τοπική κρυφή μνήμη περιέχει μία απεικόνιση ενός τμήματος της κύριας μνήμης, αν αλλάξει μία λέξη σε μία κρυφή μνήμη, αυτομάτως ακυρώνεται μία λέξη σε μία άλλη κρυφή μνήμη. Για να αποτραπεί το γεγονός αυτό, θα πρέπει οι άλλοι επεξεργαστές να ειδοποιούνται ότι έχει συμβεί μία ενημέρωση. Αυτό το πρόβλημα είναι γνωστό ως *συνοχή κρυφής μνήμης* και τυπικά αντιμετωπίζεται σε επίπεδο υλικού παρά από το λειτουργικό σύστημα. Το πρόβλημα αυτό περιγράφεται στην Ενότητα 17.3.

Ζητήματα Σχεδίασης του Λειτουργικού Συστήματος των Πολλαπλών Επεξεργαστών

Το λειτουργικό σύστημα ενός συστήματος συμμετρικών πολυεπεξεργαστών διαχειρίζεται τους επεξεργαστές και τους υπόλοιπους πόρους του υπολογιστή, ώστε ο χρήστης να αντιλαμβάνεται ότι ένα μόνο λειτουργικό σύστημα διαχειρίζεται τους πόρους του συστήματος. Στην πραγματικότητα, αυτή η μορφή διάταξης θα πρέπει να εμφανίζεται ως σύστημα πολυπρογραμματισμού μονού επεξεργαστή. Σε αμφότερες τις περιπτώσεις, μονοεπεξεργαστή και συμμετρικών επεξεργαστών, υπάρχει πιθανότητα να είναι ενεργές πολλαπλές εργασίες ή διεργασίες και αποτελεί ευθύνη του λειτουργικού συστήματος να χρονοδρομολογήσει την εκτέλεσή τους και να κατανείμει τους πόρους. Ένας χρήστης μπορεί να υλοποιήσει εφαρμογές οι οποίες χρησιμοποιούν πολλαπλές διεργασίες ή πολλαπλά νήματα εντός διεργασιών, χωρίς να ενδιαφέρεται για το αν είναι διαθέσιμος ένας ή πολλοί επεξεργαστές. Επομένως, το λειτουργικό σύστημα ενός συστήματος πολλαπλών επεξεργαστών θα πρέπει να παρέχει κάθε λειτουργικότητα ενός συστήματος πολυπρογραμματισμού συν επιπλέον γνωρίσματα, ώστε να εξυπηρετεί τους πολλαπλούς επεξεργαστές. Ορισμένα από τα βασικά ζητήματα σχεδίασης είναι τα ακόλουθα:

- **Ταυτόχρονες συντρέχουσες διεργασίες:** Οι ρουτίνες του λειτουργικού συστήματος θα πρέπει να επανακαλούνται, ώστε να επιτρέπουν σε διάφορους επεξεργαστές να εκτελούν ταυτόχρονα τον κώδικα της ίδιας ροής εντολών. Όταν ένα πλήθος από επεξεργαστές εκτελούν το ίδιο ή διαφορετικά τμήματα του λειτουργικού συστήματος, οι πίνακες του τελευταίου, αλλά και οι διαχειριστικές του δομές θα πρέπει να χρησιμοποιούνται κατάλληλα, ώστε να μην προκαλούνται αδιέξοδα και άλλες μη έγκυρες λειτουργίες.

- **Χρονοδρομολόγηση:** Κάθε επεξεργαστής έχει τη δυνατότητα να χρονοδρομολογεί διεργασίες, επομένως θα πρέπει να αποφεύγονται οι διαμάχες. Ο χρονοδρομολογητής θα πρέπει να εκχωρεί έτοιμες διεργασίες σε διαθέσιμους επεξεργαστές.

- **Συγχρονισμός:** Με πολλαπλές ενεργές διεργασίες, οι οποίες έχουν τη δυνατότητα προσπέλασης σε κοινόχρηστους χώρους διευθύνσεων ή κοινόχρηστες μονάδες Ε/Ε, θα πρέπει να δοθεί προσοχή ώστε να παρέχεται αποτελεσματικός συγχρονισμός. Ο συγχρονισμός είναι μία υπηρεσία η οποία ενισχύει τον αμοιβαίο αποκλεισμό και την οργάνωση των γεγονότων σε μία σειρά.

- **Διαχείριση μνήμης:** Η διαχείριση της μνήμης σε ένα σύστημα πολλαπλών επεξεργαστών θα πρέπει να επικεντρώνεται σε όλα τα ζητήματα τα οποία συναντώνται σε μηχανές μονοεπεξεργαστών, όπως περιγράφηκαν στο Κεφάλαιο 8. Επιπλέον, το λειτουργικό σύστημα θα πρέπει να εκμεταλλευτεί το διαθέσιμο παραλληλισμό σε επίπεδο υλικού, όπως οι μνήμες πολλαπλών θυρών, για να επιτύχει μεγαλύτερη απόδοση. Οι μηχανισμοί σελιδοποίησης σε διαφορετικούς επεξεργαστές, θα πρέπει να συντονιστούν ώστε να ενισχύουν τη σταθερότητα, σε περιπτώσεις όπου πολλοί επεξεργαστές μοιράζονται μία σελίδα ή τμήμα και αποφασίσουν ότι πρέπει να γίνει αντικάτάσταση σελίδας.

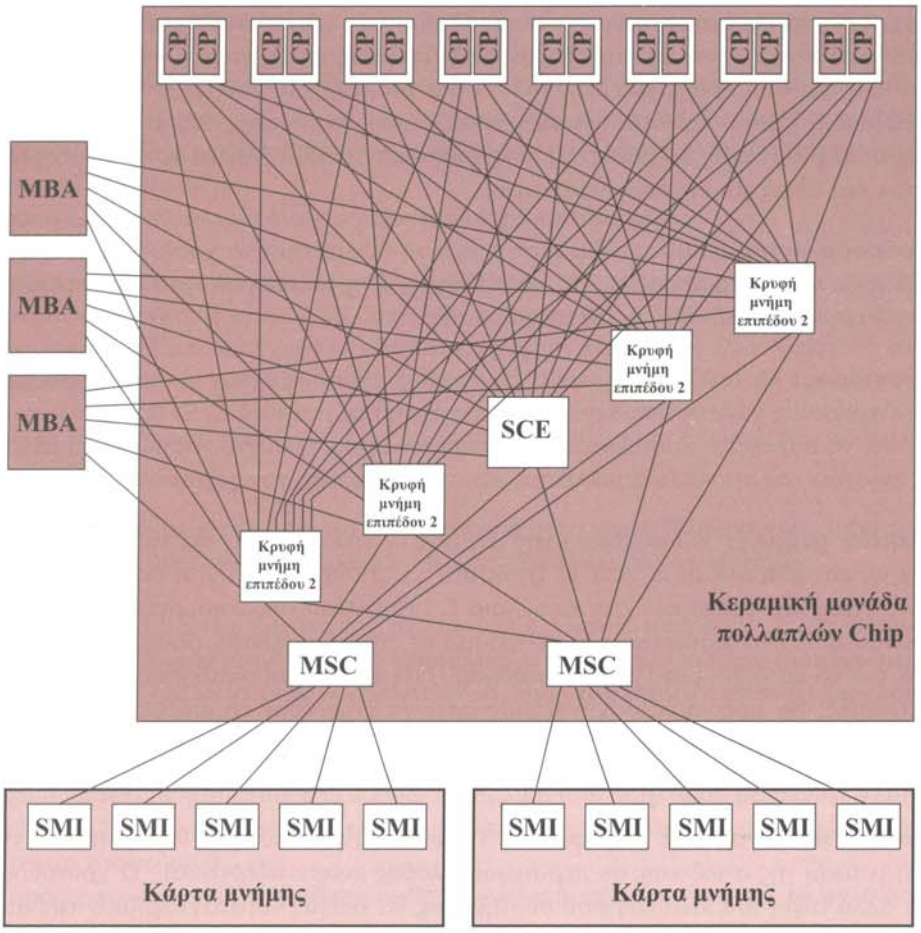
- **Αξιοπιστία και ανοχή σε σφάλματα:** Το λειτουργικό σύστημα θα πρέπει να παρέχει μία ανεκτή μείωση της απόδοσης σε περίπτωση βλάβης ενός επεξεργαστή. Ο χρονοδρομολογητής και τα άλλα μέρη του λειτουργικού συστήματος θα πρέπει να αναγνωρίζουν την απώλεια ενός επεξεργαστή και να ανακατασκευάζουν με κατάλληλο τρόπο τους πίνακες διαχείρισης.

Ένα Μεγάλο Υπολογιστικό Σύστημα SMP

Οι περισσότεροι συμμετρικοί πολυεπεξεργαστές χρησιμοποιούν μία στρατηγική διασύνδεσης διαύλου, η οποία απεικονίζεται στο Σχήμα 17.5. Είναι διδακτικό να εξετάσουμε μία εναλλακτική προσέγγιση, η οποία χρησιμοποιείται στην τρέχουσα υλοποίηση της οικογένειας μεγάλων υπολογιστικών συστημάτων zSeries της IBM [SIEG04, MAK04], με την ονομασία z990. Αυτή η οικογένεια συστημάτων εκτείνεται από συστήματα με ένα μονοεπεξεργαστή με μία κάρτα κύριας μνήμης μέχρι και συστήματα 48 επεξεργαστών με 8 κάρτες κύριας μνήμης. Τα βασικά χαρακτηριστικά της διάταξης αυτής απεικονίζονται στο Σχήμα 17.6:

- **Chip επεξεργαστή με δύο πυρήνες:** Κάθε chip επεξεργαστή συμπεριλαμβάνει δύο παρόμοιους κεντρικούς επεξεργαστές. Ο κάθε κεντρικός επεξεργαστής είναι υπερβαθμωτός με τεχνολογία CISC και οι περισσότερες εντολές του είναι προκατασκευασμένες. Οι υπόλοιπες εκτελούνται με κατακόρυφο μικροκώδικα. Κάθε κεντρικός επεξεργαστής συμπεριλαμβάνει μία κρυφή μνήμη εντολών και μία κρυφή μνήμη δεδομένων Επιπέδου 1 με χωρητικότητα 256 KB.

- **Κρυφή μνήμη Επιπέδου 2:** Κάθε κρυφή μνήμη Επιπέδου 2 έχει χωρητικότητα 32 MB. Οι κρυφές μνήμες Επιπέδου 2 διατάσσονται σε συστάδες αποτελούμενες από 5 μνήμες. Κάθε συστάδα υποστηρίζει 8 chips επεξεργαστών και παρέχει προσπέλαση σε ολόκληρο το χώρο της κύριας μνήμης.



CP = Κεντρικός επεξεργαστής
MBA = Προσαρμογέας διαύλου μνήμης
MSC = Έλεγχος κύριας αποθήκευσης
SCE = Στοιχείο ελέγχου συστήματος
SMI = Σύγχρονη διασύνδεση μνήμης

Σχήμα 17.6: Δομή του συστήματος πολλαπλών επεξεργαστών IBM z990

- **Στοιχείο ελέγχου του συστήματος (System Control Element-SCE):** Το στοιχείο ελέγχου συστήματος διαιτητεύει την επικοινωνία μέσα στο σύστημα και έχει ένα βασικό ρόλο στη διατήρηση της συνοχής της κρυφής μνήμης.

- **Έλεγχος κύριας μνήμης (Main Store Control-MSC):** Ο έλεγχος κύριας μνήμης διασυνδέει τις κρυφές μνήμες Επιπέδου 2 με την κύρια μνήμη.

- **Κάρτα μνήμης:** Κάθε κάρτα περιέχει 32 MB μνήμης. Η μέγιστη διάταξη μνήμης αποτελείται από 8 κάρτες, δίνοντας συνολική χωρητικότητα 256 MB. Οι κάρτες μνήμης διασυνδέονται με τον έλεγχο της κύριας μνήμης μέσα από σύγχρονες διεπαφές μνήμης (Synchronous Memory Interfaces-SMIs).

- **Προσαρμογέας διαύλου μνήμης (Memory Bus Adapter-MBA):** Ο προσαρμογέας διαύλου μνήμης παρέχει μία διεπαφή σε διάφορους τύπους καναλιών Ε/Ε. Η κυκλοφορία προς/από τα

κανάλια μεταφέρεται άμεσα στην κρυφή μνήμη επιπέδου 2.

Ο μικροεπεξεργαστής της οικογένειας z990 είναι σχετικά ασυνήθιστος σε σύγκριση με τους σύγχρονους επεξεργαστές, επειδή, παρά το γεγονός ότι είναι υπερβαθμωτός, εκτελεί εντολές με έναν αυστηρό τρόπο. Ωστόσο, αυτό αντισταθμίζεται από το γεγονός ότι έχει μικρότερη διασωλήνωση και πολύ μεγαλύτερες κρυφές μνήμες και παράπλευρους απομονωτές μετάφρασης σε σχέση με άλλους επεξεργαστές, ενώ παράλληλα διαθέτει και άλλα γνωρίσματα βελτίωσης της απόδοσης.

Το σύστημα z990 αποτελείται από 1 ως 4 **βιβλία**. Κάθε βιβλίο είναι μία μονάδα η οποία δύναται να ενσωματωθεί στο σύστημα και περιέχει 12 επεξεργαστές, έως 64 GB μνήμης, προσαρμογείς Ε/Ε, και ένα στοιχείο ελέγχου του συστήματος, το οποίο συνδέει τα άλλα στοιχεία. Μέσα σε κάθε βιβλίο, το στοιχείο ελέγχου συστήματος περιέχει μία κρυφή μνήμη Επιπέδου 2 με μέγεθος 32 MB, η οποία εξυπηρετεί ως κεντρικό σημείο συνοχής για το συγκεκριμένο βιβλίο. Τόσο η κρυφή μνήμη Επιπέδου 2, όσο και η κύρια μνήμη, είναι προσπελάσιμες από έναν επεξεργαστή ή έναν προσαρμογέα Ε/Ε εντός του βιβλίου, ή εντός των τριών άλλων βιβλίων του συστήματος. Τα chips του στοιχείου ελέγχου του συστήματος αλλά και των κρυφών μνημών επιπέδου 2 συνδέονται επίσης με αντίστοιχα στοιχεία που βρίσκονται στα άλλα βιβλία, με μία διάταξη δακτυλίου.

Η διάταξη του συστήματος συμμετρικών πολυεπεξεργαστών της οικογένειας z990 παρουσιάζει ορισμένα ενδιαφέροντα γνωρίσματα:

- Διασύνδεση με στοιχεία μεταγωγής
- Κοινές κρυφές μνήμες Επιπέδου 2

ΔΙΑΣΥΝΔΕΣΗ ΜΕ ΣΤΟΙΧΕΙΑ ΜΕΤΑΓΩΓΗΣ Ένας μονός κοινόχρηστος δίαυλος αποτελεί μία συνηθισμένη διάταξη σε συστήματα συμμετρικών πολυεπεξεργαστών (Σχήμα 17.5). Με αυτή τη διάταξη, ο μονός δίαυλος αποτελεί ένα στοιχείο πρόκλησης συμφορήσεων, το οποίο επηρρεάζει την κλιμάκωση (δηλαδή τη δυνατότητα επέκτασης σε μεγαλύτερα μεγέθη) της σχεδίασης. Η αρχιτεκτονική z990 αντιμετωπίζει αυτό το ζήτημα με δύο τρόπους. Πρώτον, η κύρια μνήμη διαχωρίζεται σε πολλαπλές κάρτες, κάθε μία από τις οποίες διαθέτει το δικό της ελεγκτή, ο οποίος έχει τη δυνατότητα να χειριστεί με μεγάλη ταχύτητα τις προσπελάσεις της μνήμης. Το μέσο φορτίο κυκλοφορίας προς την κύρια μνήμη περικόπτεται, λόγω της ύπαρξης ανεξάρτητων μονοπατιών προς ξεχωριστά μέρη της μνήμης. Κάθε βιβλίο περιέχει δύο κάρτες μνήμης, ώστε το συνολικό πλήθος των καρτών να είναι ίσο με 8, όταν η διάταξη είναι η μέγιστη δυνατή. Δεύτερον, η σύνδεση από τους επεξεργαστές (στην πραγματικότητα από τις κρυφές μνήμες Επιπέδου 2) προς μία κάρτα, δεν έχει τη μορφή ενός κοινόχρηστου διαύλου, αλλά συνδέσμων από σημείο σε σημείο. Κάθε chip επεξεργαστή έχει ένα σύνδεσμο προς κάθε μία από τις κρυφές μνήμες Επιπέδου 2 που βρίσκονται στο ίδιο βιβλίο, και κάθε κρυφή μνήμη Επιπέδου 2 έχει ένα σύνδεσμο, μέσα από τον έλεγχο της κύριας μνήμης (MSC), προς κάθε μία από τις δύο κάρτες μνήμης που βρίσκονται στο ίδιο βιβλίο.

Κάθε κρυφή μνήμη Επιπέδου 2 συνδέεται μόνον με τις δύο κάρτες μνήμης ενός βιβλίου. Ο ελεγκτής συστήματος παρέχει συνδέσμους (δεν απεικονίζονται) σε άλλα βιβλία της διάταξης, ώστε τελικά ολόκληρη η κύρια μνήμη να είναι προσπελάσιμη από όλους τους επεξεργαστές.

Οι σύνδεσμοι από σημείο σε σημείο και όχι ένας δίαυλος, είναι εκείνοι που παρέχουν επίσης διασυνδέσεις προς κανάλια Ε/Ε. Κάθε κρυφή μνήμη Επιπέδου 2 ενός βιβλίου συνδέεται με καθέναν από τους προσαρμογείς διαύλου μνήμης για το συγκεκριμένο βιβλίο. Με τη σειρά τους, οι προσαρμογείς συνδέονται με τα κανάλια Ε/Ε.

ΚΟΙΝΕΣ ΚΡΥΦΕΣ ΜΝΗΜΕΣ ΕΠΙΠΕΔΟΥ 2 Σε μία τυπική τεχνική κρυφής μνήμης Επιπέδου 2 ενός συστήματος συμμετρικών πολυεπεξεργαστών, κάθε επεξεργαστής διαθέτει μία δεσμευμένη κρυφή μνήμη Επιπέδου 1 και Επιπέδου 2. Προσφάτως, έχει αυξηθεί το ενδιαφέρον σχετικά με την κοινόχρηστη κρυφή μνήμη Επιπέδου 2. Σε μία προηγούμενη έκδοση του μεγάλου υπολογιστικού της

συστήματος συμμετρικών επεξεργαστών, γνωστό και ως Γενιά 3 (G3), η IBM χρησιμοποίησε δεσμευμένες κρυφές μνήμες. Στις επόμενες μηχανές (G4, G5, και z990), χρησιμοποιείται διαμοιραζόμενη κρυφή μνήμη Επιπέδου 2. Δύο ζητήματα προκάλεσαν αυτή την αλλαγή:

1. Κατά τη μετακίνηση από τη μηχανή G3 στη G4, η IBM διπλασίασε την ταχύτητα των μικροεπεξεργαστών. Αν διατηρούσε την οργάνωση της μηχανής G3, θα υπήρχε σημαντική αύξηση της κυκλοφορίας στο δίαυλο. Ταυτόχρονα, ήταν επιθυμητό να χρησιμοποιηθούν εκ νέου όσο το δυνατόν πιο πολλά από τα στοιχεία της G3. Χωρίς αναβάθμιση του διαύλου, αυτός θα αποτελούσε πηγή συμφορήσεων.

2. Η ανάλυση των τυπικών φορτίων του μεγάλου υπολογιστικού συστήματος αποκάλυψε ένα σημαντικό βαθμό κοινόχρηστων εντολών και δεδομένων μεταξύ των επεξεργαστών.

Αυτές οι θεωρήσεις οδήγησαν τους σχεδιαστές του συστήματος G4 στη χρησιμοποίηση μίας ή περισσότερων κρυφών μνημών Επιπέδου 2, κάθε μία από τις οποίες ήταν κοινόχρηστη σε ένα πλήθος πολλαπλών επεξεργαστών (κάθε επεξεργαστής διέθετε μία δεσμευμένη κρυφή μνήμη Επιπέδου 1 πάνω στο chip). Με πρώτη ματιά, η κοινή χρήση της κρυφής μνήμης Επιπέδου 2 φαίνεται κακή ιδέα. Η προσπέλαση της μνήμης από τους επεξεργαστές θα γίνει πιο αργή επειδή αυτοί θα πρέπει να ανταγωνίζονται για να προσπελάσουν μία κοινή κρυφή μνήμη Επιπέδου 2. Ωστόσο, αν οι επεξεργαστές μοιράζονται μία επαρκή ποσότητα δεδομένων, τότε μία κοινή κρυφή μνήμη αυξάνει τη ρυθμαπόδοση και δεν την επιβραδύνει. Τα δεδομένα τα οποία είναι κοινά και βρίσκονται στην κοινή κρυφή μνήμη, λαμβάνονται πιο γρήγορα σε σύγκριση με την περίπτωση λήψης τους μέσω του διαύλου.

17.3 ΣΥΝΟΧΗ ΚΡΥΦΗΣ ΜΝΗΜΗΣ ΚΑΙ ΤΟ ΠΡΩΤΟΚΟΛΛΟ MESI

Στα σύγχρονα συστήματα πολλαπλών επεξεργαστών, είναι συνηθισμένο να υπάρχει ένα ή δύο επίπεδα κρυφής μνήμης τα οποία συνδέονται με κάθε επεξεργαστή. Αυτή η οργάνωση είναι σημαντική για την προσπάθεια επίτευξης απόδοσης σε λογικά επίπεδα. Από την άλλη, δημιουργεί ένα πρόβλημα, το οποίο είναι γνωστό ως πρόβλημα *συνοχής της κρυφής μνήμης*. Η ουσία αυτού του προβλήματος είναι η εξής: Είναι δυνατόν να υπάρχουν πολλαπλά αντίγραφα των ίδιων δεδομένων, ταυτόχρονα σε πολλές κρυφές μνήμες. Αν οι επεξεργαστές επιτρέπεται να ενημερώνουν ελεύθερα τα δικά τους αντίγραφα, τότε είναι δυνατόν να προκύψει μία ασύμφωνη εικόνα της κύριας μνήμης. Στς Κεφάλαιο 4 παρουσιάστηκαν δύο συνηθισμένες πολιτικές εγγραφής:

- **Εγγραφή προς τα πίσω:** Συνήθως, οι λειτουργίες εγγραφής εκτελούνται μόνον προς την κρυφή μνήμη. Η κύρια μνήμη ενημερώνεται μόνον όταν διαγράφεται η αντίστοιχη γραμμή της κρυφής μνήμης.

- **Δια μέσου εγγραφή:** Όλες οι λειτουργίες εγγραφών γίνονται στην κύρια μνήμη καθώς και στην κρυφή, εξασφαλίζοντας την εγκυρότητα των δεδομένων της κύριας μνήμης.

Είναι φανερό ότι μία πολιτική εγγραφής προς τα πίσω, είναι δυνατόν να οδηγήσει σε ασυμφωνίες. Αν δύο κρυφές μνήμες περιέχουν την ίδια γραμμή και η γραμμή ενημερωθεί μόνον στη μία, η άλλη θα περιέχει μη έγκυρη τιμή, χωρίς να το γνωρίζει. Οι διαδοχικές λειτουργίες ανάγνωσης της μη έγκυρης γραμμής, θα έχουν ως συνέπεια να υπάρχουν μη έγκυρα αποτελέσματα. Ακόμη και στην πολιτική της εν δια μέσου εγγραφής, είναι δυνατόν να εμφανιστούν ασυμφωνίες, εκτός αν άλλες κρυφές μνήμες παρακολουθούν την κυκλοφορία της κύριας μνήμης ή λαμβάνουν κάποια άμεση ειδοποίηση σχετικά με την ενημέρωση.

Σε αυτή την ενότητα, θα μελετήσουμε σύντομα διάφορες προσεγγίσεις αναφορικά με το πρόβλημα της συνοχής της κρυφής μνήμης και στη συνέχεια, θα επικεντρωθούμε στην πιο διαδεδομένη

προσέγγιση: το πρωτόκολλο MESI (modified/exclusive/shared/invadid). Μία έκδοση αυτού του πρωτοκόλλου χρησιμοποιείται από υλοποιήσεις του Pentium 4 και του PowerPC.

Για κάθε πρωτόκολλο συνοχής της κρυφής μνήμης, ο στόχος είναι να επιτραπεί σε πρόσφατα χρησιμοποιημένες τοπικές μεταβλητές να εισέλθουν στην κατάλληλη κρυφή μνήμη και να παραμείνουν εκεί για ένα πλήθος από λειτουργίες ανάγνωσης και εγγραφής, ενώ χρησιμοποιείται το πρωτόκολλο ώστε να διατηρηθεί η συμφωνία μεταξύ των κοινών μεταβλητών, οι οποίες είναι πιθανό να βρίσκονται ταυτόχρονα σε πολλές κρυφές μνήμες. Οι προσεγγίσεις σχετικά με τη συνοχή της κρυφής μνήμης διαιρούνται γενικά σε προσεγγίσεις λογισμικού και υλικού. Ορισμένες υλοποιήσεις υιοθετούν μία στρατηγική η οποία χρησιμοποιεί τόσο στοιχεία υλικού όσο και στοιχεία λογισμικού. Παρά το γεγονός αυτό, η κατηγοριοποίηση σε προσεγγίσεις υλικού και λογισμικού εξακολουθεί να είναι διδακτική και χρησιμοποιείται συχνά στη μελέτη στρατηγικών συνοχής της κρυφής μνήμης.

Λύσεις Λογισμικού

Οι λύσεις του προβλήματος της συνοχής της κρυφής μνήμης οι οποίες βασίζονται στο λογισμικό, επιδιώκουν να αποφύγουν την ανάγκη χρησιμοποίησης επιπλέον κυκλωμάτων και λογικής, βασιζόμενες στο λειτουργικό σύστημα και στο μεταγλωττιστή. Οι προσεγγίσεις λογισμικού είναι ελκυστικές, λόγω του γεγονότος ότι η επιβάρυνση που προκύπτει λόγω της ανίχνευσης πιθανών προβλημάτων, μεταφέρεται από το χρόνο εκτέλεσης στο χρόνο μεταγλώττισης, ενώ η πολυπλοκότητα σχεδίασης μεταφέρεται από το υλικό στο λογισμικό. Από την άλλη, οι προσεγγίσεις λογισμικού οι οποίες σχετίζονται το χρόνο μεταγλώττισης, θα πρέπει σε γενικές γραμμές να λαμβάνουν συντηρητικές αποφάσεις, κάτι που οδηγεί σε μη αποτελεσματική χρήση της κρυφής μνήμης.

Οι μηχανισμοί συνοχής που βασίζονται στο μεταγλωττιστή, αναλύουν τον κώδικα για να προσδιορίσουν ποια αντικείμενα δεδομένων είναι πιθανό να καταστούν ανασφαλή προκειμένου να τοποθετηθούν στην κρυφή μνήμη και τα μαρκάρουν κατάλληλα. Στη συνέχεια, το λειτουργικό σύστημα ή το υλικό αποτρέπει την τοποθέτηση αυτών των αντικειμένων δεδομένων στην κρυφή μνήμη.

Η πιο απλή προσέγγιση, είναι να απαγορεύεται η τοποθέτηση στην κρυφή μνήμη κάθε κοινόχρηστης μεταβλητής δεδομένων. Η προσέγγιση αυτή είναι υπερβολικά συντηρητική, επειδή μία κοινόχρηστη δομή δεδομένων είναι πιθανό να χρησιμοποιείται αποκλειστικά για ένα συγκεκριμένο χρονικό διάστημα και να χρησιμοποιείται μόνον για λειτουργίες ανάγνωσης και μάλιστα με αποτελεσματικό τρόπο σε άλλα χρονικά διαστήματα. Η συνοχή αποτελεί ζήτημα μόνον κατά τη διάρκεια περιόδων όπου τουλάχιστον μία διεργασία είναι δυνατόν να ενημερώνει κάποια μεταβλητή και μία άλλη να την προσπελαύνει.

Πιο αποτελεσματικές προσεγγίσεις αναλύουν τον κώδικα για να προσδιορίσουν ασφαλείς περιόδους για τις κοινόχρηστες μεταβλητές. Στη συνέχεια, ο μεταγλωττιστής εισάγει εντολές στον κώδικα που παράγεται, για να ενισχύσει τη συνοχή της κρυφής μνήμης κατά τη διάρκεια κρίσιμων περιόδων. Έχει αναπτυχθεί ένα πλήθος από τεχνικές υλοποίησης αυτής της ανάλυσης και ενίσχυσης των αποτελεσμάτων. Δείτε τις μελέτες [LILJ93] και [STEN90].

Λύσεις Υλικού

Οι λύσεις οι οποίες βασίζονται στο υλικό, αναφέρονται γενικά ως πρωτόκολλα συνοχής της κρυφής μνήμης. Αυτές οι λύσεις παρέχεουν δυναμική αναγνώριση, κατά τη διάρκεια εκτέλεσης, των συνθηκών οι οποίες είναι πιθανό να προκαλέσουν ασυμφωνίες. Λόγω του γεγονότος ότι αυτό το πρόβλημα αντιμετωπίζεται μόνον τη στιγμή κατά την οποία στην πραγματικότητα εμφανίζεται, γίνεται πιο αποτελεσματική χρήση των κρυφών μνημών, κάτι που οδηγεί σε βελτιωμένη απόδοση σε σύγκριση με την προσέγγιση που βασίζεται στο λογισμικό. Επιπλέον, αυτές οι προσεγγίσεις δεν είναι ορατές στον προγραμματιστή και στο μεταγλωττιστή, κάτι που μειώνει την επιβάρυνση της ανάπτυξης λογισμικού.

Οι τεχνικές που βασίζονται στο υλικό διαφέρουν σε ένα πλήθος από σημεία, συμπεριλαμβανομένης της θέσης όπου αποθηκεύονται οι πληροφορίες κατάστασης που αφορούν τις γραμμές δεδομένων, του τρόπου οργάνωσης των πληροφοριών, των σημείων όπου ενισχύεται η συνοχή, αλλά και των μηχανισμών ενίσχυσης. Σε γενικές γραμμές, οι τεχνικές οι οποίες βασίζονται στο υλικό χωρίζονται σε δύο κατηγορίες: πρωτόκολλα καταλόγου και αδιάκριτα πρωτόκολλα.

ΠΡΩΤΟΚΟΛΛΑ ΚΑΤΑΛΟΓΟΥ Τα πρωτόκολλα καταλόγου συλλέγουν και διατηρούν τις πληροφορίες σχετικά με τη θέση στην οποία αποθηκεύονται αντίγραφα των γραμμών. Τυπικά, υπάρχει ένας κεντρικός ελεγκτής, ο οποίος είναι μέρος του ελεγκτή της κύριας μνήμης και ένας κατάλογος ο οποίος αποθηκεύεται στην κύρια μνήμη. Ο κατάλογος περιέχει γενικές πληροφορίες κατάστασης σχετικά με τα περιεχόμενα των διαφόρων τοπικών κρυφών μνημών. Όταν ένας ελεγκτής κρυφής μνήμης κάνει μία αίτηση, ο κεντρικός ελεγκτής ελέγχει και εκδίδει τις κατάλληλες εντολές μεταφοράς δεδομένων ανάμεσα στη μνήμη και τις κρυφές μνήμες, ή ανάμεσα στις κρυφές μνήμες. Επίσης, είναι υπεύθυνος για να διατηρεί ενημερωμένες τις πληροφορίες κατάστασης. Επομένως, κάθε ενέργεια σε τοπικό επίπεδο η οποία έχει τη δυνατότητα να επηρρεάσει τη γενικότερη κατάσταση μίας γραμμής, θα πρέπει να αναφέρεται στην κεντρικό ελεγκτή.

Τυπικά, ο ελεγκτής διατηρεί πληροφορίες σχετικά με το ποιοι επεξεργαστές έχουν αντίγραφα και από ποιες γραμμές. Πριν ένας επεξεργαστής μπορέσει να γράψει σε ένα τοπικό αντίγραφο μίας γραμμής, θα πρέπει να αιτηθεί από τον ελεγκτή άδεια αποκλειστικής χρήσης της γραμμής. Πριν εκχωρήσει την άδεια, ο ελεγκτής στέλνει ένα μήνυμα προς όλους τους επεξεργαστές, με ένα αντίγραφο αυτής της γραμμής, αναγκάζοντας κάθε επεξεργαστή να ακυρώσει αυτό το αντίγραφο. Μετά τη λήψη της επιβεβαίωσης από κάθε επεξεργαστή, ο ελεγκτής αποδέχεται την αποκλειστική προσπέλαση την οποία αιτήθηκε ένας επεξεργαστής. Όταν ένας επεξεργαστής επιδιώξει να διαβάσει μία γραμμή η οποία έχει δοθεί για αποκλειστική προσπέλαση σε έναν άλλο επεξεργαστή, θα στείλει μία ειδοποίηση αστοχίας (miss) προς τον ελεγκτή. Στη συνέχεια, ο ελεγκτής εκδίδει μία εντολή στον επεξεργαστή ο οποίος προσπελαύνει αυτή τη γραμμή, με την οποία απαιτεί από αυτόν να εκτελέσει μία λειτουργία εγγραφής προς τα πίσω, στην κύρια μνήμη. Τώρα, η γραμμή είναι δυνατόν να μοιραστεί για ανάγνωση, τόσο από τον αρχικό επεξεργαστή, όσο και από τον εκείνο που έκανε την αίτηση.

Τα πρωτόκολλα καταλόγου εμφανίζουν τα μειονεκτήματα της κεντρικής συμφόρησης και της επιβάρυνσης λόγω της επικοινωνίας ανάμεσα σε διάφορους ελεγκτές κρυφών μνημών και τον κεντρικό ελεγκτή. Ωστόσο, είναι αποτελεσματικά σε συστήματα υψηλής κλιμάκωσης, τα οποία χρησιμοποιούν πολλούς διαύλους ή κάποιο πολύπλοκο σύστημα διασύνδεσης.

ΑΔΙΑΚΡΙΤΑ ΠΡΩΤΟΚΟΛΛΑ Τα αδιάκριτα πρωτόκολλα κατανέμουν την ευθύνη για τη διατήριση της συνοχής της κρυφής μνήμης ανάμεσα σε όλους τους ελεγκτές κρυφών μνημών ενός πολυεπεξεργαστή. Μία κρυφή μνήμη θα πρέπει να αναγνωρίζει πότε μία γραμμή την οποία αποθηκεύει, είναι κοινή με άλλες κρυφές μνήμες. Όταν γίνει μία λειτουργία ενημέρωσης σε μία κοινή γραμμή κρυφής μνήμης, θα πρέπει να ανακοινωθεί σε όλες τις κρυφές μνήμες με κάποιο μηχανισμό. Κάθε ελεγκτής κρυφής μνήμης έχει τη δυνατότητα να "εξετάσει αδιάκριτα" το δίκτυο, προκειμένου να παρατηρήσει αυτές τις ειδοποιήσεις και να ενεργήσει ανάλογα.

Τα αδιάκριτα πρωτόκολλα είναι ιδανικά για πολυεπεξεργαστές που βασίζουν την επικοινωνία τους σε διαύλους, επειδή ο κοινός δίαυλος παρέχει έναν απλό τρόπο εκπομπής και εξέτασης. Ωστόσο, επειδή ένας από τους στόχους της χρήσης τοπικών κρυφών μνημών είναι η αποφυγή των προσπελάσεων του διαύλου, θα πρέπει να προσεχθεί ότι η αυξημένη κυκλοφορία στο δίαυλο, η οποία χρειάζεται για την εκπομπή αλλά και την αδιάκριτη εξέταση, δεν ακυρώνει τα κέρδη που υπάρχουν από τη χρήση των τοπικών κρυφών μνημών.

Έχουν μελετηθεί δύο βασικές προσεγγίσεις όσον αφορά τα αδιάκριτα πρωτόκολλα: ακύρωση εγγραφής και ενημέρωση εγγραφής (ή εκπομπή εγγραφής). Σε ένα πρωτόκολλο ακύρωσης εγγραφής, είναι δυνατόν να υπάρχουν αρκετές λειτουργίες ανάγνωσης, αλλά μία μόνον λειτουργία εγγραφής.

Αρχικά, μία γραμμή μοιράζεται ανάμεσα σε αρκετές κρυφές μνήμες, προκειμένου να διαβαστεί. Όταν μία από τις κρυφές μνήμες επιθυμεί να εκτελέσει μία λειτουργία εγγραφής σε αυτή τη γραμμή, τότε εκδίδει μία ειδοποίηση με την οποία ακυρώνει τη γραμμή αυτή στις άλλες μνήμες, με αποτέλεσμα η γραμμή να διατίθεται αποκλειστικά στη συγκεκριμένη κρυφή μνήμη. Από τη στιγμή που θα συμβεί αυτό, ο επεξεργαστής στον οποίο ανήκει η κρυφή μνήμη μπορεί να εκτελέσει τοπικά διάφορες εγγραφές, έως ότου η γραμμή ζητηθεί από έναν άλλο επεξεργαστή.

Σε ένα πρωτόκολλο ενημέρωσης εγγραφής είναι δυνατόν να υπάρχουν αρκετές λειτουργίες εγγραφής, αλλά και ανάγνωσης. Όταν ένας επεξεργαστής επιθυμεί να ενημερώσει μία κοινή γραμμή, η λέξη η οποία θα ενημερωθεί διανέμεται σε όλους τους άλλους επεξεργαστές, οπότε οι κρυφές μνήμες οι οποίες περιέχουν αυτή τη γραμμή μπορούν να την ενημερώσουν.

Καμία από τις δύο αυτές προσεγγίσεις δεν είναι ανώτερη της άλλης κάτω από κάθε συνθήκη. Η απόδοση εξαρτάται από το πλήθος των τοπικών κρυφών μνημών και από το υπόδειγμα των λειτουργιών εγγραφής και ανάγνωσης της μνήμης. Ορισμένα συστήματα υλοποιούν προσαρμοστικά πρωτόκολλα τα οποία χρησιμοποιούν τόσο μηχανισμούς ακύρωσης, όσο και μηχανισμούς ενημέρωσης εγγραφής.

Η προσέγγιση ακύρωσης εγγραφής είναι η πιο διαδεδομένη στους εμπορικούς υπολογιστές πολλαπλών επεξεργαστών, όπως είναι ο Pentium 4 ή ο PowerPC. Η κατάσταση κάθε γραμμής της κρυφής μνήμης μαρκάρεται (χρησιμοποιώντας δύο επιπλέον bits ετικέτας της κρυφής μνήμης), ως τροποποιημένη Modified, αποκλειστική Exclusive, κοινόχρηστη Shared, ή μη έγκυρη Invalid. Από τα αρχικά γράμματα των αγγλικών όρων, προέκυψε η ονομασία του πρωτοκόλλου ακύρωσης εγγραφής, MESI. Στο υπόλοιπο αυτής της ενότητας, θα εξετάσουμε τη χρήση του πρωτοκόλλου ανάμεσα σε διάφορες κρυφές μνήμες ενός συστήματος πολλών επεξεργαστών. Για λόγους απλούστευσης, δεν εξετάζουμε τους μηχανισμούς οι οποίοι εμπλέκονται στο συντονισμό μεταξύ των Επιπέδων 1 και 2 σε τοπικό επίπεδο και ταυτόχρονα στο συντονισμό εντός των κατανεμημένων επεξεργαστών. Κάτι τέτοιο, δεν θα πρόσθετε νέες αρχές, αλλά θα έκανε εξαιρετικά πολύπλοκη τη συζήτηση.

Το Πρωτόκολλο MESI

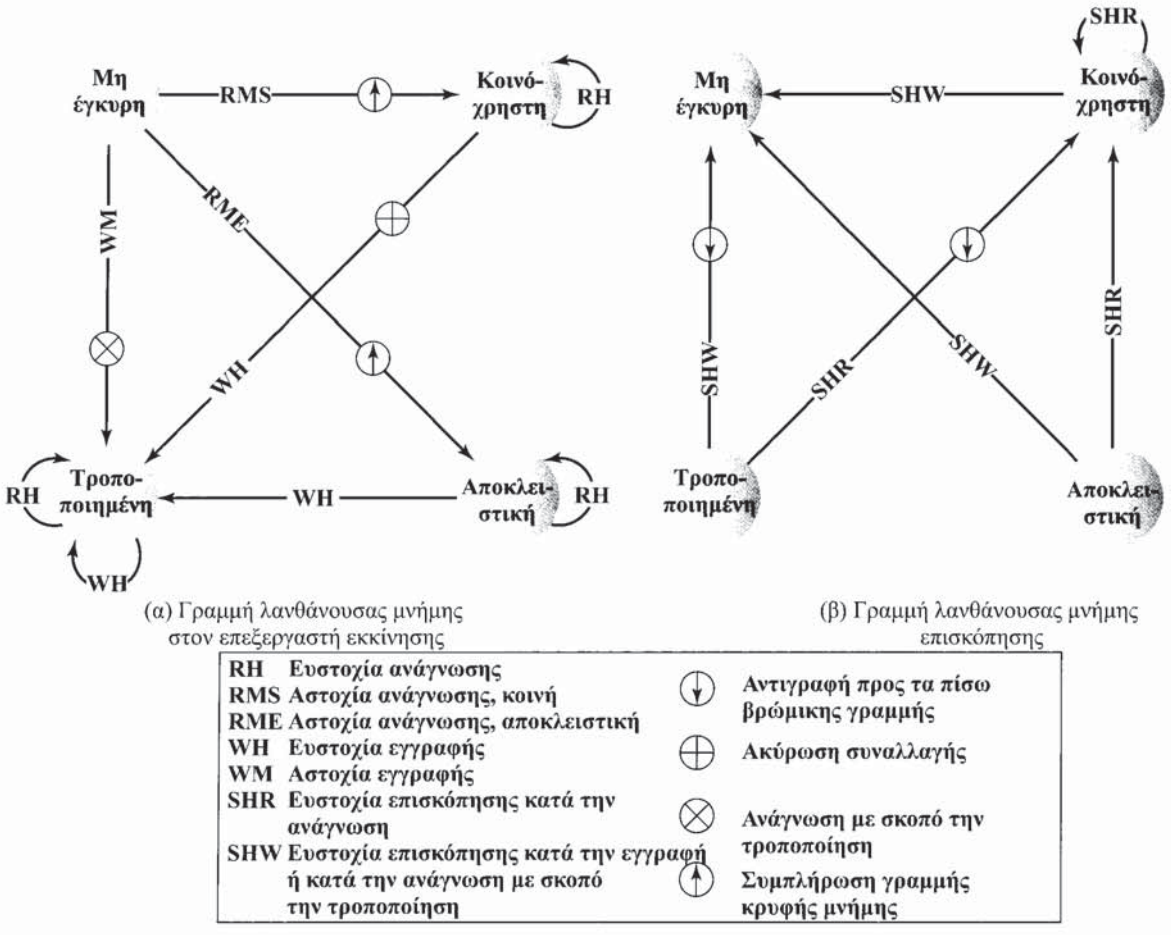
Για να παρέχει συνοχή των κρυφών μνημών σε ένα σύστημα συμμετρικών πολυεπεξεργαστών, η κρυφή μνήμη δεδομένων υποστηρίζει συχνά ένα πρωτόκολλο γνωστό με το όνομα MESI. Για το πρωτόκολλο αυτό, η κρυφή μνήμη δεδομένων συμπεριλαμβάνει δύο bits κατάστασης ανά ετικέτα, ώστε κάθε γραμμή να βρίσκεται σε μία από τις ακόλουθες καταστάσεις:

- **Τροποποιημένη:** Η γραμμή της κρυφής μνήμης έχει τροποποιηθεί (είναι διαφορετική σε σχέση με την κύρια μνήμη) και είναι διαθέσιμη μόνον σε αυτή την κρυφή μνήμη.
- **Αποκλειστική:** Η γραμμή της κρυφής μνήμης είναι η ίδια με εκείνη της κύριας μνήμης και δεν υπάρχει σε άλλη κρυφή μνήμη.
- **Κοινόχρηστη:** Η γραμμή της κρυφής μνήμης είναι ίδια με εκείνη της κύριας μνήμης και πιθανόν να υπάρχει σε άλλη κρυφή μνήμη.
- **Μη έγκυρη:** Η γραμμή της κρυφής μνήμης περιέχει μη έγκυρα δεδομένα.

Ο Πίνακας 17.1 συνοψίζει τη σημασία των τεσσάρων καταστάσεων. Το Σχήμα 17.7 απεικονίζει το διάγραμμα καταστάσεων του πρωτοκόλλου MESI. Θυμηθείτε ότι κάθε γραμμή της κρυφής μνήμης έχει τα δικά της bits κατάστασης και επομένως τη δική της υλοποίηση όσον αφορά το διάγραμμα καταστάσεων. Το Σχήμα 17.7α απεικονίζει τις μεταβάσεις οι οποίες λαμβάνουν χώρα εξαιτίας ενεργειών που ξεκινούν από τον επεξεργαστή ο οποίος συνδέεται με τη συγκεκριμένη κρυφή μνήμη. Το Σχήμα 17.7β απεικονίζει τις μεταβάσεις οι οποίες λαμβάνουν χώρα εξαιτίας γεγονότων τα οποία ανιχνεύονται μετά από αδιάκριτη εξέταση πάνω στο δίαυλο. Αυτή η αναπαράσταση των ενεργειών οι οποίες προκαλούνται από τον επεξεργαστή ή το δίαυλο με διαφορετικά διαγραμμάτων καταστάσεων, είναι χρήσιμη

Πίνακας 17.1: Καταστάσεις γραμμών της κρυφής μνήμης στο πρωτόκολλο MESI

	M Modified (Τροποποιημένη)	**E Exclusive (Αποκλειστική)**	**S Shared (Κοινόχρηστη)**	**I Invalid (Μη έγκυρη)**
Είναι έγκυρη αυτή η γραμμή της κρυφής μνήμης;	Ναι	Ναι	Ναι	Όχι
Το αντίγραφο της μνήμης είναι	μη ενημερωμένο	έγκυρο	έγκυρο	-
Υπάρχουν αντίγραφα σε άλλες κρυφές μνήμες;	Όχι	Όχι	Ίσως	Ίσως
Μία εγγραφή σε αυτή τη γραμμή	δεν πηγαίνει στο δίαυλο	δεν πηγαίνει στο δίαυλο	πηγαίνει στο δίαυλο και ενημερώνει την κρυφή μνήμη	πηγαίνει απ' ευθείας στο δίαυλο



(α) Γραμμή λανθάνουσας μνήμης στον επεξεργαστή εκκίνησης

(β) Γραμμή λανθάνουσας μνήμης επισκόπησης

Σχήμα 17.7: Διάγραμμα μετάβασης καταστάσεων του πρωτοκόλλου MESI

για να ξεκαθαρίσουμε τη λογική του πρωτοκόλλου MESI. Σε κάθε χρονική στιγμή, μία γραμμή της κρυφής μνήμης βρίσκεται σε μία κατάσταση. Αν το επόμενο γεγονός προέρχεται από το συνδεδεμένο επεξεργαστή, τότε η μετάβαση υπαγορεύεται από το Σχήμα 17.7α, ενώ αν προέρχεται από το δίαυλο,

τότε η μετάβαση υπαγορεύεται από το Σχήμα 17.7β. Στο σημείο αυτό, θα εξετάσουμε τις μεταβάσεις αυτές με μεγαλύτερη λεπτομέρεια.

ΑΣΤΟΧΙΑ ΑΝΑΓΝΩΣΗΣ Όταν συμβεί μία αστοχία ανάγνωσης στην τοπική κρυφή μνήμη, ο επεξεργαστής αρχικοποιεί μία λειτουργία ανάγνωσης από την κύρια μνήμη, με σκοπό να διαβάσει τη γραμμή της κύριας μνήμης η οποία περιέχει τη διεύθυνση η οποία δεν υπάρχει. Ο επεξεργαστής εισάγει ένα σήμα στο δίαυλο, με το οποίο ειδοποιεί όλες τις άλλες μονάδες επεξεργαστών/κρυφών μνημών να εξετάσουν αδιάκριτα τη μετάβαση. Υπάρχει ένα πλήθος από πιθανά αποτελέσματα:

- Αν μία άλλη κρυφή μνήμη διαθέτει ένα καθαρό (μη τροποποιημένο από τη στιγμή που διαβάστηκε από τη μνήμη) αντίγραφο της γραμμής σε αποκλειστική κατάσταση, επιστρέφει ένα σήμα με το οποίο δείχνει ότι μοιράζεται αυτή τη γραμμή. Στη συνέχεια, ο αποκρινόμενος επεξεργαστής αλλάζει την κατάσταση του αντιγράφου του από αποκλειστική σε κοινόχρηστη, και ο επεξεργαστής που ξεκίνησε τη διαδικασία διαβάζει τη γραμμή από την κύρια μνήμη, ενώ αλλάζει τη γραμμή της δικής του κρυφής μνήμης από μη έγκυρη σε κοινόχρηστη.

- Αν μία ή περισσότερες κρυφές μνήμες διαθέτουν ένα καθαρό αντίγραφο αυτής της γραμμής σε κατάσταση κοινόχρηστη, κάθε μία από αυτές στέλνει ένα σήμα ότι μοιράζεται τη γραμμή. Ο επεξεργαστής που ξεκίνησε τη διαδικασία διαβάζει τη γραμμή και αλλάζει τη δική του γραμμή από μη έγκυρη σε κοινόχρηστη.

- Αν μία άλλη κρυφή μνήμη διαθέτει τροποποιημένο αντίγραφο της γραμμής, τότε αυτή η κρυφή μνήμη εμποδίζει την ανάγνωση της μνήμης και παρέχει τη γραμμή στην κρυφή μνήμη η οποία τη ζήτησε, μέσα από το δίαυλο. Στη συνέχεια, η αποκριθείσα κρυφή μνήμη αλλάζει τη γραμμή της από τροποποιημένη σε κοινόχρηστη.[1] Η γραμμή η οποία αποστέλλεται στην κρυφή μνήμη η οποία έκανε την αίτηση, λαμβάνεται επίσης και υφίσταται επεξεργασία από τον ελεγκτή της κύριας μνήμης, ο οποίος αποθηκεύει το μπλοκ σε αυτή.

- Αν καμία άλλη κρυφή μνήμη δεν διαθέτει αντίγραφο της γραμμής (καθαρό ή τροποποιημένο), τότε δεν επιστρέφονται σήματα. Ο επεξεργαστής που ξεκίνησε τη διαδικασία διαβάζει τη γραμμή και αλλάζει τη γραμμή της δικής του κρυφής μνήμης από μη έγκυρη σε αποκλειστική.

ΕΥΣΤΟΧΙΑ ΑΝΑΓΝΩΣΗΣ Όταν υπάρξει ευστοχία ανάγνωσης σε μία γραμμή η οποία βρίσκεται στην τοπική κρυφή μνήμη, ο επεξεργαστής απλώς διαβάζει το ζητούμενο αντικείμενο. Δεν υπάρχει αλλαγή κατάστασης: Η κατάσταση παραμένει τροποποιημένη, κοινόχρηστη, ή αποκλειστική.

ΑΣΤΟΧΙΑ ΕΓΓΡΑΦΗΣ Όταν συμβεί αστοχία εγγραφής στην κρυφή μνήμη, ο επεξεργαστής ξεκινά μία λειτουργία ανάγνωσης από την κύρια μνήμη, με σκοπό να διαβάσει τη γραμμή της κύριας μνήμης η οποία περιέχει τη διεύθυνση η οποία δεν υπάρχει. Για το σκοπό αυτό, ο επεξεργαστής εκδίδει ένα σήμα στο δίαυλο, το οποίο έχει τη σημασία *ανάγνωση με σκοπό την τροποποίηση* (Read-with-intent-to-modify, RWITM). Όταν φορτωθεί αυτή η γραμμή, μαρκάρεται αμέσως ως τροποποιημένη. Σε σχέση με τις άλλες κρυφές μνήμες, δύο πιθανά σενάρια προηγούνται της φόρτωσης της γραμμής δεδομένων.

Πρώτον, μία άλλη κρυφή μνήμη είναι πιθανό να διαθέτει ένα τροποποιημένο αντίγραφο αυτής της γραμμής (κατάσταση=τροποποιημένη). Σε αυτή την περίπτωση, ο επεξεργαστής ο οποίος ειδοποιείται, στέλνει ένα σήμα στον επεξεργαστή που ξεκίνησε τη διαδικασία, με το οποίο δηλώνει ότι ένας

[1] Σε ορισμένες υλοποιήσεις, η κρυφή μνήμη που διαθέτει την τροποποιημένη γραμμή στέλνει ένα σήμα προς τον επεξεργαστή που ξεκίνησε τη διαδικασία, ώστε αυτός να ξαναπροσπαθήσει. Στο μεταξύ, ο επεξεργαστής με το τροποποιημένο αντίγραφο καταλαμβάνει το δίαυλο, γράφει την τροποποιημένη γραμμή προς τα πίσω στην κύρια μνήμη, και αλλάζει την κατάσταση της γραμμής του από τροποποιημένη σε κοινόχρηστη. Επομένως, ο επεξεργαστής που έχει κάνει την αίτηση προσπαθεί ξανά και βρίσκει ότι ένας ή περισσότεροι επεξεργαστές διαθέτουν ένα καθαρό αντίγραφο της γραμμής σε κατάσταση κοινόχρηστη, όπως περιγράφεται στο προηγούμενο σημείο.

άλλος επεξεργαστής διαθέτει ένα τροποποιημένο αντίγραφο της γραμμής. Ο αρχικός επεξεργαστής εγκαταλείπει το δίαυλο και περιμένει. Ο άλλος επεξεργαστής προσπελαύνει το δίαυλο και γράφει την τροποποιημένη γραμμή της κρυφής μνήμης πίσω στην κύρια μνήμη, αλλάζοντας την κατάσταση της γραμμής σε μη έγκυρη (γιατί ο αρχικός επεξεργαστής πρόκειται να την τροποποιήσει). Στη συνέχεια, ο αρχικός επεξεργαστής θα εκδόσει εκ νέου ένα σήμα RWITM προς το δίαυλο, θα διαβάσει τη γραμμή από την κύρια μνήμη, θα τροποποιήσει τη γραμμή στην κρυφή μνήμη, και θα μαρκάρει τη γραμμή σε κατάσταση τροποποιημένη.

Το δεύτερο σενάριο είναι ότι καμία άλλη κρυφή μνήμη δεν διαθέτει ένα τροποποιημένο αντίγραφο της γραμμής η οποία έχει ζητηθεί. Σε αυτή την περίπτωση, δεν θα επιστραφεί κανένα σήμα και ο επεξεργαστής που ξεκίνησε τη διαδικασία προχωρά στην ανάγνωση και έπειτα τροποποίηση της γραμμής. Στο μεταξύ, αν μία ή περισσότερες κρυφές μνήμες έχουν ένα καθαρό αντίγραφο της γραμμής σε κατάσταση κοινόχρηστη, κάθε κρυφή μνήμη ακυρώνει το δικό της αντίγραφο της γραμμής, ενώ αν μία κρυφή μνήμη έχει ένα καθαρό αντίγραφο της γραμμής σε αποκλειστική κατάσταση, ακυρώνει το αντίγραφό της.

ΕΥΣΤΟΧΙΑ ΕΓΓΡΑΦΗΣ Όταν συμβεί μία ευστοχία εγγραφής για μία γραμμή η οποία βρίσκεται στην τοπική κρυφή μνήμη, το αποτέλεσμα εξαρτάται από την τρέχουσα κατάσταση αυτής της γραμμής στην τοπική κρυφή μνήμη:

- **Κοινόχρηστη:** Πριν κάνει την ενημέρωση, ο επεξεργαστής θα πρέπει να αποκτήσει αποκλειστική ιδιοκτησία της γραμμής. Ο επεξεργαστής στέλνει ένα σήμα στο δίαυλο, κάνοντας γνωστό το σκοπό του. Κάθε επεξεργαστής ο οποίος διαθέτει ένα κοινόχρηστο αντίγραφο της γραμμής μέσα στη δική του κρυφή μνήμη, αλλάζει τη γραμμή από κοινόχρηστη σε μη έγκυρη. Στη συνέχεια, ο επεξεργαστής που ξεκίνησε τη διαδικασία εκτελεί την ενημέρωση και αλλάζει την τιμή κατάστασης της γραμμής του από κοινόχρηστη σε τροποποιημένη.

- **Αποκλειστική:** Ο επεξεργαστής έχει ήδη αποκτήσει αποκλειστική χρήση της γραμμής και απλώς εκτελεί την ενημέρωση και αλλάζει την κατάσταση από αποκλειστική σε τροποποιημένη.

- **Τροποποιημένη:** Ο επεξεργαστής έχει έχει ήδη αποκτήσει αποκλειστική χρήση της γραμμής και έχει μαρκάρει τη γραμμή ως τροποποιημένη, επομένως, εκτελεί απλώς την ενημέρωση.

ΣΥΜΦΩΝΙΑ ΚΡΥΦΩΝ ΜΝΗΜΩΝ ΕΠΙΠΕΔΟΥ 1 ΚΑΙ 2 Ως τώρα, έχουμε περιγράψει τα πρωτόκολλα συνοχής της κρυφής μνήμης με όρους μίας δραστηριότητας συνεργασίας ανάμεσα στις κρυφές μνήμες οι οποίες συνδέονται στον ίδιο δίαυλο ή σε κάποια άλλη δομή διασύνδεσης του συστήματος συμμετρικών πολυεπεξεργαστών. Τυπικά, αυτές οι κρυφές μνήμες είναι Επιπέδου 2, ενώ κάθε επεξεργαστής διαθέτει και μία κρυφή μνήμη Επιπέδου 1, η οποία δεν συνδέεται άμεσα με το δίαυλο, επομένως δεν είναι δυνατόν να εμπλακεί σε ένα αδιάκριτο πρωτόκολλο. Επομένως, είναι αναγκαία μία τεχνική διατήρισης της ακεραιότητας των δεδομένων και στα δύο επίπεδα της κρυφής μνήμης αλλά και μεταξύ όλων των κρυφών μνημών της διάταξης.

Η στρατηγική είναι να επεκταθεί το πρωτόκολλο MESI (ή κάθε άλλο πρωτόκολλο συνοχής), ώστε να υλοποιείται και σε κρυφές μνήμες Επιπέδου 1. Επομένως, κάθε γραμμή της κρυφής μνήμης Επιπέδου 1 συμπεριλαμβάνει bits, ώστε να δείχνει την κατάσταση. Ουσιαστικά, ο στόχος είναι ο εξής: για κάθε γραμμή η οποία είναι παρούσα τόσο στην κρυφή μνήμη Επιπέδου 2, όσο και στην αντίστοιχή της Επιπέδου 1, η κατάσταση γραμμής της μνήμης Επιπέδου 1 θα πρέπει να ανιχνεύει την κατάσταση της μνήμης Επιπέδου 2. Ένας απλός τρόπος για να γίνει αυτό, είναι να υιοθετηθεί από την κρυφή μνήμη Επιπέδου 1, μία πολιτική δια μέσου εγγραφής. Σε αυτή την περίπτωση, η δια μέσου εγγραφή γίνεται προς την κρυφή μνήμη Επιπέδου 2 και όχι προς τη μνήμη. Η πολιτική δια μέσου εγγραφής της μνήμης Επιπέδου 1 επιβάλλει τη μεταφορά κάθε αλλαγής που λαμβάνει χώρα σε μία δική της γραμμή, προς την κρυφή μνήμη Επιπέδου 2, κάτι που καθιστά αυτή την αλλαγή ορατή σε κάθε άλλη κρυφή μνήμη Επιπέδου 2. Η χρήση της δια μέσου εγγραφής της κρυφής μνήμης Επιπέδου 1 απαιτεί

τα περιεχόμενά της να είναι υποσύνολο της κρυφής μνήμης Επιπέδου 2. Αυτό το γεγονός σημαίνει ότι η συσχέτιση της κρυφής μνήμης Επιπέδου 2 θα πρέπει να είναι μεγαλύτερη ή ίση από εκείνη της κρυφής μνήμης Επιπέδου 1. Η πολιτική της δια μέσου εγγραφής της κρυφής μνήμης Επιπέδου 1 χρησιμοποιείται από το σύστημα S/390 της IBM.

Αν η κρυφή μνήμη Επιπέδου 1 διαθέτει μία πολιτική εγγραφής προς τα πίσω, η σχέση ανάμεσα στις δύο κρυφές μνήμες είναι πιο πολύπλοκη. Υπάρχουν διάφορες προσεγγίσεις για τη διατήρηση της συνοχής. Για παράδειγμα, η προσέγγιση την οποία χρησιμοποιεί το σύστημα Pentium II, περιγράφεται λεπτομερώς στην αναφορά [SHAN05].

17.4 ΠΟΛΥΝΗΜΑΤΩΣΗ ΚΑΙ CHIP ΠΟΛΥΕΠΕΞΕΡΓΑΣΤΩΝ

Το πιο σημαντικό μέτρο απόδοσης ενός επεξεργαστή είναι ο ρυθμός με τον οποίο εκτελεί τις εντολές. Ο ρυθμός αυτός εκφράζεται ως ακολούθως:

$$\text{Ρυθμός MIPS} = f \times IPC$$

όπου f είναι η συχνότητα του ρολογιού του επεξεργαστή και IPC (Instructions Per Cycle) είναι το μέσο πλήθος των εντολών οι οποίοι εκτελούνται ανά κύκλο. Επομένως, οι σχεδιαστές επιδιώκουν την αύξηση της απόδοσης κινούμενοι σε δύο άξονες: αύξηση της συχνότητας του ρολογιού και αύξηση του πλήθους των εκτελούμενων εντολών, ή πιο σωστά, του πλήθους των εντολών οι οποίες ολοκληρώνονται μέσα σε έναν κύκλο του επεξεργαστή. Όπως είδαμε στα προηγούμενα κεφάλαια, οι σχεδιαστές έχουν αυξήσει την τιμή IPC χρησιμοποιώντας τη μέθοδο της διασωλήνωσης και έπειτα, χρησιμοποιώντας παράλληλες διασωληνώσεις σε μία υπερβαθμωτή αρχιτεκτονική. Με τη διασωλήνωση και την πολλαπλή διασωλήνωση, το βασικό ζήτημα είναι η μεγιστοποίηση της χρησιμοποίησης κάθε σταδίου της διασωλήνωσης. Για να βελτιωθεί η ρυθμαπόδοση, οι σχεδιαστές δημιούργησαν ακόμη πιο πολύπλοκους μηχανισμούς, όπως είναι η εκτέλεση ορισμένων εντολών με διαφορετική σειρά από εκείνη με την οποία εμφανίζονται μέσα στη ροή του προγράμματος και η εκκίνηση της εκτέλεσης εντολών οι οποίες πιθανόν δεν θα χρειαστούν ποτέ. Ωστόσο, όπως περιγράφηκε στην Ενότητα 2.2, αυτή η προσέγγιση πιθανόν να φτάσει στα όριά της λόγω της πολυπλοκότητας και των ζητημάτων κατανάλωσης ισχύος.

Μία εναλλακτική προσέγγιση, η οποία επιτρέπει υψηλό βαθμό παραλληλοποίησης σε επίπεδο εντολής χωρίς να αυξάνεται η πολυπλοκότητα των κυκλωμάτων ή η κατανάλωση ισχύος, ονομάζεται πολυνημάτωση. Ουσιαστικά, η ροή των εντολών διαιρείται σε διάφορες μικρότερες ροές, γνωστές ως νήματα, με τέτοιο τρόπο, ώστε αυτά τα νήματα να εκτελούνται παράλληλα.

Η ποικιλία συγκεκριμένων μορφών σχεδίασης της πολυνημάτωσης, οι οποίες υλοποιούνται τόσο σε εμπορικά όσο και σε πειραματικά συστήματα, είναι τεράστια. Σε αυτή την ενότητα, θα δώσουμε μία σύντομη μελέτη των βασικών εννοιών.

Έμμεση και Άμεση Πολυνημάτωση

Η έννοια του νήματος το οποίο χρησιμοποιείται σε επεξεργαστές πολυνημάτωσης πιθανόν να είναι ή να μην είναι η ίδια με την έννοια των νημάτων λογισμικού την οποία συναντούμε σε ένα λειτουργικό σύστημα πολυπρογραμματισμού. Είναι χρήσιμο να δώσουμε σύντομα μερικούς ορισμούς:

- **Διεργασία:** Ένα στιγμιότυπο προγράμματος το οποίο εκτελείται σε έναν υπολογιστή. Η διεργασία ενσωματώνει δύο βασικά χαρακτηριστικά:

  - **Ιδιοκτησία πόρων:** Μία διεργασία συμπεριλαμβάνει ένα χώρο ιδεατών διευθύνσεων για να διατηρεί την εικόνα της διεργασίας. Η εικόνα διεργασίας είναι η συλλογή η οποία περιέχει το πρόγραμμα, τα δεδομένα, τη στοίβα, και τα χαρακτηριστικά τα οποία ορίζουν τη διεργασία. Από χρονική στιγμή σε χρονική στιγμή, κατανέμεται στη διεργασία ο έλεγχος

ή η ιδιοκτησία των πόρων, όπως είναι η κύρια μνήμη, τα κανάλια Ε/Ε, οι μονάδες Ε/Ε, και τα αρχεία.

- **Χρονοδρομολόγηση/εκτέλεση:** Η εκτέλεση μίας διεργασίας ακολουθεί ένα μονοπάτι εκτέλεσης (ίχνος) μέσα από ένα ή περισσότερα προγράμματα. Αυτή η εκτέλεση είναι πιθανόν διαστρωματωμένη με την εκτέλεση άλλων διεργασιών. Επομένως, η διεργασία έχει μία κατάσταση εκτέλεσης (Εκτελούμενη, Έτοιμη, κ.ο.κ), και μία αναφορά προτεραιότητας, και αποτελεί την οντότητα η οποία έχει χρονοδρομολογηθεί και έχει λάβει προτεραιότητα από το λειτουργικό σύστημα.

- **Μεταγωγή διεργασίας:** Μία λειτουργία, η οποία εναλλάσσει τον επεξεργαστή μεταξύ διαφορετικών διεργασιών, αποθηκεύοντας όλα τα δεδομένα ελέγχου διεργασίας, τους καταχωρητές, αλλά και τις υπόλοιπες πληροφορίες της πρώτης και αντικαθιστώντας τις με τις αντίστοιχες πληροφορίες της δεύτερης. [2]

- **Νήμα:** Μία εκτελούμενη μονάδα εργασίας μέσα σε μία διεργασία. Ένα νήμα συμπεριλαμβάνει ένα περιβάλλον επεξεργαστή (μειρητής προγράμματος και δείκτης στοίβας) και τη δική του περιοχή δεδομένων για τη στοίβα (ώστε να επιτρέπεται η διακλάδωση σε ρουτίνες). Ένα νήμα εκτελείται ακολουθιακά και υπάρχει δυνατότητα διακοπής του, ώστε ο επεξεργαστής να στραφεί σε ένα άλλο νήμα.

- **Μεταγωγή νήματος:** Η ενέργεια της μεταφοράς του ελέγχου του επεξεργαστή από ένα νήμα σε ένα άλλο, εντός της ίδιας διεργασίας. Τυπικά, αυτή η μορφή μεταγωγής είναι πολύ λιγότερο χρονοβόρα σε σχέση με τη μεταγωγή διεργασίας.

Επομένως, ένα νήμα επικεντρώνεται στη χρονοδρομολόγηση και στην εκτέλεση, ενώ μία διεργασία επικεντρώνεται τόσο στη χρονοδρομολόγηση/εκτέλεση, αλλά και στην ιδιοκτησία των πόρων. Τα πολλαπλά νήματα εντός μίας διεργασίας μοιράζονται τους ίδιους πόρους. Αυτός είναι ο λόγος για τον οποίο η μεταγωγή νήματος είναι πολύ λιγότερο χρονοβόρα σε σχέση με τη μεταγωγή διεργασίας. Τα κλασικά λειτουργικά συστήματα, όπως είναι οι αρχικές εκδόσεις του UNIX, δεν υποστήριζαν νήματα. Τα πιο σύγχρονα λειτουργικά συστήματα, όπως είναι το Linux και άλλες εκδόσεις του UNIX, αλλά και τα Windows, υποστηρίζουν τα νήματα. Γίνεται μία διάκριση ανάμεσα στα νήματα επιπέδου χρήστη, τα οποία είναι ορατά στο πρόγραμμα εφαρμογής, και στα νήματα επιπέδου πυρήνα, τα οποία είναι ορατά μόνον στο λειτουργικό σύστημα. Αμφότερα, αναφέρονται ως άμεσα νήματα και ορίζονται σε επίπεδο λογισμικού.

Ως τώρα, όλοι οι εμπορικοί επεξεργαστές και οι περισσότεροι πειραματικοί επεξεργαστές, χρησιμοποιούσαν την άμεση πολυνημάτωση. Αυτά τα συστήματα εκτελούν ταυτόχρονα εντολές από διαφορετικά άμεσα νήματα, είτε μέσω φύλλωσης εντολών διαφορετικών νημάτων σε κοινόχρηστες διασωληνώσεις, είτε με παράλληλη εκτέλεση σε παράλληλες διασωληνώσεις. Η έμμεση πολυνημάτωση αναφέρεται στην ταυτόχρονη εκτέλεση πολλαπλών νημάτων τα οποία εξάγονται από ένα ακολουθιακό πρόγραμμα. Αυτά τα έμμεσα νήματα ορίζονται είτε στατικά από το μεταγλωττιστή, είτε δυναμικά από το υλικό. Στο υπόλοιπο αυτής της ενότητας, θα εξετάσουμε την άμεση πολυνημάτωση.

[2] Ο όρος *μεταγωγή περιβαλλοντος* είναι αρκετά συχνός στην αρθρογραφία και στα βιβλία. Δυστυχώς, παρά το γεγονός ότι η αρθρογραφία χρησιμοποιεί αυτό τον όρο για να ονομάσει αυτό που εδώ ονομάζεται μεταγωγή διεργασίας, άλλες πηγές τον χρησιμοποιούν για να ονομάσουν το μεταγωγή νήματος. Για να αποφύγουμε τη σύγχυση, δεν χρησιμοποιούμε τον όρο αυτό σε αυτό το βιβλίο.

Προσεγγίσεις Άμεσης Πολυνημάτωσης

Κατ' ελάχιστο, ένας επεξεργαστής πολλαπλών νημάτων θα πρέπει να διαθέτει έναν ξεχωριστό μετρητή προγράμματος για κάθε νήμα το οποίο πρόκειται να εκτελεστεί ταυτόχρονα. Οι μορφές σχεδίασης διαφέρουν ως προς την ποσότητα και τη μορφή του επιπλέον υλικού το οποίο χρησιμοποιείται για να υποστηριχθεί η ταυτόχρονη εκτέλεση των νημάτων. Γενικά η ανάκληση των εντολών λαμβάνει χώρα στη βάση ενός νήματος. Ο επεξεργαστής διαχειρίζεται κάθε νήμα ξεχωριστά και πιθανόν να χρησιμοποιεί ένα πλήθος από τεχνικές βελτιστοποίησης της εκτέλεσης απλών νημάτων, συμπεριλαμβανομένης της πρόβλεψης διακλάδωσης, της μετονομασίας καταχωρητών, και των υπερβαθμωτών τεχνικών. Αυτό το οποίο επιτυγχάνεται, είναι ένας παραλληλισμός σε επίπεδο νήματος, ο οποίος μπορεί να δώσει σημαντικά καλύτερη απόδοση, αν συνδυαστεί με τον παραλληλισμό σε επίπεδο μηχανής.

Μιλώντας γενικά, υπάρχουν τέσσερις βασικές προσεγγίσεις όσον αφορά την πολυνημάτωση:

- **Φυλλωμένη πολυνημάτωση:** Αναφέρεται και ως **εκλεπτυσμένη πολυνημάτωση**. Ο επεξεργαστής διαχειρίζεται δύο ή περισσότερα νήματα ταυτόχρονα, μεταφέροντας τον έλεγχο από το ένα στο άλλο σε κάθε κύκλο του ρολογιού. Αν ένα νήμα παρεμποδιστεί λόγω εξαρτήσεων στα δεδομένα ή λόγω λανθάνοντος χρόνου της μνήμης, τότε ο επεξεργαστής υπερπηδά αυτό το νήμα και προχωρά στην εκτέλεση ενός άλλου, το οποίο είναι έτοιμο.

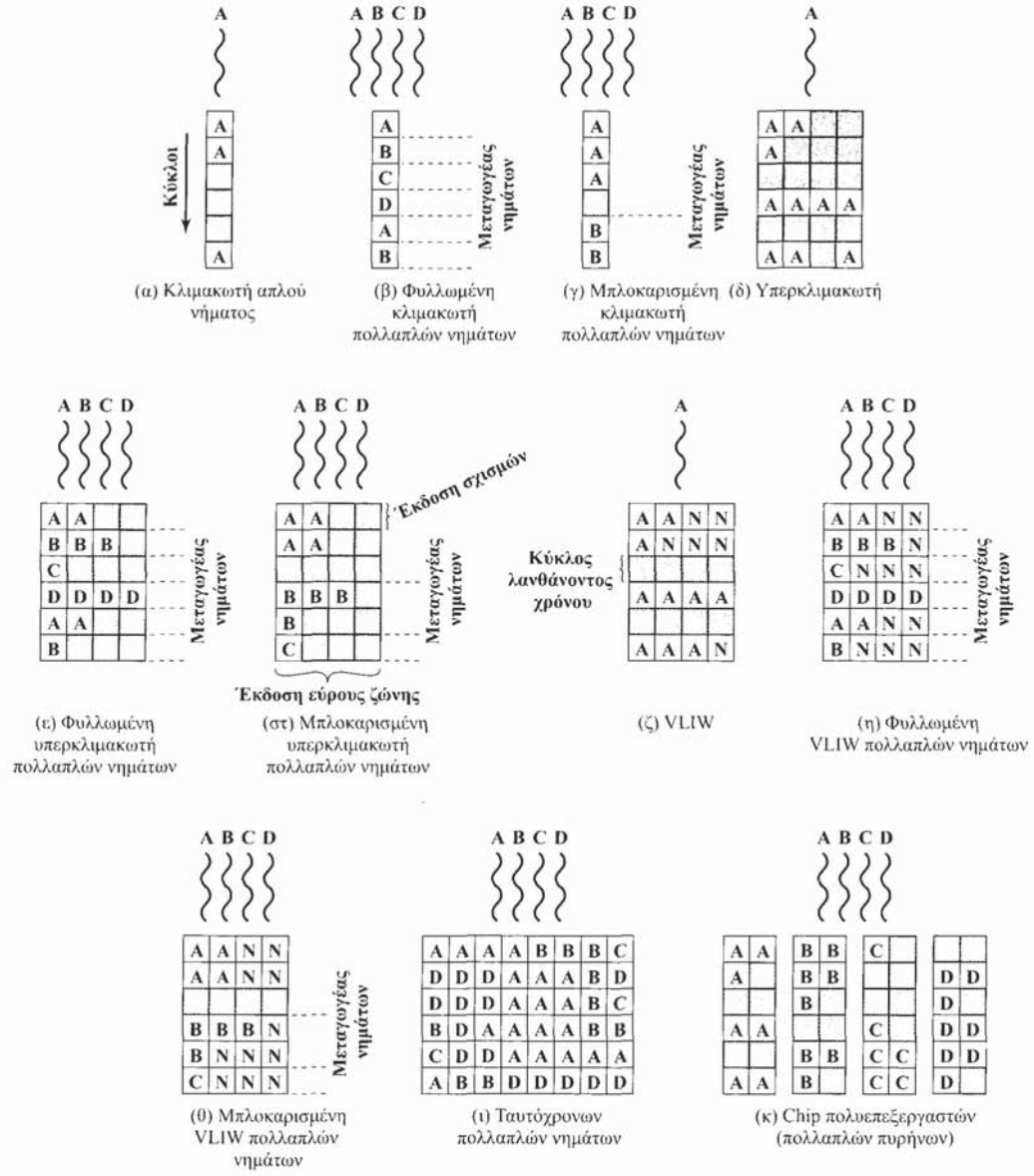
- **Παρεμποδισμένη πολυνημάτωση:** Αναφέρεται και ως **μη εκλεπτυσμένη πολυνημάτωση**. Οι εντολές ενός νήματος εκτελούνται διαδοχικά έως ότου λάβει χώρα ένα γεγονός το οποίος είναι δυνατόν να προκαλέσει καθυστέρηση, όπως είναι η αστοχία της κρυφής μνήμης. Αυτό το γεγονός συμπεριλαμβάνει και τη μετάβαση σε ένα άλλο νήμα. Η προσέγγιση αυτή είναι αποτελεσματική σε έναν επεξεργαστή ο οποίος εκδίδει εντολές σε σειρά και θα διακόψει τη διασωλήνωση αν υπάρξει ένα γεγονός καθυστέρησης όπως είναι η αστοχία της κρυφής μνήμης.

- **Ταυτόχρονη πολυνημάτωση:** Οι εντολές εκδίδονται ταυτόχρονα από πολλαπλά νήματα προς τις μονάδες εκτέλεσης ενός υπερβαθμωτού επεξεργαστή. Η προσέγγιση αυτή συνδυάζει τη δυνατότητα ευρείας έκδοσης εντολών των υπερβαθμωτών επεξεργαστών με τη χρήση πολλαπλών νημάτων.

- **Πολυεπεξεργασία σε επίπεδο chip:** Σε αυτή την περίπτωση, ολόκληρος ο επεξεργαασιής αντιγράφεται σε ένα chip και κάθε επεξεργαστής διαχειρίζεται ξεχωριστά νήματα. Το πλεονέκτημα αυτής της προσέγγισης είναι ότι η διαθέσιμη περιοχή λογικής ενός chip χρησιμοποιείται αποτελεσματικά, χωρίς να εξαρτάται από την ολοένα αυξανόμενη πολυπλοκότητα σχεδίασης της διασωλήνωσης. Αυτό αναφέρεται και ως πολλαπλοί πυρήνες και εξετάζεται ξεχωριστά στο Κεφάλαιο 18.

Στις δύο πρώτες προσεγγίσεις, οι εντολές που εκδίδονται από διαφορετικά νήματα δεν εκτελούνται ταυτόχρονα. Αντίθετα, ο επεξεργαστής έχει τη δυνατότητα να μεταβεί με μεγάλη ταχύτητα από ένα νήμα σε ένα άλλο, χρησιμοποιώντας ένα διαφορετικό σύνολο από καταχωρητές και άλλες πληροφορίες. Αυτό έχει ως αποτέλεσμα την καλύτερη χρήση των πόρων του επεξεργαστή και οδηγεί στην αποφυγή μεγάλων επιβαρύνσεων οι οποίες προκύπτουν από αστοχίες της κρυφής μνήμης και άλλα γεγονότα τα οποία σχετίζονται με λανθάνοντα χρόνο. Η προσέγγισης της ταυτόχρονης πολυνημάτωσης εμπεριέχει πραγματικά ταυτόχρονες εκτελέσεις εντολών οι οποίες εκδίδονται από διαφορετικά νήματα, με χρήση αντιγράφων των πόρων του επεξεργαστή. Η πολυεπεξεργασία σε επίπεδο chip επίσης επιτρέπει την ταυτόχρονη εκτέλεση εντολών, οι οποίες εκδίδονται από διαφορετικά νήματα.

Το Σχήμα 17.8, το οποίο βασίζεται σε ένα αντίστοιχο σχήμα της αναφοράς [UNGE02], απεικονίζει ορισμένες από τις πιθανές αρχιτεκτονικές διασωλήνωσης οι οποίες συμπεριλαμβάνουν την πολυνημάτωση και τις αντιπαραβάλει με εκείνες οι οποίες δεν χρησιμοποιούν πολυνημάτωση. Κάθε οριζόντια γραμμή αναπαριστά την πιθανή ή πιθανές σχισμές έκδοσης για έναν απλό κύκλο εκτέλεσης. Με άλλα



Σχήμα 17.8: Προσεγγίσεις της εκτέλεσης πολλαπλών νημάτων

λόγια, το πλάτος κάθε γραμμής αντιστοιχεί στο μέγιστο πλήθος των εντολών οι οποίες είναι δυνατόν να εκδοθούν εντός ενός κύκλου του ρολογιού.[3] Η κατακόρυφη διάσταση αναπαριστά τη χρονική ακολουθία των κύκλων ρολογιού. Ένα κενό (σκιασμένο) τετράγωνο αναπαριστά μία αχρησιμοποίητη σχισμή εκτέλεσης μίας διασωλήνωσης. Με Ν δηλώνεται μία εντολή No-op.

Τα τρία πρώτα μέρη του Σχήματος 17.8 απεικονίζουν διαφορετικές προσεγγίσεις, όταν χρησιμοποιείται ένας βαθμωτός (π.χ. απλής έκδοσης εντολών) επεξεργαστής:

[3] Οι σχισμές έκδοσης είναι η θέση από την οποία οι εντολές είναι δυνατόν να εκδοθούν εντός ενός κύκλου του ρολογιού. Από το Κεφάλαιο 14, θυμηθείτε ότι η έκδοση μίας εντολής είναι η διαδικασία αρχικοποίησης της εκτέλεσής της από τις λειτουργικές μονάδες του επεξεργαστή. Αυτό συμβαίνει όταν μία εντολή μεταφέρεται από το στάδιο αποκωδικοποίησης της διασωλήνωσης, στο πρώτο στάδιο εκτέλεσης.

- **Βαθμωτός, απλής νημάτωσης:** Πρόκειται για την απλή διασωλήνωση την οποία συναντάμε στις κλασικές μηχανές RISC και CISC, χωρίς πολυνημάτωση.

- **Βαθμωτός, φυλλωμένης πολυνημάτωσης:** Πρόκειται για την απλούστερη στην υλοποίηση προσέγγιση όσον αφορά την πολυνημάτωση. Μεταβαίνοντας από το ένα νήμα στο άλλο σε κάθε κύκλο του ρολογιού, τα στάδια της διασωλήνωσης διατηρούνται πλήρως απασχολημένα, ή σχεδόν πλήρως απασχολημένα. Το υλικό θα πρέπει να έχει τη δυνατότητα να μεταβαίνει από το ένα νήμα στο άλλο, μεταξύ των κύκλων του ρολογιού.

- **Βαθμωτός, παρεμποδισμένης πολυνημάτωσης:** Σε αυτή την περίπτωση, εκτελείται ένα απλό νήμα έως ότου να εμφανιστεί κάποιος λανθάνων χρόνος, ο οποίος θα σταματήσει τη διασωλήνωση. Εκείνη τη χρονική στιγμή ο επεξεργαστής θα μεταβεί σε ένα άλλο νήμα.

Το Σχήμα 17.8γ απεικονίζει μία κατάσταση στην οποία ο χρόνος που απαιτείται για μία μετάβαση από νήμα σε νήμα ισούται με ένα κύκλο, ενώ στο Σχήμα 17.8β απεικονίζεται ότι η μετάβαση γίνεται σε μηδέν κύκλους. Στην περίπτωση της φυλλωμένης πολυνημάτωσης, θεωρείται ότι δεν υπάρχουν εξαρτήσεις ελέγχου ή δεδομένων ανάμεσα στα νήματα, κάτι το οποίο απλοποιεί τη σχεδίαση της διασωλήνωσης και επομένως, θα πρέπει να επιτρέπει τη μετάβαση μεταξύ νημάτων χωρίς καθυστέρηση. Ωστόσο, αναλόγως με τη σχεδίαση και υλοποίηση, η παρεμποδισμένη πολυνημάτωση είναι πιθανό να απαιτεί έναν κύκλο για τις ανάγκες της μετάβασης, όπως απεικονίζεται στο Σχήμα 17.8. Αυτό αληθεύει αν η εντολή που προσκομίζεται ενεργοποιεί τη μετάβαση ανάμεσα σε δύο νήματα και πρέπει να απορριφθεί από τη διασωλήνωση [UNGE03].

Παρά το γεγονός ότι η φυλλωμένη πολυνημάτωση φαίνεται να δίνει μεγαλύτερο βαθμό χρήσης του επεξεργαστή σε σύγκριση με την παρεμποδισμένη, αυτό επιτυγχάνεται θυσιάζοντας την απόδοση του απλού νήματος. Τα πολλαπλά νήματα ανταγωνίζονται για τους πόρους της κρυφής μνήμης, κάτι το οποίο αυξάνει την πιθανότητα αστοχίας για ένα συγκεκριμένο νήμα.

Περισσότερες δυνατότητες για παράλληλη εκτέλεση διατίθενται αν ο επεξεργαστής έχει τη δυνατότητα να εκδίδει πολλαπλές εντολές ανά κύκλο. Τα Σχήματα 17.8δ ως 18.8θ απεικονίζουν ένα πλήθος από ποικιλίες επεξεργαστών οι οποίοι διαθέτουν το υλικό για να εκδίδουν τέσσερις εντολές ανά κύκλο. Σε όλες αυτές τις περιπτώσεις, σε έναν κύκλο εκδίδονται μόνον οι εντολές από ένα απλό νήμα. Οι παρακάτω εναλλακτικές λύσεις απεικονίζονται στο σχήμα:

- **Υπερβαθμωτή:** Πρόκειται για τη βασική υπερβαθμωτή προσέγγιση, η οποία δεν χρησιμοποιεί πολυνημάτωση. Έως πρόσφατα, ήταν η πιο ισχυρή προσέγγιση όσον αφορά την παροχή δυνατοτήτων παραλληλισμού μέσα σε έναν επεξεργαστή. Παρατηρείστε ότι, κατά τη διάρκεια ορισμένων κύκλων, δεν χρησιμοποιούνται όλες οι διαθέσιμες σχισμές. Κατά τη διάρκεια αυτών των κύκλων, εκδίδονται λιγότερες εντολές από το μέγιστο δυνατό πλήθος. Αυτό το γεγονός αναφέρεται ως *οριζόντια απώλεια*. Κατά τη διάρκεια άλλων κύκλων εντολών, δεν χρησιμοποιούνται καθόλου σχισμές. Πρόκειται για κύκλους, κατά τη διάρκεια των οποίων δεν είναι δυνατή η έκδοση εντολών. Αυτό το γεγονός αναφέρεται ως *κατακόρυφη απώλεια*.

- **Υπερβαθμωτή φυλλωμένη πολυνημάτωση:** Κατά τη διάρκεια κάθε κύκλου, εκδίδονται όσο το δυνατόν περισσότερες εντολές από ένα απλό νήμα. Με αυτή την τεχνική, εξαλείφονται οι πιθανές καθυστερήσεις οι οποίες οφείλονται σε μεταβάσεις μεταξύ νημάτων, όπως συζητήθηκε προηγουμένως. Ωστόσο, το πλήθος των εντολών οι οποίες εκδίδονται σε κάθε κύκλο, εξακολουθεί να είναι περιορισμένο λόγω των εξαρτήσεων οι οποίες υπάρχουν σε κάθε νήμα.

- **Υπερβαθμωτή παρεμποδισμένη πολυνημάτωση:** Ξανά, οι εντολές ενός νήματος είναι πιθανό να εκδοθούν κατά τη διάρκεια οποιουδήποτε κύκλου και χρησιμοποιείται η παρεμποδισμένη πολυνημάτωση.

- **Λέξη εντολής πολύ μεγάλου μήκους (Very Long Instruction Word-VLIW):** Η αρχιτεκτονική VLIW όπως εκείνη της μηχανής IA-64, τοποθετεί πολλαπλές εντολές σε μία λέξη. Τυπικά, μία VLIW δημιουργείται από το μεταγλωττιστή, ο οποίος τοποθετεί στην ίδια λέξη τις λειτουργίες οι οποίες είναι πιθανό να εκτελεστούν παράλληλα. Σε μία απλή μηχανή VLIW (Σχήμα 17.8ζ), αν δεν υπάρχει δυνατότητα να συμπληρωθεί πλήρως η λέξη με εντολές οι οποίες θα εκδοθούν παράλληλα, χρησιμοποιούνται εντολές NO-OP.

- **Φυλλωμένη πολυνημάτωση VLIW:** Αυτή η προσέγγιση θα πρέπει να παρέχει παρόμοια αποδοτικότητα με εκείνη που παρέχει η φυλλωμένη πολυνημάτωση σε μία υπερβαθμωτή αρχιτεκτονική.

- **Παρεμποδισμένη πολυνημάτωση VLIW:** Αυτή η προσέγγιση θα πρέπει να παρέχει παρόμοια αποδοτικότητα με εκείνη που παρέχει η παρεμποδισμένη πολυνημάτωση σε μία υπερβαθμωτή αρχιτεκτονική.

Οι δύο τελευταίες προσεγγίσεις οι οποίες απεικονίζονται στο Σχήμα 17.8 επιτρέπουν την παράλληλη, ταυτόχρονη εκτέλεση πολλαπλών νημάτων:

- **Ταυτόχρονη πολυνημάτωση:** Το Σχήμα 17.8θ απεικονίζει ένα σύστημα το οποίο έχει τη δυνατότητα να εκδίδει 8 εντολές κάθε φορά. Αν ένα νήμα έχει μεγάλο βαθμό παραλληλισμού σε επίπεδο εντολής, πιθανόν να έχει τη δυνατότητα να γεμίζει τις οριζόντιες σχισμές σε μερικούς κύκλους. Σε άλλους κύκλους, είναι πιθανό να εκδοθούν οι εντολές από δύο ή περισσότερα νήματα. Αν είναι ενεργό ένα επαρκές πλήθος από νήματα, συνήθως υπάρχει η δυνατότητα έκδοσης του μέγιστου πλήθους εντολών σε κάθε κύκλο, κάτι που παρέχει μεγάλη αποδοτικότητα.

- **Chip Πολυεπεξεργαστών (πολλαπλών πυρήνων):** Το Σχήμα 17.8κ απεικονίζει ένα chip το οποίο περιέχει τέσσερις επεξεργαστές, καθένας εκ των οποίων είναι υπερβαθμωτός και εκδίδει δύο εντολές. Σε κάθε επεξεργαστή εκχωρείται ένα νήμα, από το οποίο έχει τη δυνατότητα έκδοσης έως και δύο εντολών ανά κύκλο. Οι επεξεργαστές πολλαπλών πυρήνων περιγράφονται στο Κεφάλαιο 18.

Συγκρίνοντας τα Σχήματα 17.8ι και 17.8κ, διαπιστώνουμε ότι ένα chip πολυεπεξεργαστών, το οποίο έχει την ίδια δυνατότητα έκδοσης εντολών με έναν επεξεργαστή ταυτόχρονης πολυνημάτωσης, δεν είναι δυνατόν να επιτύχει τον ίδιο βαθμό παραλληλισμού σε επίπεδο εντολής. Αυτό οφείλεται στο γεγονός ότι ο πολυεπεξεργαστής δεν έχει τη δυνατότητα να επικαλύψει τον λανθάνοντα χρόνο εκδίδοντας εντολές από άλλα νήματα. Από την άλλη, το chip πολυεπεξεργαστών θα πρέπει να έχει καλύτερη απόδοση από τον υπερβαθμωτό με την ίδια δυνατότητα έκδοσης εντολών, επειδή ο τελευταίος θα εμφανίζει μεγαλύτερες οριζόντιες απώλειες. Επιπλέον, είναι εφικτό να χρησιμοποιηθεί η πολυνημάτωση εντός καθενός από τους επεξεργαστές του πολυεπεξεργαστή και αυτό γίνεται ήδη σε ορισμένες σύγχρονες μηχανές.

Παραδείγματα Συστημάτων

PENTIUM 4 Τα πιο πρόσφατα μοντέλα του επεξεργαστή Pentium 4 χρησιμοποιούν μία τεχνική πολυνημάτωσης, η οποία στα εγχειρίδια της Intel αναφέρεται ως *υπερνημάτωση* [MARR02]. Ουσιαστικά, η προσέγγιση του Pentium 4 είναι να χρησιμοποιηθεί η ταυτόχρονη πολυνημάτωση με υποστήριξη για δύο νήματα. Επομένως, ο απλός επεξεργαστής πολλαπλών νημάτων, είναι λογικά δύο επεξεργαστές.

IBM POWER5 Το chip του επεξεργαστή Power5, το οποίο χρησιμοποιείται στα πιο σύγχρονα προϊόντα της οικογένειας PowerPC, συνδυάζει την πολυεπεξεργασία σε επίπεδο chip με την ταυτόχρονη πολυνημάτωση [KALL04]. Το chip διαθέτει δύο ξεχωριστούς επεξεργαστές, καθένας εκ των οποίων

είναι ένας επεξεργαστής πολλαπλών νημάτων με δυνατότητα ταυτόχρονης υποστήριξης δύο νημάτων χρησιμοποιώντας ταυτόχρονη πολυνημάτωση. Παρουσιάζει ενδιαφέρον το γεγονός ότι, οι σχεδιαστές προσομοίωσαν διάφορες εναλλακτικές λύσεις και διαπίστωσαν ότι η ύπαρξη δύο επεξεργαστών ταυτόχρονης πολυνημάτωσης οι οποίοι υποστηρίζουν δύο νήματα πάνω σε ένα chip, παρείχε μεγαλύτερη απόδοση σε σύγκριση με έναν επεξεργαστή ο οποίος υποστηρίζει 4 νήματα. Οι προσομοιώσεις έδειξαν ότι η επιπλέον πολυνημάτωση πέραν της υποστήριξης δύο νημάτων είναι πιθανό να μειώσει την απόδοση λόγω του γεγονότος ότι τα δεδομένα που χρησιμοποιεί ένα νήμα εκτοπίζουν από την κρυφή μνήμη εκείνα τα οποία χρησιμοποιεί ένα άλλο νήμα.

Το Σχήμα 17.9 απεικονίζει το διάγραμμα ροής των εντολών του επεξεργαστή Power5 της IBM. Μόνον λίγα από τα στοιχεία του επεξεργαστή είναι ανάγκη να αντιγραφούν και γενικά, διαφορετικά στοιχεία δεσμεύονται σε διαφορετικά νήματα. Χρησιμοποιούνται δύο μετρητές προγράμματος. Ο επεξεργαστής εναλλάσσει την ανάκληση εντολών, μέχρι 8 κάθε φορά, ανάμεσα σε δύο νήματα. Όλες οι εντολές αποθηκεύονται σε μία κοινή κρυφή μνήμη εντολών και μοιράζονται ένα μηχανισμό μετάφρασης εντολών, ο οποίος εκτελεί μία μερική αποκωδικοποίηση. Όταν συναντάται μία διακλάδωση υπό συνθήκη, ο μηχανισμός πρόβλεψης διακλάδωσης προβλέπει την κατεύθυνση της διακλάδωσης και, αν αυτό είναι εφικτό, υπολογίζει τη διεύθυνση προορισμού. Για να προβλεφθεί ο προορισμός της επιστροφής από μία υπορουτίνα, ο επεξεργαστής χρησιμοποιεί μία στοίβα επιστροφής για κάθε νήμα.

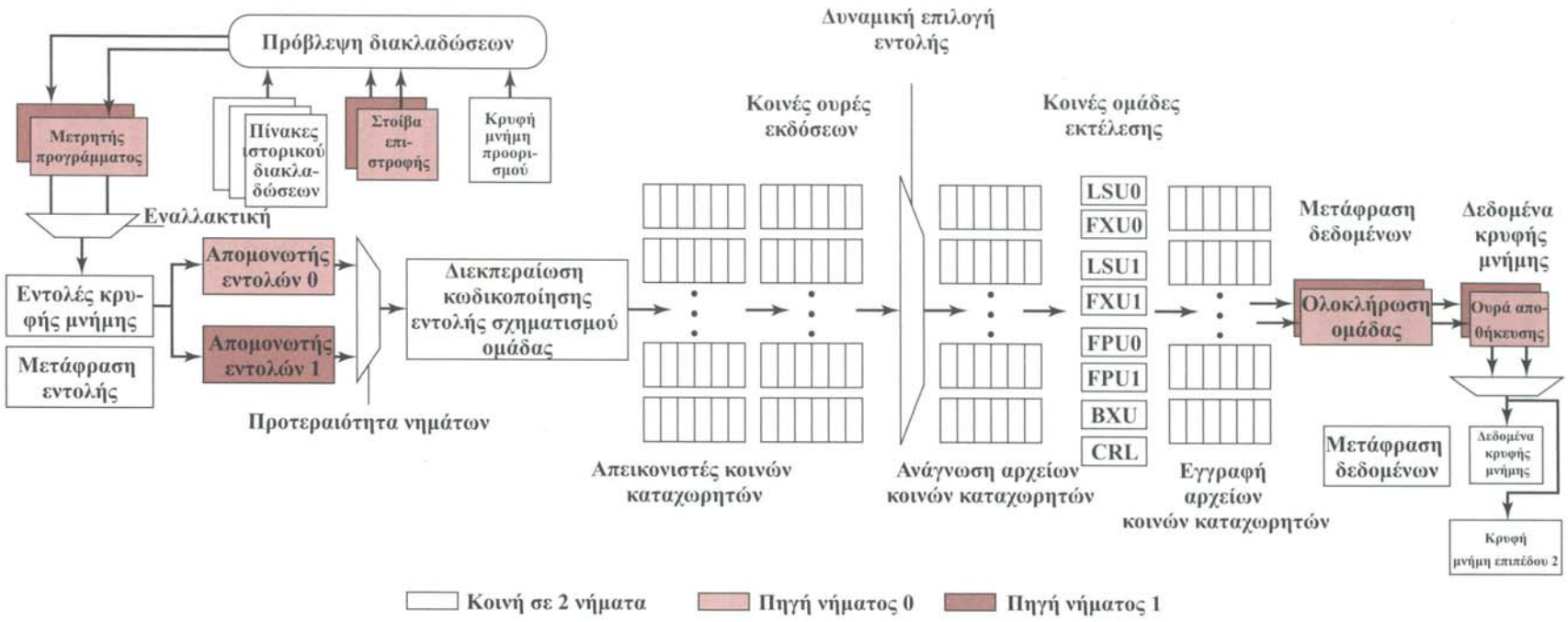
Στη συνέχεια, οι εντολές μεταφέρονται σε δύο ξεχωριστούς απομονωτές εντολών. Έπειτα, βάσει της προτεραιότητας των νημάτων, επιλέγεται και αποκωδικοποιείται παράλληλα μία ομάδα εντολών. Στη συνέχεια, οι εντολές περνούν από μία μετονομασία καταχωρητή, με τη σειρά εμφάνισής τους στο πρόγραμμα. Οι λογικοί καταχωρητές απεικονίζονται στους φυσικούς. Η μηχανή PowerPC διαθέτει 120 φυσικούς καταχωρητές κινητής υποδιαστολής. Έπειτα, οι εντολές μετακινούνται σε ουρές έκδοσης. Από αυτές τις ουρές, γίνεται η έκδοσή τους με χρήση συμμετρικής πολυνημάτωσης. Με άλλα λόγια, ο επεξεργαστής διαθέτει μία υπερβαθμωτή αρχιτεκτονική και έχει τη δυνατότητα να εκδίδει εντολές από ένα ή δύο νήματα παράλληλα. Στο τέλος της διασωλήνωσης, είναι αναγκαίο τα νήματα να διαθέτουν ξεχωριστούς πόρους για την εξυπηρέτηση των εντολών.

17.5 ΣΥΣΤΑΔΕΣ

Μία πολύ σημαντική και σχετικά πρόσφατη εξέλιξη της σχεδίασης των υπολογιστικών συστημάτων, είναι η συσταδοποίηση. Η συσταδοποίηση αποτελεί μία εναλλακτική λύση σε σχέση με τη συμμετρική πολυεπεξεργασία, ως προσέγγιση η οποία παρέχει υψηλή απόδοση και διαθεσιμότητα και είναι ιδιαίτερα ελκυστική για εφαρμογές εξυπηρετών. Μπορούμε να ορίσουμε ως συστάδα ένα σύνολο διασυνδεδεμένων, αυτόνομων υπολογιστών, οι οποίοι εργάζονται μαζί ως ενοποιημένος υπολογιστικός πόρος, δημιουργώντας την ψευδαίσθηση ότι πρόκειται για μία μηχανή. Ο όρος *αυτόνομος υπολογιστής* σημαίνει ότι το σύστημα έχει τη δυνατότητα να λειτουργήσει από μόνο του, πέραν της συστάδος. Στην αρθρογραφία, κάθε υπολογιστής μίας συστάδας, αναφέρεται τυπικά ως *κόμβος*.

Στην αναφορά [BREW97] παρατίθενται τέσσερα ωφέλη της συσταδοποίησης. Αυτά, θεωρούνται είτε ως στόχοι, είτε ως απαιτήσεις σχεδίασης:

- **Απόλυτη κλιμάκωση:** Είναι εφικτό να δημιουργήσουμε μεγάλες συστάδες οι οποίες ξεπερνούν κατά πολύ την ισχύ ακόμη και των μεγαλύτερων αυτόνομων μηχανών. Μία συστάδα μπορεί να περιέχει δεκάδες, εκατοντάδες, ακόμη και χιλιάδες μηχανές, κάθε μία από τις οποίες είναι ένας πολυεπεξεργαστής.
- **Επαυξητική κλιμάκωση:** Μία συστάδα διαμορφώνεται με τέτοιο τρόπο ώστε να είναι εφικτή η προσθήκη νέων συστημάτων σε αυτή, με μικρές επαυξήσεις. Επομένως, ένας χρήστης είναι δυνατόν να ξεκινήσει με ένα μεσαίου μεγέθους σύστημα, το οποίο να επεκτείνει καθώς αυξάνονται οι ανάγκες, χωρίς να πρέπει να προβεί σε μεγάλη αναβάθμιση αντικαθιστώντας το υπάρχον μικρό σύστημα με ένα μεγαλύτερο.



BXU = Μονάδα εκτέλεσης διακλάδωσης
CRL = Καταχωρητής συνθήκης λογικής μονάδας εκτέλεσης
FPU = Μονάδα εκτέλεσης πράξεων κινητής υποδιαστολής
FXU = Μονάδα εκτέλεσης πράξεων σταθερής υποδιαστολής
LSU = Μονάδα φόρτωσης/αποθήκευσης

Σχήμα 17.9: Ροή δεδομένων εντολών του επεξεργαστή Power5

- **Υψηλή διαθεσιμότητα:** Λόγω του γεγονότος ότι κάθε κόμβος σε μία συστάδα αποτελεί αυτόνομο υπολογιστή, η βλάβη ενός κόμβου δεν σημαίνει την απώλεια της λειτουργίας της συστάδας. Σε πολλά προϊόντα, η ανοχή σφαλμάτων αντιμετωπίζεται αυτόματα σε επίπεδο λογισμικού.

- **Ανώτερη τιμή/απόδοση:** Χρησιμοποιώντας δομικά τμήματα, είναι εφικτό να συναρμολογήσουμε τους υπολογιστές μίας συστάδας με ίση ή μεγαλύτερη υπολογιστική ισχύ σε σύγκριση με μία απλή μεγάλη μηχανή, και μάλιστα με πολύ χαμηλό κόστος.

Διατάξεις Συστάδων

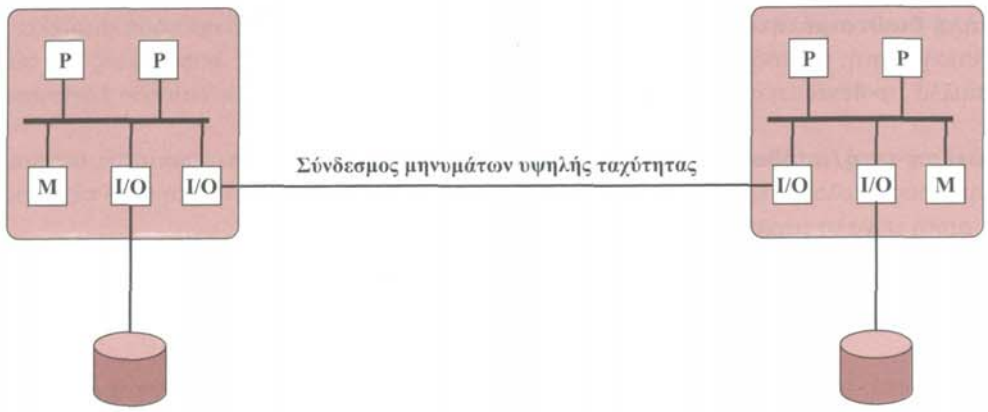
Στην αρθρογραφία, οι συστάδες κατηγοριοποιούνται με ένα πλήθος διαφορετικών τρόπων. Πιθανόν, η πιο απλή κατηγοριοποίηση βασίζεται στο αν οι υπολογιστές της συστάδας προσπελαύνουν κοινούς δίσκους. Το Σχήμα 17.10α απεικονίζει μία συστάδα αποτελούμενη από δύο κόμβους, όπου η μοναδική διασύνδεση είναι ένας σύνδεσμος υψηλής ταχύτητας, ο οποίος χρησιμοποιείται για την ανταλλαγή μηνυμάτων, ώστε να συντονίζεται η λειτουργία της συστάδας. Ο σύνδεσμος μπορεί να είναι ένα τοπικό δίκτυο το οποίο μοιράζεται με άλλους υπολογιστές οι οποίοι δεν αποτελούν μέρος της συστάδας, ή μπορεί να αποτελεί μία δεσμευμένη μονάδα διασύνδεσης. Στην τελευταία περίπτωση, ένας ή περισσότεροι υπολογιστές της συστάδας θα διαθέτουν ένα σύνδεσμο σε ένα τοπικό δίκτυο ή δίκτυο ευρείας περιοχής, έτσι ώστε να υπάρχει σύνδεση ανάμεσα στη συστάδα εξυπηρέτη και τα απομακρυσμένα συστήματα πελατών. Παρατηρείστε ότι στο σχήμα, κάθε υπολογιστής απεικονίζεται ως πολυεπεξεργαστής. Αυτό δεν είναι αναγκαίο, αλλά ενισχύει την απόδοση και τη διαθεσιμότητα.

Στην απλή κατηγοριοποίηση που απεικονίζεται στο Σχήμα 17.10, η άλλη εναλλακτική λύση είναι μία συστάδα κονόχρηστων δίσκων. Σε αυτή την περίπτωση, εξακολουθεί γενικά να υπάρχει ένας σύνδεσμος για την ανταλλαγή μηνυμάτων μεταξύ των κόμβων. Επιπλέον, υπάρχει ένα υποσύστημα δίσκων το οποίο συνδέεται άμεσα με πολλούς υπολογιστές εντός της συστάδας. Σε αυτό το σχήμα, το συνηθισμένο υποσύστημα είναι τεχνολογίας RAID. Η χρήση της τεχνολογίας RAID ή κάποιας άλλης τεχνολογίας πλεοναζουσών πληροφοριών είναι συνηθισμένη στις δομές συστάδων, έτσι ώστε η υψηλή διαθεσιμότητα η οποία οφείλεται στην παρουσία πολλαπλών υπολογιστών να μην μειώνεται από έναν κοινόχρηστο δίσκο ο οποίος έχει υποστεί βλάβη.

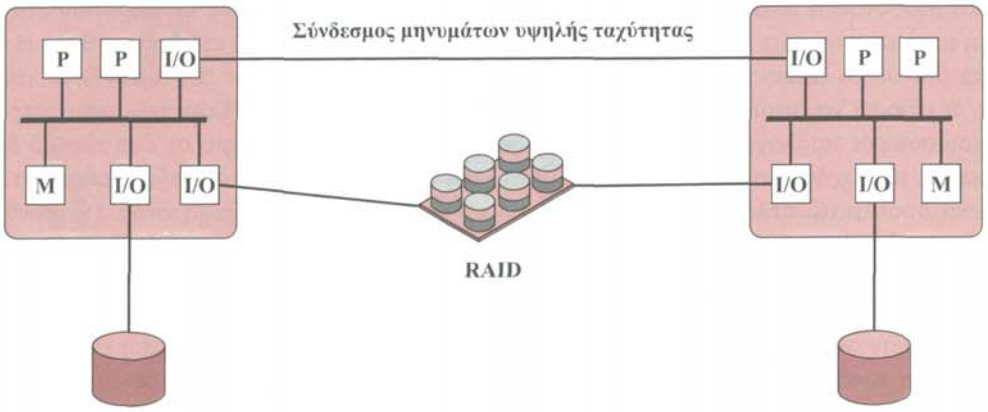
Μία πιο ξεκάθαρη εικόνα του εύρους των επιλογών λαμβάνεται αν εξεταστούν οι λειτουργικές εναλλακτικές λύσεις. Ο Πίνακας 17.2 παρέχει μία χρήσιμη κατηγοριοποίηση βάσει λειτουργιών, την οποία θα συζητήσουμε σε αυτό το σημείο.

Μία συνηθισμένη, πιο παλιά μέθοδος, γνωστή ως **παθητική αναμονή**, λειτουργεί απλώς έχοντας έναν υπολογιστή υπεύθυνο για τη διαχείριση ολόκληρου του φορτίου επεξεργασίας, ενώ ο άλλος υπολογιστής παραμένει ανενεργός, περιμένοντας να αναλάβει σε περίπτωση βλάβης του κύριου υπολογιστή. Για να συγχρονιστούν οι μηχανές, το ενεργό ή κύριο σύστημα στέλνει περιοδικά ένα μήνυμα-"καρδιοχτύπι" προς τη μηχανή η οποία βρίσκεται σε αναμονή. Αν σταματήσει η άφιξη μηνυμάτων αυτής της μορφής, η μηχανή που βρίσκεται σε αναμονή υποθέτει ότι ο κύριος εξυπηρέτης έχει εμφανίσει βλάβη και ενεργοποιείται. Αυτή η προσέγγιση αυξάνει τη διαθεσιμότητα, αλλά δεν βελτιώνει την απόδοση. Επιπλέον, αν οι μοναδικές πληροφορίες οι οποίες ανταλλάσσονται ανάμεσα στα δύο συστήματα είναι αυτό το μήνυμα και αν τα δύο συστήματα δεν μοιράζονται κοινούς δίσκους, τότε η μηχανή που βρίσκεται σε αναμονή παρέχει μία λειτουργική εφεδρεία, αλλά δεν έχει πρόσβαση σε καμία από τις βάσεις δεδομένων που χειρίζεται η κύρια.

Γενικά, η παθητική αναμονή δεν αναφέρεται ως συστάδα. Ο όρος *συστάδα* είναι δεσμευμένος για να περιγράφει πολλούς διασυνδεδεμένους υπολογιστές οι οποίοι εκτελούν εργασίες επεξεργασίας με ενεργό τρόπο ενώ διατηρούν την εικόνα ενός ενιαίου συστήματος προς τον έξω κόσμο. Ο όρος **ενεργός δευτερεύων εξυπηρέτης** χρησιμοποιείται συχνά για να αναφερθούμε σε αυτή τη διάταξη. Ορίζονται τρεις κατηγορίες συσταδοποίησης: ξεχωριστοί εξυπηρέτες, τίποτε κοινόχρηστο, και κοινόχρηστη μνήμη.



(α) Εξυπηρέτης σε αναμονή χωρίς κοινόχρηστο δίσκο



(β) Κοινόχρηστος δίσκος

Σχήμα 17.10: Διατάξεις συστάδων

Σε μία προσέγγιση της συσταδοποίησης, κάθε υπολογιστής είναι ένας **ξεχωριστός εξυπηρέτης** με τους δικούς του δίσκους και δεν υπάρχουν κοινόχρηστοι δίσκοι μεταξύ των συστημάτων (Σχήμα 17.10α). Αυτή η διάταξη δίνει υψηλή απόδοση και διαθεσιμότητα. Σε αυτή την περίπτωση, είναι αναγκαία κάποια μορφή λογισμικού διαχείρισης ή χρονοδρομολόγησης με την οποία οι εισερχόμενες αιτήσεις πελατών αποδίδονται στους εξυπηρέτες, με τέτοιο τρόπο ώστε να μοιράζεται εξίσου το φορτίο και να επιτυγχάνεται υψηλός βαθμός χρησιμοποίησης. Είναι επιθυμητό να υπάρχει δυνατότητα αποκατάστασης σε περίπτωση βλάβης, κάτι που σημαίνει ότι αν ένας υπολογιστής πάθει βλάβη κατά τη διάρκεια εκτέλεσης μίας εφαρμογής, υπάρχει δυνατότητα να αναλάβει και να ολοκληρώσει την εφαρμογή ένας άλλος υπολογιστής. Για να συμβεί αυτό, τα δεδομένα θα πρέπει να αντιγράφονται ανά σταθερά διαστήματα μεταξύ των συστημάτων, ώστε κάθε σύστημα να έχει πρόσβαση στα τρέχοντα δεδομένα των άλλων. Η επιβάρυνση αυτής της ανταλλαγής δεδομένων εξασφαλίζει την υψηλή διαθεσιμότητα με κόστος μία μικρή μείωση της απόδοσης.

Για να μειωθεί η επιβάρυνση που οφείλεται στην επικοινωνία, οι περισσότερες συστάδες αποτελούνται τώρα από εξυπηρέτες οι οποίοι συνδέονται σε κοινόχρηστους δίσκους (Σχήμα 17.10β). Σε μία παραλλαγή αυτής της προσέγγισης με την ονομασία **τίποτε κοινόχρηστο**, οι δίσκοι χωρίζονται σε τόμους, και κάθε τόμος ανήκει σε έναν υπολογιστή. Αν αυτός ο υπολογιστής υποστεί βλάβη, η συστάδα θα πρέπει να αναδιαταχθεί έτσι ώστε ένας άλλος υπολογιστής να λάβει στην κατοχή του τους τόμους εκείνου που έπαθε τη βλάβη.

Επίσης, είναι δυνατόν να υπάρχουν πολλοί υπολογιστές οι οποίοι μοιράζονται τους ίδιους δίσκους

Πίνακας 17.2: Μέθοδοι συσταδοποίησης: Ωφέλη και περιορισμοί

Μέθοδος Συσταδοποίησης	Περιγραφή	Ωφέλη	Περιορισμοί
Παθητική Αναμονή	Ένας δευτερεύων εξυπηρέτης αναλαμβάνει σε περίπτωση βλάβης του κύριου εξυπηρέτη.	Εύκολη υλοποίηση.	Υψηλό κόστος λόγω του γεγονότος ότι ο δευτερεύων εξυπηρέτης δεν είναι διαθέσιμος για άλλες εργασίες επεξεργασίας.
Ενεργός Δευτερεύων Εξυπηρέτης	Ο δευτερεύων εξυπηρέτης χρησιμοποιείται επίσης για την επεξεργασία εργασιών.	Μειωμένο κόστος επειδή οι δευτερεύοντες εξυπηρέτες είναι δυνατόν να χρησιμοποιηθούν για επεξεργασία.	Αυξημένη πολυπλοκότητα.
Ξεχωριστοί εξυπηρέτες	Οι ξεχωριστοί εξυπηρέτες έχουν τους δικούς τους δίσκους. Τα δεδομένα αντιγράφονται συνεχώς από τον κύριο στο δευτερεύοντα εξυπηρέτη.	Υψηλή διαθεσιμότητα.	Υψηλή επιβάρυνση στο δίκτυο και στον εξυπηρέτη, λόγω των αντιγραφών που λαμβάνουν χώρα.
Οι εξυπηρέτες συνδέονται στους δίσκους	Οι εξυπηρέτες συνδέονται στους ίδιους δίσκους, αλλά καθένας έχει τους δικούς του. Αν ένας εξυπηρέτης παρουσιάσει βλάβη, ο άλλος θα αναλάβει και θα χρησιμοποιήσει αυτούς τους δίσκους.	Μειώνεται η επιβάρυνση στο δίκτυο και στους εξυπηρέτες, λόγω της εξάλειψης των λειτουργιών αντιγραφής.	Συνήθως απαιτείται καθρέπτισμα δίσκων ή τεχνολογία RAID για να αντισταθμιστεί ο κίνδυνος που υπάρχει από πιθανή βλάβη ενός δίσκου.
Οι εξυπηρέτες μοιράζονται τους δίσκους	Πολλοί εξυπηρέτες μοιράζονται ταυτόχρονα την προσπέλαση των δίσκων	Μειωμένη επιβάρυνση στο δίκτυο και στους εξυπηρέτες. Μειωμένος κίνδυνος διακοπής λειτουργίας του συστήματος λόγω βλάβης ενός δίσκου.	Απαιτείται υλικό διαχείρισης κλειδώματος. Συνήθως, αυτό χρησιμοποιείται με την απεικόνιση δίσκων ή την τεχνολογία RAID.

την ίδια χρονική στιγμή (η προσέγγιση αυτή ονομάζεται προσέγγιση **κοινόχρηστων δίσκων**), έτσι ώστε κάθε υπολογιστής να έχει πρόσβαση σε όλους τους τόμους όλων των δίσκων. Αυτή η προσέγγιση απαιτεί τη χρήση μίας δυνατότητας κλειδώματος, ώστε να εξασφαλιστεί ότι τα δεδομένα προσπελαύνονται από έναν υπολογιστή κάθε φορά.

Ζητήματα Σχεδίασης Λειτουργικών Συστημάτων

Η πλήρης εκμετάλλευση της διάταξης υλικού μίας συστάδας απαιτεί ορισμένες βελτιώσεις πάνω στο λειτουργικό σύστημα ενός απλού συστήματος.

ΔΙΑΧΕΙΡΙΣΗ ΒΛΑΒΩΝ Ο τρόπος διαχείρισης των βλαβών εξαρτάται από τη μέθοδο συσταδοποίησης η οποία χρησιμοποιείται (Πίνακας 17.2). Γενικά, δύο προσεγγίσεις ακολουθούνται όταν συναντάμε βλάβες: συστάδες υψηλής διαθεσιμότητας και συστάδες με ανοχή στα σφάλματα. Μία συστάδα με υψηλή διαθεσιμότητα προσφέρει μεγάλη πιθανότητα όλοι οι πόροι να είναι διαθέσιμοι. Αν συμβεί βλάβη, όπως για παράδειγμα καταρρεύσει ένα σύστημα ή χαθεί ένας τόμος κάποιου δίσκου, τότε

χάνονται όλα τα ερωτήματα το οποία βρισκόταν σε επεξεργασία. Οποιοδήποτε χαμένο ερώτημα, σε περίπτωση που τεθεί πάλι, θα εξυπηρετηθεί από άλλο υπολογιστή της συστάδας. Ωστόσο, το λειτουργικό σύστημα της συστάδας δεν παρέχει καμία εγγύηση για την κατάσταση των μερικώς εκτελεσμένων συναλλαγών. Αυτό θα πρέπει να γίνει σε επίπεδο εφαρμογής.

Μία συστάδα με ανοχή σε σφάλματα εξασφαλίζει ότι όλοι οι πόροι είναι πάντοτε διαθέσιμοι. Αυτό επιτυγχάνεται με τη χρήση πλεονάζοντων κοινόχρηστων δίσκων και μηχανισμών εφεδρικής αποθήκευσης μη ολοκληρωμένων συναλλαγών και αναφοράς των ολοκληρωμένων συναλλαγών.

Η λειτουργία της μεταφοράς εφαρμογών και πόρων δεδομένων από ένα σύστημα το οποίο έχει υποστεί ζημιά σε ένα εναλλακτικό σύστημα της συστάδας, ονομάζεται **βλάβη με μεταφορά** (failover). Μία σχετική λειτουργία, είναι η αποκατάσταση των εφαρμογών και των πόρων δεδομένων στο αρχικό σύστημα, αφότου αυτό επιδιορθωθεί. Αυτή η λειτουργία αναφέρεται ως **βλάβη με επιστροφή** (failback). Η βλάβη με επιστροφή είναι δυνατόν να αυτοματοποιηθεί, αλλά αυτό είναι επιθυμητό μόνον αν το πρόβλημα έχει πραγματικά επιδιορθωθεί και είναι απίθανο να εμφανιστεί ξανά. Αν δεν συμβαίνει αυτό, ο αυτοματισμός μπορεί να προκαλέσει τη μεταφορά ανάμεσα σε διάφορους υπολογιστές, πόρων οι οποίοι υφίστανται βλάβες μελλοντικά. Αυτό το γεγονός θα έχει ως αποτέλεσμα τη μείωση της απόδοσης και την εμφάνιση προβλημάτων αποκατάστασης.

ΙΣΟΡΡΟΠΙΑ ΦΟΡΤΙΟΥ Μία συστάδα απαιτεί τη δυνατότητα *εξισορρόπησης του φορτίου* ανάμεσα στους υπολογιστές που την αποτελούν. Αυτό περιλαμβάνει την απαίτηση η συστάδα να είναι επαυξητικά κλιμακωτή. Όταν προστίθεται στη συστάδα ένας νέος υπολογιστής, θα πρέπει να συμπεριλαμβάνεται αυτομάτως σε αυτή την απαίτηση εξισορρόπησης. Οι μηχανισμοί του ενδιάμεσου στρώματος λογισμικού θα πρέπει να αναγνωρίζουν ότι είναι δυνατή η εμφάνιση υπηρεσιών σε διαφορετικά μέλη της συστάδας, αλλά και η μετακόμισή τους από ένα μέλος σε ένα άλλο.

ΠΑΡΑΛΛΗΛΙΣΜΟΣ ΥΠΟΛΟΓΙΣΜΩΝ Σε μερικές περιπτώσεις, η αποτελεσματική χρήση μίας συστάδας απαιτεί την παράλληλη εκτέλεση του λογισμικού μίας εφαρμογής. Στην αναφορά [KAPP00] παρατίθενται τρεις γενικές προσεγγίσεις σε αυτό το πρόβλημα:

- **Παραλληλοποίηση του μεταγλωττιστή:** Ένας παραλληλοποιημένος μεταγλωττιστής προσδιορίζει, κατά τη διάρκεια της μεταγλώττισης, ποια μέρη της εφαρμογής είναι δυνατόν να εκτελεστούν παράλληλα. Στη συνέχεια, τα μέρη αυτά χωρίζονται ώστε να εκχωρηθούν σε ξεχωριστούς υπολογιστές της συστάδας. Η απόδοση εξαρτάται από τη φύση του προβλήματος και από το πόσο καλά είναι σχεδιασμένος ο μεταγλωττιστής. Γενικά, οι μεταγλωττιστές αυτής της μορφής δεν είναι εύκολοι στην υλοποίηση.

- **Παραλληλοποίηση της εφαρμογής:** Σε αυτή την προσέγγιση, ο προγραμματιστής γράφει την εφαρμογή από την αρχή, έτσι ώστε αυτή να εκτελείται σε μία συστάδα και χρησιμοποιεί την ανταλλαγή μηνυμάτων για τη μετακίνηση των δεδομένων μεταξύ των κόμβων, όπου αυτό χρειάζεται. Με τον τρόπο αυτό επιβαρύνεται σημαντικά ο προγραμματιστής, αλλά ίσως είναι η πιο καλή προσέγγιση όσον αφορά την εκμετάλλευση των συστάδων από κάποιες εφαρμογές.

- **Παραμετρικοί υπολογισμοί:** Αυτή η προσέγγιση χρησιμοποιείται αν η ουσία της εφαρμογής είναι ένας αλγόριθμος ή πρόγραμμα το οποίο πρέπει να εκτελεστεί πολλές φορές, κάθε φορά με διαφορετικό σύνολο συνθηκών και παραμέτρων εκκίνησης. Ένα καλό παράδειγμα αποτελεί ένα μοντέλο προσομοίωσης, το οποίο θα εκτελέσει ένα μεγάλο πλήθος διαφορετικών σεναρίων και θα δώσει συνοπτικά στατιστικά αποτελέσματα. Για να είναι αποτελεσματική αυτή η προσέγγιση, είναι αναγκαία κάποια εργαλεία παραμετρικής επεξεργασίας, τα οποία θα οργανώσουν, θα εκτελέσουν, και θα διαχειριστούν αποτελεσματικά τις εργασίες.

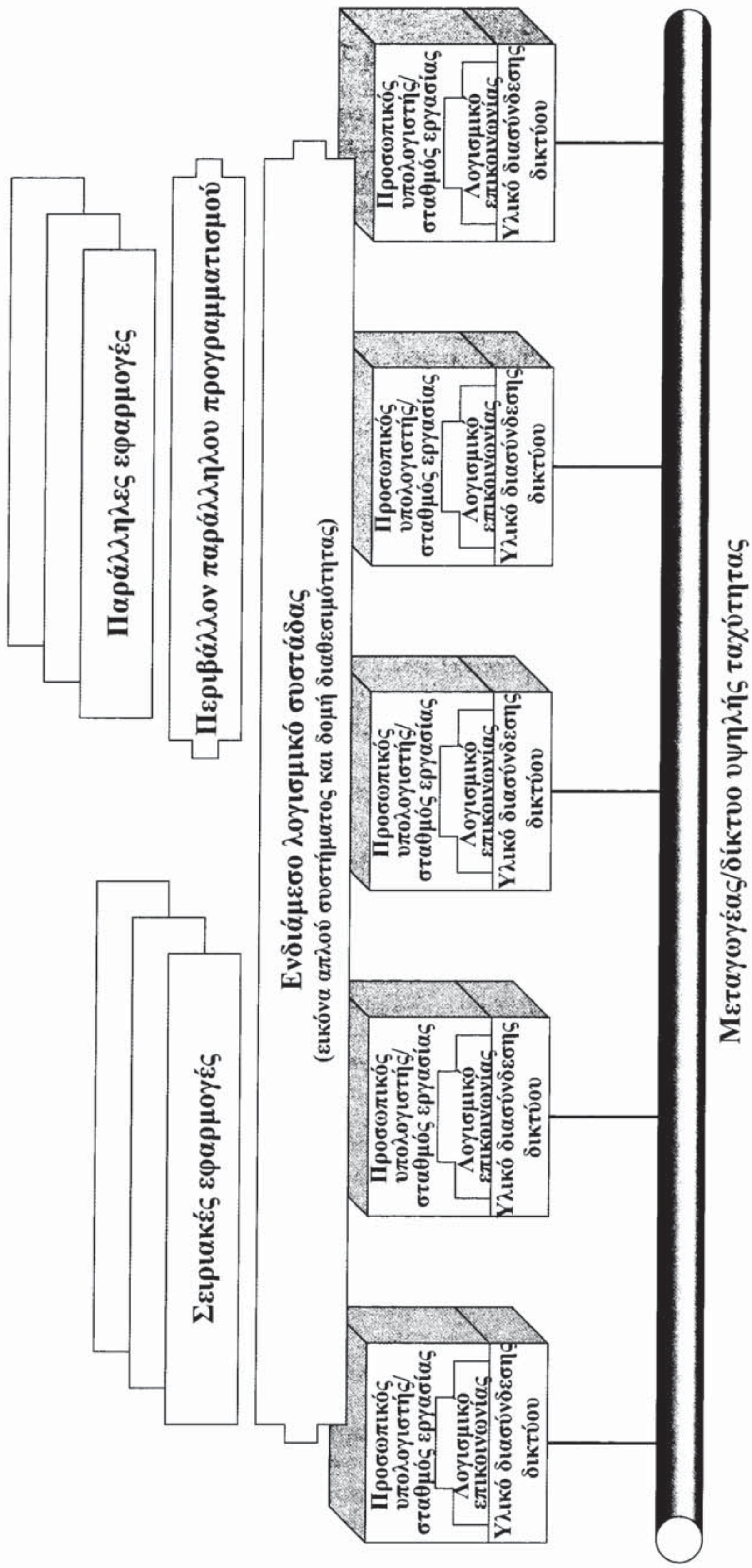
Αρχιτεκτονική Υπολογιστών σε Μορφή Συστάδων

Το Σχήμα 17.11 απεικονίζει μία τυπική αρχιτεκτονική συστάδας υπολογιστών. Οι υπολογιστές συνδέονται με κάποιο τοπικό δίκτυο υψηλής ταχύτητας ή με κάποιο υλικό μεταγωγής. Κάθε υπολογιστής έχει τη δυνατότητα να λειτουργεί ανεξάρτητα. Επιπλέον, σε κάθε υπολογιστή υπάρχει εγκατεστημένο ένα ενδιάμεσο στρώμα λογισμικού, ώστε να επιτρέπει τη λειτουργία της συστάδας. Το ενδιάμεσο αυτό στρώμα παρέχει μία εικόνα ενοποιημένου συστήματος στο χρήστη, η οποία είναι γνωστή και ως **εικόνα μονού συστήματος**. Το ενδιάμεσο στρώμα είναι επίσης υπεύθυνο για να παρέχει υψηλή διαθεσιμότητα, μέσα από την εξισορρόπηση του φορτίου και την απόκριση σε βλάβες των διαφόρων στοιχείων. Στην αναφορά HWAN99 παρατίθενται οι ακόλουθες επιθυμητές υπηρεσίες και λειτουργίες του ενδιάμεσου στρώματος λογισμικού μίας συστάδας:

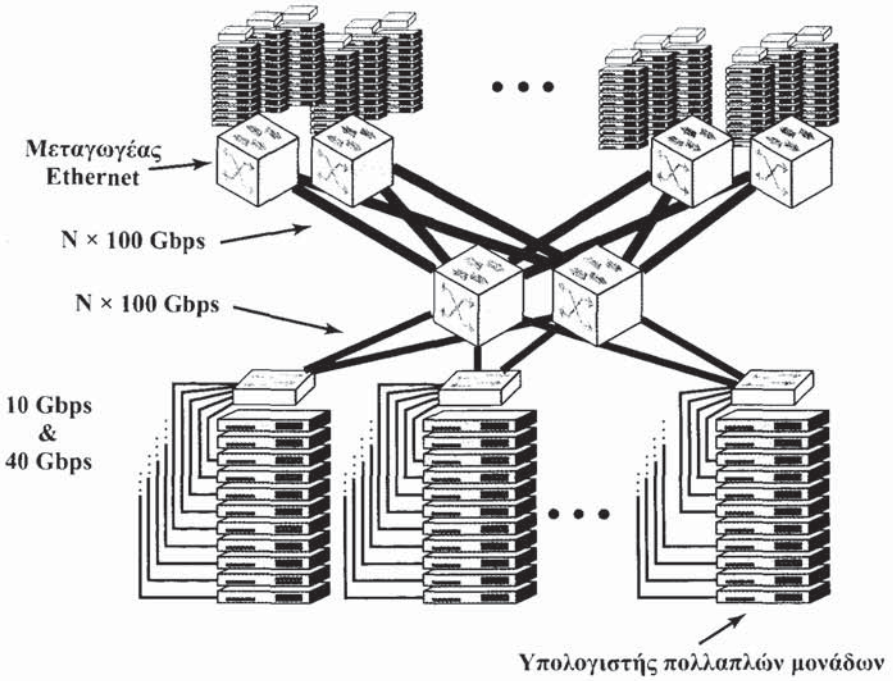
- **Μονό σημείο εισόδου:** Ένας χρήστης εισέρχεται σε μία συστάδα και όχι σε μία ξεχωριστή μηχανή.
- **Μονή ιεραρχία αρχείων:** Ο χρήστης βλέπει μία μονή ιεραρχία καταλόγων αρχείων κάτω από τον ίδιο κατάλογο-ρίζα.
- **Μονό σημείο ελέγχου:** Υπάρχει ένας προεπιλεγμένος σταθμός εργασίας ο οποίος χρησιμοποιείται για τη διαχείριση και τον έλεγχο της συστάδας.
- **Μονή ιδεατή δικτύωση:** Κάθε κόμβος έχει τη δυνατότητα προσπέλασης κάθε άλλου σημείου της συστάδας, ακόμη και αν η πραγματική διάταξη της αποτελείται από μεγάλο πλήθος διασυνδεδεμένων δικτύων. Υπάρχει μία απλή λειτουργία ιδεατού δικτύου.
- **Μονός χώρος μνήμης:** Η κατανεμημένη κοινόχρηστη μνήμη επιτρέπει στα προγράμματα να μοιράζονται μεταβλητές.
- **Μονό σύστημα διαχείρισης εργασιών:** Με τη βοήθεια ενός χρονοδρομολογητή εργασιών της συστάδας, ένας χρήστης μπορεί να υποβάλλει μία εργασία χωρίς να καθορίσει τον υπολογιστή ο οποίος θα την εκτελέσει.
- **Μονή διεπαφή χρήστη:** Μία μονή διεπαφή με χρήση γραφικών υποστηρίζει όλους τους χρήστες, ανεξάρτητα από το σταθμό από τον οποίο εισήλθαν στη συστάδα.
- **Μονός χώρος Ε/Ε:** Κάθε κόμβος μπορεί να προσπελάσει απομακρυσμένα κάθε περιφερειακή μονάδα Ε/Ε ή κάθε δίσκο, χωρίς να γνωρίζει τίποτα για τη φυσική τους θέση.
- **Μονός χώρος διεργασιών:** Χρησιμοποιείται μία ομοιόμορφη τεχνική αναγνώρισης διεργασιών. Μία διεργασία σε οποιονδήποτε κόμβο έχει τη δυνατότητα να δημιουργήσει μία άλλη διεργασία ή να επικοινωνήσει με μία άλλη διεργασία, σε έναν απομακρυσμένο κόμβο.
- **Έλεγχος σημείων:** Αυτή η λειτουργία αποθηκεύει περιοδικά την κατάσταση μίας διεργασίας και τα ενδιάμεσα αποτελέσματα των υπολογισμών, ώστε να υπάρχει δυνατότητα αποκατάστασης μετά από κάποια βλάβη.
- **Μετακόμιση διεργασίας:** Αυτή η λειτουργία επιτρέπει την εξισορρόπηση του φορτίου.

Τα τέσσερα τελευταία στοιχεία της παραπάνω λίστας ενισχύουν τη διαθεσιμότητα της συστάδας. Τα υπόλοιπα επικεντρώνονται στην παροχή μίας εικόνας ενιαίου συστήματος.

Επιστρέφοντας στο Σχήμα 17.11, μία συστάδα θα περιέχει επίσης εργαλεία λογισμικού μέσω των οποίων θα επιτρέπεται η αποτελεσματική εκτέλεση προγραμμάτων τα οποία έχουν τη δυνατότητα να εκτελεστούν παράλληλα.



Σχήμα 17.11: Αρχιτεκτονική συστάδας υπολογιστών [BUYY99a]



Σχήμα 17.12: Παράδειγμα διάταξης δικτύου Ethernet 100 Gbps για ένα μεγάλο σύστημα εξυπηρετών φέτας.

Φέτες Εξυπηρετών

Μία συνηθισμένη υλοποίηση των συστάδων είναι ο εξυπηρέτης φετών. Πρόκειται για μία αρχιτεκτονική εξυπηρετών όπου στεγάζονται πολλές μονάδες εξυπηρετών ("φέτες") σε ένα απλό σασσί. Η αρχιτεκτονική αυτή χρησιμοποιείται εκτενώς σε κέντρα δεδομένων, ώστε να εξοικονομείται χώρος και να βελτιώνεται η διαχείριση του συστήματος. Είτε αυτόνομο, είτε τοποθετημένο σε βάση στήριξης, το σασσί παρέχει το ρεύμα και κάθε φέτα διαθέτει το δικό της επεξεργαστή, μνήμη, και σκληρό δίσκο.

Ένα παράδειγμα της εφαρμογής απεικονίζεται στο Σχήμα 17.12, το οποίο έχει ληφθεί από την αναφορά [NOWE07]. Η τάση που υπάρχει στα μεγάλα κέντρα δεδομένων με τράπεζες εξυπηρετών φετών, είναι να χρησιμοποιούνται θύρες ταχύτητας 10 Gbps σε κάθε εξυπηρέτη, ώστε να διαχειρίζεται τη μεγάλη πολυμεσική κυκλοφορία την οποία παράγουν αυτοί οι εξυπηρέτες. Αυτές οι διατάξεις βασίζονται σε μεταγωγείς Ethernet, οι οποίοι είναι αναγκαίοι για να διασυνδεθούν μεγάλα πλήθη εξυπηρετών. Ένας ρυθμός 100 Gbps παρέχει το απαιτούμενο εύρος ζώνης για τη διαχείριση του αυξανόμενου όγκου δεδομένων. Οι μεταγωγείς Ethernet 100 Gbps χρησιμοποιούνται σε ανοδικούς συνδέσμους μέσα στο κέντρο δεδομένων, ενώ παρέχουν και διασύνδεση για επιχειρησιακά δίκτυα, σε επίπεδο κτιρίου, σε τοπικό επίπεδο, αλλά και σε επίπεδο ευρείας περιοχής.

Συστάδες σε Σύγκριση με Συμμετρικούς Πολυπεπεξεργαστές

Τόσο οι συστάδες, όσο και οι συμμετρικοί πολυεπεξεργαστές παρέχουν μία διάταξη πολλαπλών επεξεργαστών για να υποστηρίζουν εφαρμογές υψηλών απαιτήσεων. Αμφότερες οι λύσεις είναι διαθέσιμες στο εμπόριο, παρά το γεγονός ότι οι συμμετρικοί πολυεπεξεργαστές βρίσκονται στο εμπόριο πολύ περισσότερο καιρό.

Το δυνατό σημείο των συμμετρικών πολυεπεξεργαστών είναι ότι είναι πιο εύκολοι στη διαχείριση και στη διαμόρφωση σε σχέση με τη συστάδα. Ο συμμετρικός πολυεπεξεργαστής βρίσκεται πολύ πιο κοντά στο μοντέλο του απλού επεξεργαστή για το οποίο είναι γραμμένες οι πιο πολλές εφαρμογές. Η

βασική αλλαγή η οποία απαιτείται για τη μετάβαση από έναν απλό επεξεργαστή σε ένα συμμετρικό πολυεπεξεργαστή, είναι η λειτουργία του χρονοδρομολογητή. Ένα άλλο κέρδος που προκύπτει από τη χρήση του συμμετρικού πολυεπεξεργαστή, είναι ότι καταλαμβάνει πολύ μικρότερο χώρο και καταναλώνει λιγότερη ενέργεια σε σχέση με τη συστάδα. Ένα τελευταίο σημαντικό σημείο, είναι ότι τα προϊόντα συμμετρικών πολυεπεξεργαστών είναι καθιερωμένα και σταθερά.

Ωστόσο, μακροπρόθεσμα, τα πλεονεκτήματα της προσέγγισης των συστάδων είναι τέτοια που πιθανόν να είναι οι συστάδες οι οποίες θα κυριαρχήσουν στην αγορά. Οι συστάδες είναι πολύ ανώτερες των συμμετρικών πολυεπεξεργαστών τόσο με όρους της επαυξητικής όσο και με όρους της απόλυτης κλιμάκωσης. Οι συστάδες είναι επίσης ανώτερες με όρους της διαθεσιμότητας, επειδή, με ευκολία, όλα τα στοιχεία του συστήματος είναι δυνατόν να καταστούν σε μεγάλο βαθμό υπεράριθμα.

17.6 ΜΗ ΟΜΟΙΟΜΟΡΦΗ ΠΡΟΣΠΕΛΑΣΗ ΜΝΗΜΗΣ

Με όρους των εμπορικών προϊόντων, οι δύο συνηθισμένες προσεγγίσεις για την παροχή ενός συστήματος πολλαπλών επεξεργαστών υποστήριξης εφαρμογών είναι οι συμμετρικοί πολυεπεξεργαστές και οι συστάδες. Για ορισμένα χρόνια, μία άλλη προσέγγιση γνωστή και ως μη ομοιόμορφη προσπέλαση μνήμης (Nonuniform Memory Access-NUMA), αποτέλεσε αντικείμενο έρευνας και τώρα, τα προϊόντα NUMA είναι διαθέσιμα στο εμπόριο.

Πριν προχωρήσουμε, θα δώσουμε μερικούς ορισμούς οι οποίοι υπάρχουν στην αρθρογραφία σχετικά με την προσέγγιση αυτή.

- **Ομοιόμορφη προσπέλαση μνήμης (Uniform Memory Access-UMA):** Όλοι οι επεξεργαστές έχουν πρόσβαση σε όλα τα μέρη της κύριας μνήμης χρησιμοποιώντας λειτουργίες φόρτωσης και αποθήκευσης. Ο χρόνος προσπέλασης κάθε περιοχής της μνήμης είναι ίδιος για έναν επεξεργαστή. Οι χρόνοι προσπέλασης διαφορετικών επεξεργαστών είναι ίδιοι. Η οργάνωση των συμμετρικών πολυεπεξεργαστών που παρουσιάστηκε στις Ενότητες 17.2 και 17.3, είναι UMA.

- **Μη ομοιόμορφη προσπέλαση μνήμης (Non Uniform Memory Access-NUMA):** Όλοι οι επεξεργαστές έχουν πρόσβαση σε όλα τα μέρη της κύριας μνήμης χρησιμοποιώντας λειτουργίες φόρτωσης και αποθήκευσης. Ο χρόνος προσπέλασης της μνήμης διαφέρει αναλόγως με την περιοχή η οποία προσπελαύνεται. Η τελευταία πρόταση ισχύει για κάθε επεξεργαστή. Ωστόσο, για διαφορετικούς επεξεργαστές, διαφέρουν οι πιο αργές και οι πιο γρήγορες περιοχές μνήμης.

- **Μη ομοιόμορφη προσπέλαση μνήμης με συνοχή κρυφής μνήμης (Cache-Coherent Non Uniform Memory Access-CC-NUMA):** Ένα σύστημα NUMA στο οποίο διατηρείται η συνοχή ανάμεσα στις κρυφές μνήμες των διαφόρων επεξεργαστών.

Ένα σύστημα NUMA το οποίο δεν διαθέτει συνοχή κρυφής μνήμης, είναι περίπου παρόμοιο με μία συστάδα. Τα εμπορικά προϊόντα τα οποία έχουν συγκεντρώσει τη μεγαλύτερη προσοχή τα τελευταία χρόνια, είναι τα συστήματα CC-NUMA, τα οποία διαφέρουν αρκετά από τους συμμετρικούς πολυεπεξεργαστές και τις συστάδες. Στην πραγματικότητα, συνήθως αλλά δυστυχώς όχι πάντοτε, αυτά τα συστήματα αναφέρονται στην εμπορική αρθρογραφία ως CC-NUMA. Αυτή η ενότητα επικεντρώνεται μόνον σε αυτά τα συστήματα.

Κίνητρα

Σε ένα σύστημα συμμετρικών πολυεπεξεργαστών, υπάρχει ένα πρακτικό όριο σχετικά με το πλήθος των επεξεργαστών που είναι δυνατόν να χρησιμοποιηθούν. Μία αποτελεσματική τεχνική κρυφής μνήμης μειώνει την κυκλοφορία η οποία διεξάγεται μέσω του διαύλου, ανάμεσα σε οποιονδήποτε

επεξεργαστή και την κύρια μνήμη. Καθώς αυξάνεται το πλήθος των επεξεργαστών, αυξάνεται και αυτή η μορφή κυκλοφορίας. Επίσης, ο δίαυλος χρησιμοποιείται για την ανταλλαγή σημάτων τα οποία σχετίζονται με τη συνοχή της κρυφής μνήμης, κάτι το οποίο προσθέτει επιπλέον επιβάρυνση. Σε κάποιο σημείο, ο δίαυλος θα αποτελεί τροχοπέδη όσον αφορά την απόδοση. Η μείωση της απόδοσης φαίνεται ότι περιορίζει το πλήθος των επεξεργαστών σε μία διάταξη συμμετρικών πολυεπεξεργαστών, σε μία τιμή από 16-64 επεξεργαστές. Για παράδειγμα, η μηχανή Silicon Graphics Power Challenge SMP περιορίζεται σε 64 επεξεργαστές R10000. Πέρα από αυτό τον αριθμό, η απόδοση μειώνεται σημαντικά.

Το όριο του πλήθους των επεξεργαστών σε ένα σύστημα συμμετρικών πολυεπεξεργαστών είναι ένα από τα κίνητρα τα οποία οδήγησαν στη δημιουργία των συστάδων. Ωστόσο, σε μία συστάδα, κάθε κόμβος διαθέτει τη δική του μνήμη. Οι εφαρμογές δεν βλέπουν μία μεγάλη συνολική μνήμη. Στην πραγματικότητα, η συνοχή διατηρείται σε επίπεδο λογισμικού και όχι υλικού. Αυτή η διασπορά της μνήμης επηρρεάζει την απόδοση και για να επιτύχουμε μέγιστη απόδοση θα πρέπει να προσαρμόσουμε κάποιο λογισμικό σε αυτό το περιβάλλον. Μία προσέγγιση για την επίτευξη πολυεπεξεργασίας μεγάλης κλίμακας και ταυτόχρονα για τη διατήρηση ενός "αρώματος" συμμετρικών πολυεπεξεργαστών, είναι η προσέγγιση NUMA. Για παράδειγμα, το σύστημα NUMA Silicon Graphics Origin είναι σχεδιασμένο ώστε να υποστηρίζει μέχρι και 1024 επεξεργαστές MIPS R10000 [WHIT97], ενώ το σύστημα NUMA-Q είναι σχεδιασμένο ώστε να υποστηρίζει μέχρι και 252 επεξεργαστές Pentium II [LOVE96].

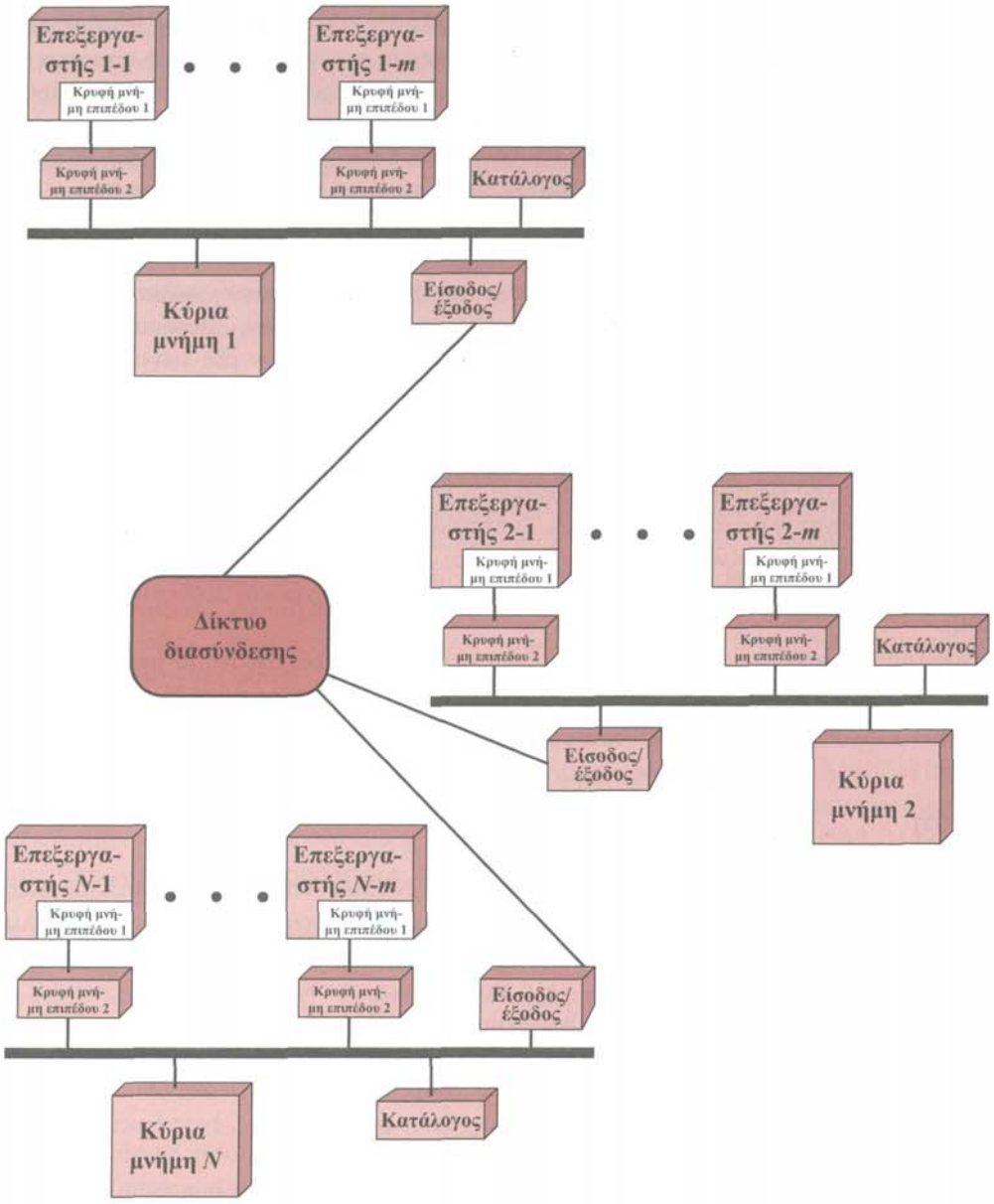
Ο στόχος της προσέγγισης NUMA είναι να διατηρεί ένα διαφανές σύστημα μεγάλης μνήμης ενώ επιτρέπει την ύπαρξη πολλαπλών κόμβων πολυεπεξεργαστών, καθένας από τους οποίους έχει το δικό του δίαυλο ή άλλο σύστημα διασύνδεσης.

Οργάνωση

Το Σχήμα 17.13 απεικονίζει μία τυπική οργάνωση CC-NUMA. Υπάρχουν πολλαπλοί ανεξάρτητοι κόμβοι, καθένας εκ των οποίων είναι, στην πραγματικότητα μία οργάνωση συμμετρικού πολυεπεξεργαστή. Επομένως, κάθε κόμβος περιέχει πολλούς επεξεργαστές, καθένας εκ των οποίων διαθέτει τη δική του κρυφή μνήμη Επιπέδου 1 και 2, αλλά και τη δική του κύρια μνήμη. Ο κόμβος αποτελεί το βασικό δομικό μπλοκ ολόκληρης της οργάνωσης CC-NUMA. Για παράδειγμα, κάθε κόμβος της μηχανής NUMA Silicon Graphics Origin περιέχει δύο επεξεργαστές MIPS R10000. Κάθε κόμβος της μηχανής Sequent NUMA-Q περιέχει τέσσερις επεξεργαστές Pentium II. Οι κόμβοι διασυνδέονται μέσω κάποιων μηχανισμών επικοινωνίας, όπως ένας μηχανισμός μεταγωγής, ένας δακτύλιος, ή κάποιος άλλος δικτυακός μηχανισμός.

Κάθε κόμβος του συστήματος CC-NUMA συμπεριλαμβάνει κάποια κύρια μνήμη. Ωστόσο, από την πλευρά των επεξεργαστών, υπάρχει μία μόνον διευθυνσιοδοτούμενη μνήμη και η κάθε θέση της χαρακτηρίζεται από μία μοναδική διεύθυνση συστήματος. Όταν ένας επεξεργαστής εκκινήσει μία προσπέλαση της μνήμης, αν η θέση αυτή δεν βρίσκεται στην κρυφή του μνήμη, τότε η κρυφή μνήμη Επιπέδου 2 ξεκινά μία λειτουργία ανάκλησης. Αν η επιθυμητή γραμμή βρίσκεται στο τοπικό κομμάτι της κύριας μνήμης, τότε στέλνεται μία αυτόματη αίτηση στο δίκτυο διασύνδεσης, ώστε να προσκομιστεί αυτή η γραμμή, να παραδοθεί στον τοπικό δίαυλο, και στη συνέχεια, να παραδοθεί στην κρυφή μνήμη που έκανε την αίτηση ανάκλησης. Όλη αυτή η δραστηριότητα είναι αυτοματοποιημένη και μη ορατή στον επεξεργαστή και την κρυφή μνήμη.

Σε αυτή τη διάταξη, η συνοχή της κρυφής μνήμης αποτελεί ένα σημαντικό ζήτημα. Παρά το γεγονός ότι οι διάφορες υλοποιήσεις διαφέρουν ως προς τις λεπτομέρειές τους, γενικά, μπορούμε να πούμε ότι κάθε κόμβος θα πρέπει να διατηρεί μία μορφή καταλόγου η οποία δίνει μία ένδειξη της θέσης διάφορων τμημάτων της μνήμης, αλλά και πληροφορίες σχετικά με την κατάσταση της κρυφής μνήμης. Για να εξετάσουμε τον τρόπο με τον οποίο λειτουργεί αυτή η τεχνική, θα δώσουμε ένα παράδειγμα το οποίο έχει ληφθεί από την αναφορά [PFIS98]. Έστω ότι ο επεξεργαστής 3 του κόμβου



Σχήμα 17.13: Οργάνωση CC-NUMA.

2 (P2-3) ζητά τη θέση μνήμης 798, η οποία βρίσκεται στη μνήμη του κόμβου 1. Στην περίπτωση αυτή, θα συμβεί η εξής ακολουθία γεγονότων:

1. Ο επεξεργαστής P2-3 εκδίδει μία αίτηση ανάγνωσης της θέσης 798, στο δίαυλο του κόμβου 2.
2. Ο κατάλογος του κόμβου 2 βλέπει την αίτηση και αναγνωρίζει ότι η θέση βρίσκεται στον κόμβο 1.
3. Ο κατάλογος του κόμβου 2 στέλνει μία αίτηση στον κόμβο 1, η οποία λαμβάνεται από τον κατάλογο του κόμβου 1.
4. Ο κατάλογος του κόμβου 1, ενεργώντας ως πληρεξούσιος του επεξεργαστή P2-3, ζητά τα περιε-

χόμενα της θέσης 798, σαν να ήταν ο ίδιος ο επεξεργαστής.

5. Η κύρια μνήμη του κόμβου 1 αποκρίνεται τοποθετώντας τα δεδομένα που έχουν ζητηθεί στο δίαυλο.

6. Ο κατάλογος του κόμβου 1 λαμβάνει τα δεδομένα από το δίαυλο.

7. Η τιμή μεταφέρεται πίσω στον κατάλογο του κόμβου 2.

8. Ο κατάλογος του κόμβου 2 τοποθετεί το μπλοκ δεδομένων στο δίαυλο του κόμβου 2, ενεργώντας ως πληρεξούσιος της μνήμης, η οποία αποθήκευε αρχικά αυτά τα δεδομένα.

9. Η τιμή λαμβάνεται και τοποθετείται στην κρυφή μνήμη του επεξεργαστή P2-3 και παραδίδεται σε εκείνον.

Η παραπάνω ακολουθία εξηγεί τον τρόπο με τον οποίο τα δεδομένα διαβάζονται από μία απομακρυσμένη μνήμη χρησιμοποιώντας μηχανισμούς του υλικού, οι οποίοι καθιστούν τη συναλλαγή μη ορατή για τον επεξεργαστή. Στην κορυφή αυτών των μηχανισμών, είναι αναγκαία η ύπαρξη μίας μορφής πρωτοκόλλου συνοχής της κρυφής μνήμης. Διάφορα συστήματα διαφέρουν ακριβώς στον τρόπο με τον οποίο γίνεται αυτό. Στο σημείο αυτό, θα κάνουμε μόνον μερικά σχόλια. Πρώτον, ως μέρος της προηγούμενης ακολουθίας, ο κατάλογος του κόμβου 1 διατηρεί μία εγγραφή η οποία δείχνει ότι κάποια απομακρυσμένη κρυφή μνήμη διαθέτει ένα αντίγραφο της γραμμής η οποία περιέχει τη θέση 798. Επομένως, πρέπει να υπάρχει ένα πρωτόκολλο το οποίο θα αναλάβει να διαχειριστεί τις αλλαγές. Για παράδειγμα, αν συμβεί μία αλλαγή στην κρυφή μνήμη, αυτό το γεγονός είναι δυνατόν να μεταδοθεί προς τους άλλους κόμβους. Στη συνέχεια, ο κατάλογος κάθε κόμβου ο οποίος λαμβάνει αυτή την εκπομπή μπορεί να προσδιορίσει αν κάποια τοπική κρυφή μνήμη διαθέτει αυτή τη γραμμή, και αν αυτό ισχύει, να προκαλέσει τον καθαρισμό της. Αν η πραγματική θέση μνήμης βρίσκεται στον κόμβο ο οποίος λαμβάνει την εκπομπή μεταβολής, τότε ο κατάλογος αυτού του κόμβου θα πρέπει να διατηρεί μία καταχώρηση η οποία δείχνει ότι αυτή η γραμμή της μνήμης δεν είναι έγκυρη και παραμένει μη έγκυρη έως ότου λάβει χώρα μία λειτουργία εγγραφής προς τα πίσω. Αν ένας άλλος επεξεργαστής (τοπικός ή απομακρυσμένος) ζητήσει τη μη έγκυρη γραμμή, τότε ο τοπικός κατάλογος θα πρέπει να επιβάλλει μία εγγραφή προς τα πίσω, ώστε να ενημερωθεί η μνήμη πριν παραδώσει τα δεδομένα.

Τα Θετικά και Τα Αρνητικά της Μη Ομοιόμορφης Προσπέλασης Μνήμης

Το βασικό πλεονέκτημα ενός συστήματος CC-NUMA είναι ότι έχει τη δυνατότητα να δίνει καλή απόδοση σε υψηλότερα επίπεδα παραλληλισμού σε σχέση με το σύστημα συμμετρικών πολυεπεξεργαστών, χωρίς να απαιτεί σημαντικές αλλαγές σε επίπεδο λογισμικού. Με πολλούς κόμβους NUMA, η κυκλοφορία του διαύλου για κάθε κόμβο, περιορίζεται στις απαιτήσεις τις οποίες ο δίαυλος είναι σε θέση να διαχειριστεί. Ωστόσο, αν πολλές από τις προσπελάσεις μνήμης αφορούν απομακρυσμένους κόμβους, η απόδοση αρχίζει να μειώνεται. Υπάρχει λόγος να πιστέψουμε ότι αυτή η μείωση είναι δυνατόν να αποφευχθεί. Πρώτον, η χρήση των κρυφών μνημών Επιπέδου 1 και 2 είναι σχεδιασμένη ώστε να ελαχιστοποιεί όλες τις προσπελάσεις στη μνήμη, συμπεριλαμβανομένων και των απομακρυσμένων. Αν ένα μεγάλο μέρος του λογισμικού διαθέτει ικανοποιητική χρονική τοπικότητα, τότε οι απομακρυσμένες προσπελάσεις μνήμης δεν θα πρέπει να είναι υπερβολικές. Δεύτερον, αν το λογισμικό έχει ικανοποιητική χωρική τοπικότητα και αν χρησιμοποιείται η ιδεατή μνήμη, τότε τα δεδομένα τα οποία χρειάζονται σε μία εφαρμογή θα βρίσκονται τοποθετημένα σε ένα περιορισμένο πλήθος πρόσφατα χρησιμοποιούμενων σελίδων, οι οποίες είναι δυνατόν να φορτωθούν αρχικά στην τοπική μνήμη της

τρέχουσας εφαρμογής. Οι σχεδιαστές της μηχανής Sequent αναφέρουν ότι αυτή η χωρική τοπικότητα δεν παρουσιάζεται σε αντιπροσωπευτικές εφαρμογές [LOVE96]. Τέλος, η τεχνική της ιδεατής μνήμης είναι δυνατόν να ενισχυθεί συμπεριλαμβάνοντας στο λειτουργικό σύστημα ένα μηχανισμό μετακόμισης σελίδων, ο οποίος θα μετακινεί μία σελίδα της ιδεατής μνήμης σε έναν κόμβο ο οποίος τη χρησιμοποιεί συχνά. Οι σχεδιαστές της μηχανής Silicon Graphics αναφέρουν την επιτυχία αυτής της τεχνικής [WHIT97].

Ακόμη και αν αντιμετωπιστεί η μείωση της απόδοσης η οποία ωφείλεται σε απομακρυσμένες προσπελάσεις, τα συστήματα CC-NUMA εμφανίζουν δύο ακόμη μειονεκτήματα. Αυτά τα δύο συγκεκριμένα, περιγράφονται αναλυτικά στην αναφορά [PFIS98]. Πρώτο, ένα σύστημα CC-NUMA δεν μοιάζει ξεκάθαρα με ένα σύστημα συμμετρικών πολυεπεξεργαστών, αλλά απαιτούνται αλλαγές στο λογισμικό, ώστε να μεταφερθεί το λειτουργικό σύστημα και οι εφαρμογές ενός συστήματος συμμετρικών πολυεπεξεργαστών σε ένα σύστημα CC-NUMA. Αυτές οι αλλαγές συμπεριλαμβάνουν την κατανομή των σελίδων η οποία ήδη αναφέρθηκε, την κατανομή των διεργασιών, και την ισορροπία του φορτίου από το λειτουργικό σύστημα. Ένα δεύτερο ζήτημα, είναι εκείνο της διαθεσιμότητας. Πρόκειται για ένα αρκετά πολύπλοκο ζήτημα και εξαρτάται από την ακριβή υλοποίηση του συστήματος CC-NUMA. Ένας αναγνώστης ο οποίος ενδιαφέρεται, μπορεί να ανατρέξει στην αναφορά [PFIS98].

Προσομοιωτής επεξεργαστή διανυσμάτων

17.7 ΥΠΟΛΟΓΙΣΜΟΙ ΔΙΑΝΥΣΜΑΤΩΝ

Παρά το γεγονός ότι η απόδοση των μεγάλων υπολογιστικών συστημάτων γενικού σκοπού συνεχίζει να βελτιώνεται με αμείωτο ρυθμό, συνεχίζουν να υπάρχουν εφαρμογές οι οποίες βρίσκονται πέραν των δυνατοτήτων ενός σύγχρονου μεγάλου υπολογιστικού συστήματος. Υπάρχει η ανάγκη ύπαρξης υπολογιστών οι οποίοι να επιλύουν μαθηματικά προβλήματα φυσικών διεργασιών, όπως προβλήματα αεροδυναμικής, σεισμολογίας, ατομικής και πυρηνικής φυσικής, και φυσικής πλάσματος.

Τυπικά, αυτά τα προβλήματα χαρακτηρίζονται από την ανάγκη ύπαρξης υψηλής ακρίβειας και ενός προγράμματος το οποίο εκτελεί επαναληπτικά αριθμητικές πράξεις κινητής υποδιαστικής σε μεγάλους πίνακες αριθμών. Τα περισσότερα από αυτά τα προβλήματα ανήκουν στην κατηγορία η οποία είναι γνωστή ως *προσομοίωση συνεχούς πεδίου*. Ουσιαστικά, μία φυσική κατάσταση είναι δυνατόν να περιγραφεί από μία επιφάνεια ή περιοχή τριών διαστάσεων (π.χ. η ροή του αέρα στην επιφάνεια ενός πυραύλου). Αυτή η επιφάνεια προσεγγίζεται από ένα πλέγμα σημείων. Ένα σύνολο από διαφορικές εξισώσεις ορίζει τη φυσική συμπεριφορά της επιφάνειας σε κάθε σημείο. Οι εξισώσεις αναπαρίστανται ως πίνακας τιμών και συντελεστών και η λύση χρησιμοποιεί επαναλαμβανόμενες αριθμητικές πράξεις πάνω στους πίνακες των δεδομένων.

Οι υπερυπολογιστές αναπτύχθηκαν με σκοπό να διαχειρίζονται τα προβλήματα αυτής της μορφής. Τυπικά, αυτές οι μηχανές είναι σε θέση να εκτελούν δισεκατομμύρια πράξεις κινητής υποδιαστολής ανά δευτερόλεπτο. Σε αντίθεση με τα μεγάλα υπολογιστικά συστήματα, τα οποία είναι σχεδιασμένα για πολυπρογραμματισμό και μεγάλου όγκου λειτουργίες Ε/Ε, ο υπερυπολογιστής βελτιστοποιείται με σκοπό να εκτελεί τους αριθμητικούς υπολογισμούς οι οποίοι μόλις περιγράφηκαν.

Η χρήση του υπερυπολογιστή είναι περιορισμένη και λόγω της υψηλής τιμής, η αγορά τους είναι επίσης περιορισμένη. Συγκριτικά, λίγες μόνον μηχανές είναι λειτουργικές, και συναντώνται κυρίως σε ερευνητικά κέντρα και σε μερικές κυβερνητικές οργανώσεις οι οποίες εκτελούν επιστημονικές ή

$$\begin{bmatrix} 1.5 \\ 7.1 \\ 6.9 \\ 100.5 \\ 0 \\ 59.7 \end{bmatrix} + \begin{bmatrix} 2.0 \\ 39.7 \\ 1000.003 \\ 11 \\ 21.1 \\ 19.7 \end{bmatrix} = \begin{bmatrix} 3.5 \\ 46.8 \\ 1006.093 \\ 111.5 \\ 21.1 \\ 79.4 \end{bmatrix}$$

$$A + B = C$$

Σχήμα 17.14: Παράδειγμα πρόσθεσης διανυσμάτων

μηχανολογικές λειτουργίες. Όπως συμβαίνει και σε άλλες περιοχές της τεχνολογίας των υπολογιστών, υπάρχει μία σταθερή απαίτηση της αύξησης της απόδοσης του υπερυπολογιστή. Επομένως, η τεχνολογία και η απόδοσή του συνεχίζει να εξελίσσεται.

Υπάρχει ένας ακόμη τύπος συστήματος το οποίο έχει σχεδιαστεί για να καλύπτει τις ανάγκες διανυσματικών υπολογισμών, με την ονομασία *επεξεργαστής πινάκων* (Array Processor). Παρά το γεγονός ότι ένας υπερυπολογιστής βελτιστοποιείται για να εκτελεί υπολογισμούς διανυσμάτων, πρόκειται για έναν υπολογιστή γενικού σκοπού, με δυνατότητα διαχείρισης της επεξεργασίας βαθμωτών δεδομένων και γενικών εργασιών επεξεργασίας δεδομένων. Οι επεξεργαστές πινάκων δεν συμπεριλαμβάνουν την επεξεργασία βαθμωτών δεδομένων. Αντίθετα, είναι διαμορφωμένοι ως περιφερειακές μονάδες, τόσο από τους χρήστες των μεγάλων υπολογιστικών συστημάτων, όσο και από τους χρήστες των μινι-υπολογιστών, ώστε να εκτελούν τμήματα των προγραμμάτων, τα οποία έχουν αναπαρασταθεί σε μορφή διανυσμάτων.

Προσεγγίσεις Σχετικά με τους Υπολογισμούς Διανυσμάτων

Το κλειδί όσον αφορά τη σχεδίαση ενός υπερυπολογιστή ή ενός επεξεργαστή πινάκων, είναι να αναγνωριστεί ότι η βασική του λειτουργία είναι να εκτελεί αριθμητικές πράξεις σε πίνακες ή διανύσματα αριθμών κινητής υποδιαστολής. Σε έναν υπολογιστή γενικού σκοπού, αυτό απαιτεί επαναλήψεις μέσα από κάθε στοιχείο του πίνακα. Για παράδειγμα, θεωρείστε δύο διανύσματα (μονοδιάστατοι πίνακες) των αριθμών A και B. Επιθυμούμε να προσθέσουμε αυτά τα διανύσματα και να αποθηκεύσουμε το αποτέλεσμα στο διάνυσμα C. Στο παράδειγμα του Σχήματος 17.14, η πράξη αυτή απαιτεί έξι ξεχωριστές προσθέσεις. Με ποιόν τρόπο θα μπορούσαμε να αυξήσουμε την ταχύτητα των υπολογισμών; Η απάντηση είναι να εισάγουμε κάποια μορφή παραλληλισμού.

Διάφορες προσεγγίσεις έχουν ακολουθηθεί σχετικά με την υλοποίηση παραλληλοποιημένων διανυσματικών υπολογισμών. Αυτό θα δειχθεί με ένα παράδειγμα. Θεωρείστε τον πολλαπλασιασμό διανυσμάτων $C = A \times B$, όπου A, B και C είναι πίνακες διαστάσεων $N \times N$. Η σχέση που διέπει τον υπολογισμό κάθε στοιχείου του πίνακα C είναι:

$$c_{i,j} = \sum_{k=1}^{N} a_{i,k} \times b_{k,j}$$

όπου οι πίνακες A, B και C έχουν τα στοιχεία $a_{i,j}, b_{i,j}$, και $c_{i,j}$ αντίστοιχα. Το Σχήμα 17.15α παρουσιάζει ένα πρόγραμμα γραμμένο σε γλώσσα FORTRAN, με το οποίο υλοποιείται ο παραπάνω υπολογισμός. Το πρόγραμμα αυτό είναι δυνατόν να εκτελεστεί σε ένα κοινό βαθμωτό επεξεργαστή.

Μία προσέγγιση σχετικά με τη βελτίωση της απόδοσης αναφέρεται και ως *επεξεργασία διανυσμάτων* (Vector Processing). Αυτή η προσέγγιση, υποθέτει ότι είναι δυνατή η εκτέλεση πράξεων σε ένα μονοδιάστατο διάνυσμα δεδομένων. Το Σχήμα 17.15β παρουσιάζει ένα πρόγραμμα γραμμένο σε

```
      DO 100 I = 1, N
      DO 100 J = 1, N
      C(I, J) 5 0.0
      DO 100 K = 1, N
      C(I, J) = C(I, J) + A(I, K) + B(K, J)
100   CONTINUE
```

(α) Κλιμακωτή επεξεργασία

```
      DO 100 I = 1, N
      C(I, J) = 0.0 (J = 1, N)
      DO 100 K = 1, N
      C(I, J) 5 C(I, J) + A(I, K) + B(K, J) (J = 1, N)
100   CONTINUE
```

(β) Διανυσματική επεξεργασία

```
      DO 50 J = 1, N - 1
      FORK 100
50    CONTINUE
      J = N
100   DO 200 I = 1, N
      C(I, J) = 0.0
      DO 200 K = 1, N
      C(I, J) = C(I, J) + A(I, K) + B(K, J)
200   CONTINUE
```

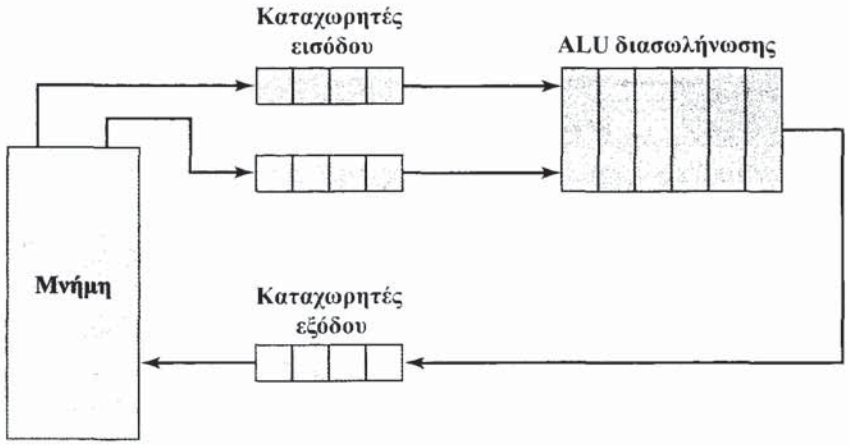
(γ) Παράλληλη επεξεργασία

Σχήμα 17.15: Πολλαπλασιασμός πινάκων (C=A × B)

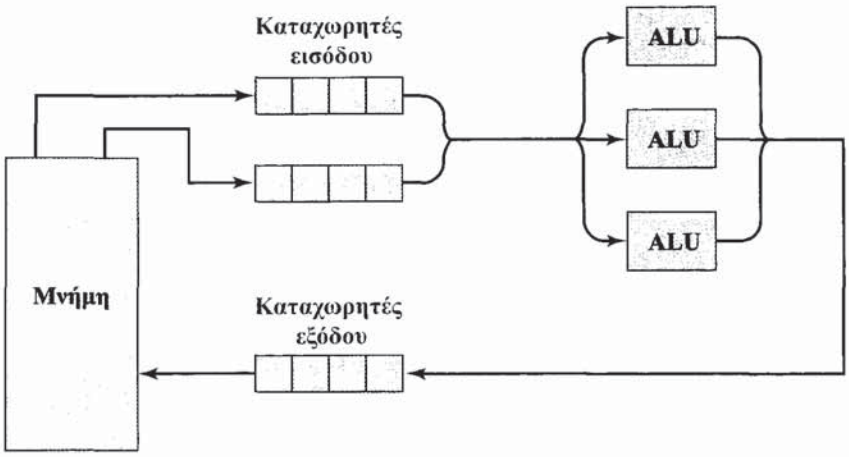
γλώσσα FORTRAN, με μία νέα μορφή εντολής, η οποία επιτρέπει τον προσδιορισμό του διανυσματικού υπολογισμού. Ο συμβολισμός J=1,N δείχνει ότι οι πράξεις πάνω σε όλους τους δείκτες J μέσα στο δοθέν διάστημα, θα εκτελεστούν ως μία. Ο τρόπος με τον οποίο θα γίνει αυτό, θα παρουσιαστεί σύντομα.

Το πρόγραμμα που παρουσιάζεται στο Σχήμα 17.15β δείχνει ότι όλα τα στοιχεία της i-οστής γραμμής πρόκειται να υπολογιστούν παράλληλα. Κάθε στοιχείο της γραμμής είναι ένα άθροισμα και τα αθροίσματα (καθώς αυξάνεται το K) εκτελούνται σειριακά και όχι παράλληλα. Ακόμη και κάτω από αυτές τις συνθήκες, είναι απαραίτητοι μόνον N^2 πολλαπλασιασμοί διανυσμάτων σε αυτό τον αλγόριθμο, σε σύγκριση με τους N^3 βαθμωτούς πολλαπλασιασμούς του βαθμωτού αλγορίθμου.

Μία άλλη προσέγγιση, η *παράλληλη επεξεργασία* (Parallel Processing), απεικονίζεται στο Σχήμα 17.15γ. Αυτή η προσέγγιση, υποθέτει την ύπαρξη N ανεξάρτητων επεξεργαστών, οι οποίοι είναι δυνατόν να λειτουργήσουν παράλληλα. Για να χρησιμοποιηθούν αποτελεσματικά αυτοί οι επεξεργαστές, θα πρέπει να μοιράσουμε το υπολογιστικό φορτίο σε αυτούς. Δύο στοιχειώδεις εντολές χρησιμοποιούνται. Η εντολή FORK n αναγκάζει μία ανεξάρτητη διεργασία να ξεκινήσει από τη θέση n. Στο μεταξύ, η αρχική διεργασία συνεχίζει την εκτέλεσή της από την εντολή η οποία ακολουθεί αμέσως μετά τη FORK. Κάθε εκτέλεση μίας εντολής FORK παράγει μία νέα διεργασία. Η εντολή JOIN αποτελεί ουσιαστικά την αντίστροφη της FORK. Η πρόταση JOIN N αναγκάζει N ανεξάρτητες διεργασίες να συγχωνευτούν σε μία, η οποία συνεχίζει την εκτέλεση από την εντολή η οποία ακολουθεί την JOIN. Το λειτουργικό σύστημα θα πρέπει να συντονίζει αυτές τις συγχωνεύσεις. Επομένως, η εκτέλεση δεν συνεχίζεται έως ότου όλες οι N σε πλήθος διεργασίες έχουν φτάσει στην εντολή JOIN.



(α) ALU διασωλήνωσης



(β) Παράλληλες αριθμητικές λογικές μονάδες (ALUs)

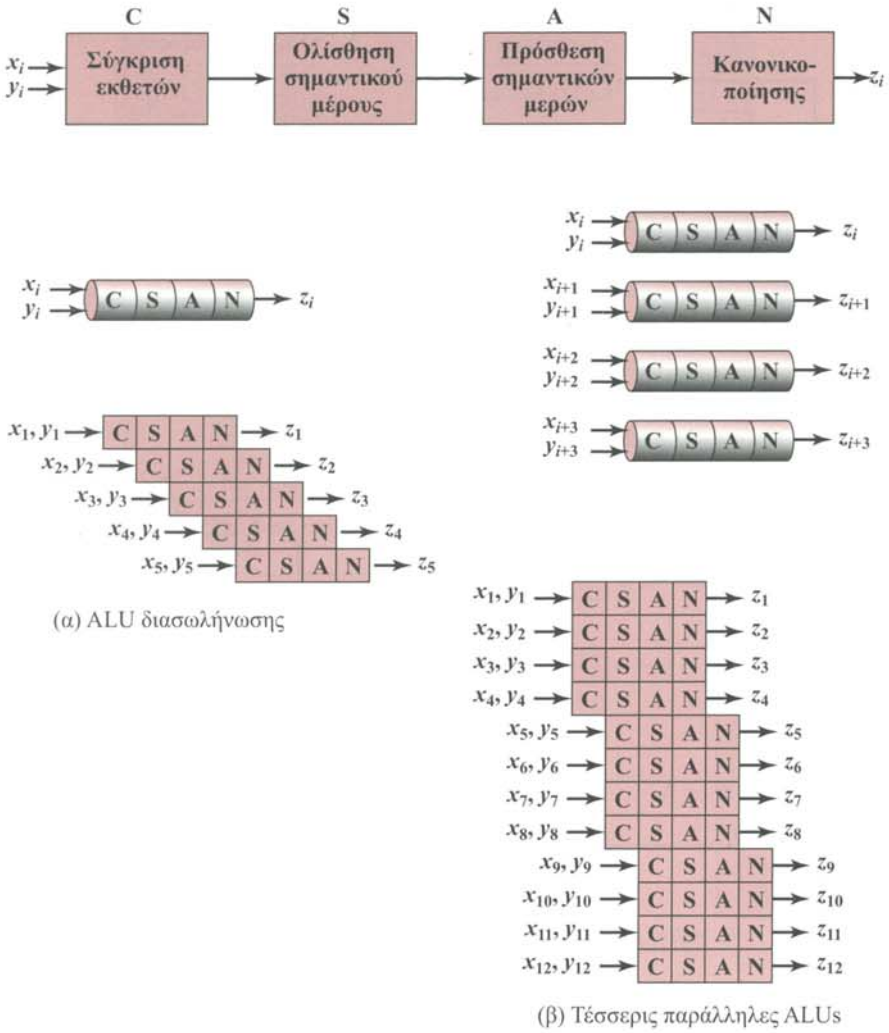
Σχήμα 17.16: Προσεγγίσεις όσον αφορά τον πολλαπλασιασμό των διανυσμάτων

Το πρόγραμμα του Σχήματος 17.15γ είναι γραμμένο ώστε να μιμείται τη συμπεριφορά του προγράμματος επεξεργασίας διανυσμάτων. Σε ένα πρόγραμμα παράλληλης επεξεργασίας, κάθε στήλη του C υπολογίζεται από μία διαφορετική διεργασία. Επομένως, τα στοιχεία μίας γραμμής του C υπολογίζονται παράλληλα.

Η παραπάνω συζήτηση περιγράφει προσεγγίσεις σχετικά με τον υπολογισμό διανυσμάτων, με όρους λογικής και αρχιτεκτονικής. Στο σημείο αυτό, ας στραφούμε στην εξέταση των μορφών οργάνωσης των επεξεργαστών, οι οποίες είναι δυνατόν να χρησιμοποιηθούν προκειμένου να υλοποιηθούν αυτές οι προσεγγίσεις. Έχει αναζητηθεί και εξακολουθεί να αναζητείται, μία μεγάλη ποικιλία από μορφές οργάνωσης. Οι βασικές κατηγορίες είναι οι ακόλουθες:

- Αριθμητική και λογική μονάδα με διασωλήνωση
- Παράλληλες αριθμητικές και λογικές μονάδες
- Παράλληλοι επεξεργαστές

Το Σχήμα 17.16 απεικονίζει τις πρώτες δύο προσεγγίσεις. Έχουμε ήδη εξετάσει τη διασωλήνωση στο Κεφάλαιο 12. Στο σημείο αυτό, η έννοια επεκτείνεται στη λειτουργία της αριθμητικής και λογικής μονάδας. Επειδή οι πράξεις με αριθμούς κινητής υποδιαστολής είναι αρκετά πολύπλοκες, υπάρχει



Σχήμα 17.17: Επεξεργασία πράξεων κινητής υποδιαστολής με χρήση διασωλήνωσης

δυνατότητα διάσπασής τους σε στάδια, έτσι ώστε τα διαφορετικά στάδια να είναι δυνατόν να λειτουργούν ταυτόχρονα πάνω σε διαφορετικά σύνολα δεδομένων. Αυτό απεικονίζεται στο Σχήμα 17.17α. Η πρόσθεση αριθμών κινητής υποδιαστολής διασπάται σε 4 στάδια (δείτε το Σχήμα 9.22). σύγκριση, ολίσθηση, πρόσθεση, και κανονικοποίηση. Ένα διάνυσμα αριθμών παρουσιάζεται ακολουθιακά στο πρώτο στάδιο. Καθώς προχωρά η επεξεργασία, θα εκτελούνται ταυτόχρονες πράξεις πάνω σε 4 σύνολα αριθμών, μέσα στη διασωλήνωση.

Θα πρέπει να γίνει σαφές ότι αυτή η οργάνωση είναι κατάλληλη για την επεξεργασία διανυσμάτων. Για να διαπιστώσουμε αυτό το γεγονός, θεωρείστε τη διασωλήνωση εντολών η οποία περιγράφηκε στο Κεφάλαιο 12. Ο επεξεργαστής περνά μέσα από έναν επαναλαμβανόμενο κύκλο ανάκλησης και επεξεργασίας εντολών. Αν δεν υπάρχουν διακλαδώσεις, ο επεξεργαστής προσκομίζει διαρκώς εντολές από θέσεις οι οποίες βρίσκονται η μία μετά την άλλη. Επομένως, η διασωλήνωση διατηρείται πλήρης και επιτυγχάνεται εξοικονόμηση χρόνου. Ομοίως, η αριθμητική και λογική μονάδα με διασωλήνωση, θα εξοικονομεί χρόνο μόνον αν τροφοδοτείται με μία ροή δεδομένων από θέσεις οι οποίες βρίσκονται η μία μετά την άλλη. Μία απλή, απομονωμένη πράξη με αριθμούς κινητής υποδιαστολής δεν εκτελείται πιο γρήγορα μέσα από τη διασωλήνωση. Η αύξηση της ταχύτητας επιτυγχάνεται όταν ένα διάνυσμα παραγόντων παρουσιαστεί στην αριθμητική και λογική μονάδα. Η μονάδα ελέγχου μετακινεί διαδο-

χικά τα δεδομένα εντός της αριθμητικής και λογικής μονάδας, έως ότου ολοκληρωθεί η επεξεργασία ολόκληρου του διανύσματος.

Η λειτουργία της διασωλήνωσης είναι δυνατόν να βελτιωθεί περαιτέρω, αν τα στοιχεία του διανύσματος είναι διαθέσιμα σε καταχωρητές αντί της κύριας μνήμης. Αυτό προτείνεται στο Σχήμα 17.16α. Τα στοιχεία κάθε παράγοντα του διανύσματος φορτώνονται ως μπλοκ σε έναν καταχωρητή διανυσμάτων, ο οποίος είναι ουσιαστικά μία μεγάλη ομάδα από παρόμοιους καταχωρητές. Το αποτέλεσμα επίσης τοποθετείται στον καταχωρητή διανυσμάτων. Επομένως, οι περισσότερες πράξεις εμπλέκουν μόνον τη χρήση καταχωρητών και οι μοναδικές λειτουργίες οι οποίες απαιτούν προσπέλαση της μνήμης είναι οι λειτουργίες *φόρτωσης* και *αποθήκευσης στην αρχή και στο τέλος μίας πράξης* μεταξύ διανυσμάτων.

Ο μηχανισμός ο οποίος απεικονίζεται στο Σχήμα 17.17 θα μπορούσε να αναφερθεί και ως *διασωλήνωση μέσα σε μία πράξη*. Με άλλα λόγια, έχουμε μία απλή αριθμητική πράξη (π.χ. $C = A + B$), η οποία πρόκειται να εφαρμοστεί σε παράγοντες διανυσμάτων και η διασωλήνωση επιτρέπει την παράλληλη επεξεργασία πολλών στοιχείων του διανύσματος. Αυτός ο μηχανισμός είναι δυνατόν να ενισχυθεί με τη *διασωλήνωση κατά μήκος των πράξεων*. Στην τελευταία περίπτωση, υπάρχει μία ακολουθία από αριθμητικές πράξεις διανυσμάτων και η διασωλήνωση χρησιμοποιείται για να αυξηθεί η ταχύτητα της επεξεργασίας. Μία προσέγγιση αυτής της μορφής ονομάζεται **αλυσίδωση** και συναντάται στους υπολογιστές Cray. Ο βασικός κανόνας της αλυσίδωσης είναι ο εξής: Μία διανυσματική πράξη ξεκινά αμέσως μόλις είναι διαθέσιμο το πρώτο στοιχείο του παράγοντα διανύσματος (διανυσμάτων) και απελευθερωθεί η λειτουργική μονάδα (π.χ πρόσθεσης, αφαίρεσης, πολλαπλασιασμού, διαίρεσης). Ουσιαστικά, η αλυσίδωση αναγκάζει τα αποτελέσματα τα οποία εκδίδονται από μία λειτουργική μονάδα να τροφοδοτήσουν αμέσως την επόμενη, κ.ο.κ. Αν χρησιμοποιηθούν καταχωρητές διανυσμάτων, τα ενδιάμεσα αποτελέσματα δεν είναι αναγκαίο να αποθηκευτούν στη μνήμη και είναι δυνατόν να χρησιμοποιηθούν πριν ολοκληρωθεί η διανυσματική πράξη η οποία τα παρήγαγε.

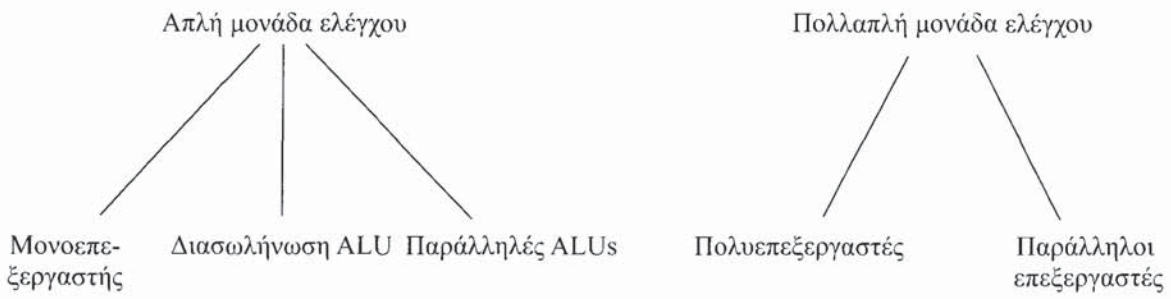
Για παράδειγμα, όταν εκτελείται ο υπολογισμός $C = (s \times A) + B$, όπου A, B και C είναι διανύσματα και s είναι μία βαθμωτή μεταβλητή, ο υπολογιστής Cray πιθανόν να εκτελέσει τρεις εντολές ταυτόχρονα. Τα στοιχεία που προσκομίζονται για φόρτωση, εισέρχονται αμέσως στον πολλαπλασιαστή της διασωλήνωσης, τα γινόμενα στέλνονται στον αθροιστή της διασωλήνωσης, και τα αθροίσματα τοποθετούνται σε έναν καταχωρητή διανύσματος, αμέσως μόλις παραχθούν από τον αθροιστή:

1. Φόρτωση διανύσματος	A→ Καταχωρητής διανύσματος (VR1)
2. Φόρτωση διανύσματος	B→ VR2
3. Πολλαπλασιασμός διανύσματος	s × VR1 → VR3
4. Πρόσθεση διανύσματος	VR3 + VR2 → VR4
5. Αποθήκευση διανύσματος	VR4→ C

Οι εντολές 2 και 3 είναι δυνατόν να αλυσιδωθούν (διασωληνωθούν) επειδή χρησιμοποιούν διαφορετικές θέσεις μνήμης και καταχωρητές. Η εντολή 4 θα πρέπει να περιμένει τα αποτελέσματα των εντολών 2 και 3, αλλά είναι δυνατόν να αλυσιδωθεί με αυτές. Μόλις είναι διαθέσιμα τα πρώτα στοιχεία των καταχωρητών διανύσματος 2 και 3, είναι δυνατή η εκκίνηση της πράξης της εντολής 4.

Ένας άλλος τρόπος υλοποίησης της επεξεργασίας διανυσμάτων είναι μέσω της χρήσης πολλαπλών αριθμητικών και λογικών μονάδων σε έναν επεξεργαστή, υπό τον έλεγχο της μονάδας ελέγχου. Σε αυτή την περίπτωση, η μονάδα ελέγχου δρομολογεί τα δεδομένα προς τις αριθμητικές και λογικές μονάδες, ώστε αυτές να λειτουργούν παράλληλα. Επίσης, είναι εφικτό να χρησιμοποιηθεί η διασωλήνωση για κάθε παράλληλη αριθμητική και λογική μονάδα. Αυτό απεικονίζεται στο Σχήμα 17.17β. Το παράδειγμα δείχνει μία περίπτωση παράλληλης λειτουργίας τεσσάρων αριθμητικών και λογικών μονάδων.

Όπως συμβαίνει και με την οργάνωση της διασωλήνωσης, μία μορφή παράλληλης οργάνωσης των αριθμητικών και λογικών μονάδων είναι κατάλληλη για την επεξεργασία διανυσμάτων. Η μονάδα ελέγχου δρομολογεί διαδοχικά τα στοιχεία των διανυσμάτων στις αριθμητικές και λογικές μονάδες,



Σχήμα 17.18: Ταξινόμηση των μορφών οργάνωσης του υπολογιστή

έως ότου ολοκληρωθεί η επεξεργασία όλων των στοιχείων.

Τέλος, η επεξεργασία διανυσμάτων υλοποιείται χρησιμοποιώντας πολλούς παράλληλους επεξεργαστές. Σε αυτή την περίπτωση, είναι αναγκαίο να διασπάσουμε την εργασία σε πολλές διεργασίες οι οποίες εκτελούνται παράλληλα. Αυτή η μορφή οργάνωσης είναι αποτελεσματική μόνον αν είναι διαθέσιμο το υλικό και το λογισμικό με το οποίο θα συντονιστούν αποτελεσματικά οι παράλληλοι επεξεργαστές.

Μπορούμε να επεκτείνουμε την ταξινόμηση της Ενότητας 17.1, ώστε να εμφανιστούν οι νέες αυτές δομές, με τον τρόπο που απεικονίζεται στο Σχήμα 17.18. Οι μορφές οργάνωσης των υπολογιστών διακρίνονται από την παρουσία μίας ή περισσότερων μονάδων ελέγχου. Οι πολλές μονάδες ελέγχου σημαίνουν την ύπαρξη πολλών επεξεργαστών. Ακολουθώντας την προηγούμενη συζήτησή μας, αν οι πολλοί επεξεργαστές έχουν τη δυνατότητα να συνεργαστούν πάνω σε κάποια εργασία, τότε ονομάζονται *παράλληλοι επεξεργαστές*.

Ο αναγνώστης θα πρέπει να γνωρίζει κάποια ατυχή ορολογία, την οποία είναι πιθανό να συναντήσει στην αρθρογραφία. Ο όρος *επεξεργαστής διανύσματος* συχνά θεωρείται ισοδύναμος με μία μορφή διασωληνωμένης οργάνωσης της αριθμητικής και λογικής μονάδας, παρά το γεγονός ότι η παράλληλη οργάνωση της αριθμητικής και λογικής μονάδας είναι επίσης σχεδιασμένη για την επεξεργασία διανυσμάτων και, όπως έχουμε συζητήσει, η οργάνωση παράλληλων επεξεργαστών επίσης είναι σχεδιασμένη για την επεξεργασία διανυσμάτων. Ο όρος *επεξεργασία πινάκων* χρησιμοποιείται συχνά για να αναφερθεί σε μία παράλληλη αριθμητική και λογική μονάδα, παρά το γεγονός ότι, ξανά, οποιαδήποτε από τις τρεις μορφές οργάνωσης βελτιστοποιείται προκειμένου να επεξεργάζεται πίνακες. Ακόμη χειρότερα, ο όρος *επεξεργαστής πινάκων*, συνήθως αναφέρεται σε ένα βοηθητικό επεξεργαστή, ο οποίος συνδέεται με έναν επεξεργαστή γενικού σκοπού και χρησιμοποιείται για να εκτελεί υπολογισμούς διανυσμάτων. Ένας επεξεργαστής πινάκων έχει τη δυνατότητα να χρησιμοποιήσει είτε την προσέγγιση της διασωλήνωσης, είτε την προσέγγιση της παράλληλης αριθμητικής και λογικής μονάδας.

Σήμερα, η οργάνωση της αριθμητικής και λογικής μονάδας με διασωλήνωση, είναι εκείνη η οποία κυριαρχεί στην αγορά. Τα συστήματα που χρησιμοποιούν διασωλήνωση είναι λιγότερο πολύπλοκα σε σχέση με εκείνα τα οποία χρησιμοποιούν τις δύο άλλες προσεγγίσεις. Η μονάδα ελέγχου αυτών των συστημάτων, καθώς και η σχεδίαση του λειτουργικού συστήματος έχουν αναπτυχθεί κατά τέτοιο τρόπο, ώστε να επιτυγχάνουν αποτελεσματική κατανομή των πόρων και υψηλή απόδοση. Το υπόλοιπο αυτής της ενότητας είναι αφιερωμένο σε μία πιο λεπτομερή εξέταση αυτής της προσέγγισης, με χρήση ενός συγκεκριμένου παραδείγματος.

Ο Μηχανισμός Επεξεργασίας Διανυσμάτων της Μηχανής IBM 3090

Ένα καλό παράδειγμα μορφής οργάνωσης της αριθμητικής και λογικής μονάδας με διασωλήνωση, ώστε να υλοποιείται η επεξεργασία των διανυσμάτων, είναι ο μηχανισμός επεξεργασίας διανυσμάτων ο οποίος υλοποιήθηκε για την αρχιτεκτονική IBM 370 και υλοποιήθηκε στην πιο σύγχρονη σειρά 3090

[PADE88, TUCK87]. Αυτός ο μηχανισμός είναι μία προαιρετική προσθήκη στο βασικό σύστημα, αλλά ενσωματώνεται σε μεγάλο βαθμό σε αυτό. Ο μηχανισμός μοιάζει με αντίστοιχους τους οποίους συναντούμε σε υπερυπολογιστές όπως είναι η οικογένεια Cray.

Η οικογένεια της IBM χρησιμοποιεί ένα πλήθος από καταχωρητές διανύσματος. Κάθε καταχωρητής είναι στην πραγματικότητα μία ομάδα από βαθμωτούς καταχωρητές. Για να υπολογιστεί το διανυσματικό άθροισμα $C = A + B$, τα διανύσματα A κα B φορτώνονται σε δύο καταχωρητές διανύσματος. Τα δεδομένα από αυτούς τους καταχωρητές περνούν από την αριθμητική και λογική μονάδα με όσο το δυνατόν μεγαλύτερη ταχύτητα, και τα αποτελέσματα καταχωρούνται σε ένα τρίτο καταχωρητή. Η επικάλυψη των υπολογισμών και η φόρτωση των δεδομένων εισόδου σε καταχωρητές με τη μορφή μπλοκ, έχει ως αποτέλεσμα τη σημαντική αύξηση της ταχύτητας σε σύγκριση με τη συνηθισμένη λειτουργία της αριθμητικής και λογικής μονάδας.

ΟΡΓΑΝΩΣΗ Η αρχιτεκτονική διανυσμάτων της IBM, αλλά και παρόμοιες αριθμητικές και λογικές μονάδες που χρησιμοποιούν διασωλήνωση, παρέχουν αυξημένη απόδοση όταν εμφανίζονται βρόχοι εντολών που αφορούν βαθμωτά αριθμητικά δεδομένα. Αυτό συμβαίνει με τρεις τρόπους:

- Η σταθερή και προκαθορισμένη δομή των διανυσματικών δεδομένων επιτρέπει την αντικατάσταση των συνηθισμένων εντολών ενός βρόχου από ταχύτερες εσωτερικές λειτουργίες μηχανής (σε επίπεδο υλικού ή μικροκώδικα).

- Η προσπέλαση των δεδομένων και οι αριθμητικές πράξεις πάνω σε διάφορα διαδοχικά στοιχεία των διανυσμάτων, είναι δυνατόν να προχωρήσουν ταυτόχρονα αν επικαλυφθούν μέσα από μία διασωλήνωση ή αν εκτελεστούν παράλληλες λειτουργίες πάνω σε πολλά στοιχεία.

- Η χρήση των καταχωρητών διανύσματος για την αποθήκευση των ενδιάμεσων αποτελεσμάτων αποτρέπει τις επιπλέον αναφορές στη μνήμη.

Το Σχήμα 17.19 απεικονίζει τη γενική οργάνωση του μηχανισμού επεξεργασίας διανυσμάτων. Παρά το γεγονός ότι ο μηχανισμός φαίνεται να είναι μία φυσικά διαχωρισμένη προσθήκη στον επεξεργαστή, η αρχιτεκτονική του αποτελεί επέκταση της αρχιτεκτονικής S/370, με τους ακόλουθους τρόπους:

- Οι υπάρχουσες εντολές της αρχιτεκτονικής S/370 χρησιμοποιούνται για όλες τις πράξεις μεταξύ βαθμωτών δεδομένων.

- Οι αριθμητικές πράξεις πάνω σε ξεχωριστά διανυσματικά στοιχεία παράγουν ακριβώς το ίδιο αποτέλεσμα με τις αντίστοιχες εντολές του συστήματος S/370. Για παράδειγμα, μία σχεδιαστική απόφαση σχετιζόταν με τον ορισμό του αποτελέσματος μίας πράξης διαίρεσης μεταξύ αριθμών κινητής υποδιαστολής. Θα έπρεπε το αποτέλεσμα να είναι ακριβές, όπως συμβαίνει με τη διαίρεση κινητής υποδιαστολής βαθμωτών δεδομένων, ή θα έπρεπε να επιτραπεί μία προσέγγιση, η οποία θα οδηγούσε σε ταχύτερη υλοποίηση αλλά μερικές φορές θα εισήγαγε σφάλμα σε μία ή περισσότερες από τις λιγότερο σημαντικές θέσεις bit; Η απόφαση η οποία ελήφθη ήταν να υπάρξει πλήρης συμβατότητα με το σύστημα S/370, με κόστος μία μικρή μείωση της απόδοσης.

- Οι εντολές διανυσμάτων είναι δυνατόν να διακοπούν και να συνεχιστεί η εκτέλεσή τους από το σημείο διακοπής μετά από την υλοποίηση της κατάλληλης ενέργειας, με ένα τρόπο συμβατό με την τεχνική διακοπών η οποία συναντάται στο σύστημα S/370.

- Οι αριθμητικές εξαιρέσεις είναι οι ίδιες, ή επεκτάσεις των εξαιρέσεων τις οποίες συναντάμε στις εντολές βαθμωτών αριθμητικών δεδομένων της μηχανής S/370. Επίσης, είναι δυνατόν να χρησιμοποιηθούν οι ίδιες προκατασκευασμένες ρουτίνες. Για να συμβεί αυτό, χρησιμοποιείται ένας δείκτης διακοπών διανύσματος, ο οποίος δείχνει τη θέση του καταχωρητή διανύσματος ο

Σχήμα 17.19: Η μηχανή 3090 της IBM με μηχανισμό επεξεργασίας διανυσμάτων

οποίος επηρρεάζεται από μία συνθήκη εξαίρεσης (π.χ. υπερχείλιση). Επομένως, όταν ξεκινά εκ νέου η εκτέλεση της εντολής διανύσματος, προσπελαύνεται η κατάλληλη θέση ενός καταχωρητή διανύσματος.

- Τα διανυσματικά δεδομένα βρίσκονται σε μία ιδεατή μνήμη, όπου τα σφάλματα σελίδων αντιμετωπίζονται με τυπικό τρόπο.

Αυτό το επίπεδο συνένωσης παρέχει ένα πλήθος από πλεονεκτήματα. Τα υπάρχοντα λειτουργικά συστήματα έχουν τη δυνατότητα να υποστηρίξουν το μηχανισμό επεξεργασίας διανυσμάτων με μικρές επεκτάσεις. Τα υπάρχοντα προγράμματα εφαρμογών, οι μεταγλωττιστές, και άλλα λογισμικά, μπορούν να εκτελούνται χωρίς αλλαγές. Το λογισμικό το οποίο έχει τη δυνατότητα να εκμεταλλευτεί το μηχανισμό επεξεργασίας διανυσμάτων, μπορεί να τροποποιηθεί κατάλληλα.

ΚΑΤΑΧΩΡΗΤΕΣ Ένα βασικό ζήτημα της σχεδίασης ενός μηχανισμού επεξεργασίας διανυσμάτων, είναι αν οι παράγοντες θα αποθηκεύονται σε καταχωρητές ή στη μνήμη. Η οργάνωση της IBM αναφέρεται με την ονομασία *από καταχωρητή σε καταχωρητή*, επειδή οι διανυσματικοί παράγοντες, τόσο εισόδου όσο και εξόδου, είναι δυνατόν να αποθηκευτούν σε καταχωρητές διανύσματος. Αυτή η προσέγγιση μπορεί επίσης να χρησιμοποιηθεί στους υπερυπολογιστές Cray. Μία εναλλακτική προσέγγιση, η οποία χρησιμοποιείται στις μηχανές της Control Data, είναι να λαμβάνονται οι παράγοντες άμεσα από τη μνήμη. Το βασικό μειονέκτημα της χρήσης των καταχωρητών διανύσματος είναι ότι ο προγραμματιστής ή ο μεταγλωττιστής θα πρέπει να τους λάβει υπόψη προκειμένου να επιτευχεί καλή απόδοση. Για παράδειγμα, έστω ότι το μήκος των καταχωρητών διανύσματος είναι K και το μήκος των διανυσμάτων προς επεξεργασία είναι $N > K$. Σε αυτή την περίπτωση, θα πρέπει να εκτελεστεί ένας βρόχος στον οποίο η πράξη εκτελείται πάνω σε K στοιχεία κάθε φορά. Ο βρόχος αυτός εκτελείται N/K

FORTRAN ROUTINE:

```
      DO 100 J = 1, 50
      CR(J) = AR(J) * BR(J) - AI(J) * BI(J)
100   CI(J) = AR(J) * BI(J) + AI(J) * BR(J)
```

Λειτουργία	Κύκλοι
AR(J) * BR(J) → T1(J)	3
AI(J) * BI(J) → T2(J)	3
T1(J) 2 T2(J) → CR(J)	3
AR(J) * BI(J) → T3(J)	3
AI(J) * BR(J) → T4(J)	3
T3(J) 1 T4(J) → CI(J)	
TOTAL	18

(α) Από περιοχή αποθήκευσης σε περιοχή αποθήκευσης

Λειτουργία	Κύκλοι
AR(J) → V1(J)	1
BR(J) → V2(J)	1
V1(J) * V2(J) → V3(J)	1
AI(J) → V4(J)	1
BI(J) → V5(J)	1
V4(J) * V5(J) → V6(J)	1
V3(J) 2 V6(J) → V7(J)	1
V7(J) → CR(J)	1
V1(J) * V5(J) → V8(J)	1
V4(J) * V2(J) → V9(J)	1
V8(J) 1 V9(J) → V0(J)	1
V0(J) → CI(J)	
TOTAL	12

(β) Από καταχωρητή σε καταχωρητή

Λειτουργία	Κύκλοι
AR(J) → V1(J)	1
V1(J) * BR(J) → V2(J)	1
AI(J) → V3(J)	1
V3(J) * BI(J) → V4(J)	1
V2(J) 2 V4(J) → V5(J)	1
V5(J) → CR(J)	1
V1(J) * BI(J) → V6(J)	1
V4(J) * BR(J) → V7(J)	1
V6(J) 1 V7(J) → V8(J)	1
V8(J) → CI(J)	
TOTAL	10

(γ) Από περιοχή αποθήκευσης σε καταχωρητή

Λειτουργία	Κύκλοι
AR(J) → V1(J)	1
V1(J) * BR(J) → V2(J)	1
AI(J) → V3(J)	1
V2(J) 2 V3(J) * BI(J) → V2(J)	1
V2(J) → CR(J)	1
V1(J) * BI(J) → V4(J)	1
V4(J) 1 V3(J) * BR(J) → V5(J)	1
V5(J) → CI(J)	
TOTAL	8

(δ) Σύνθετη εντολή

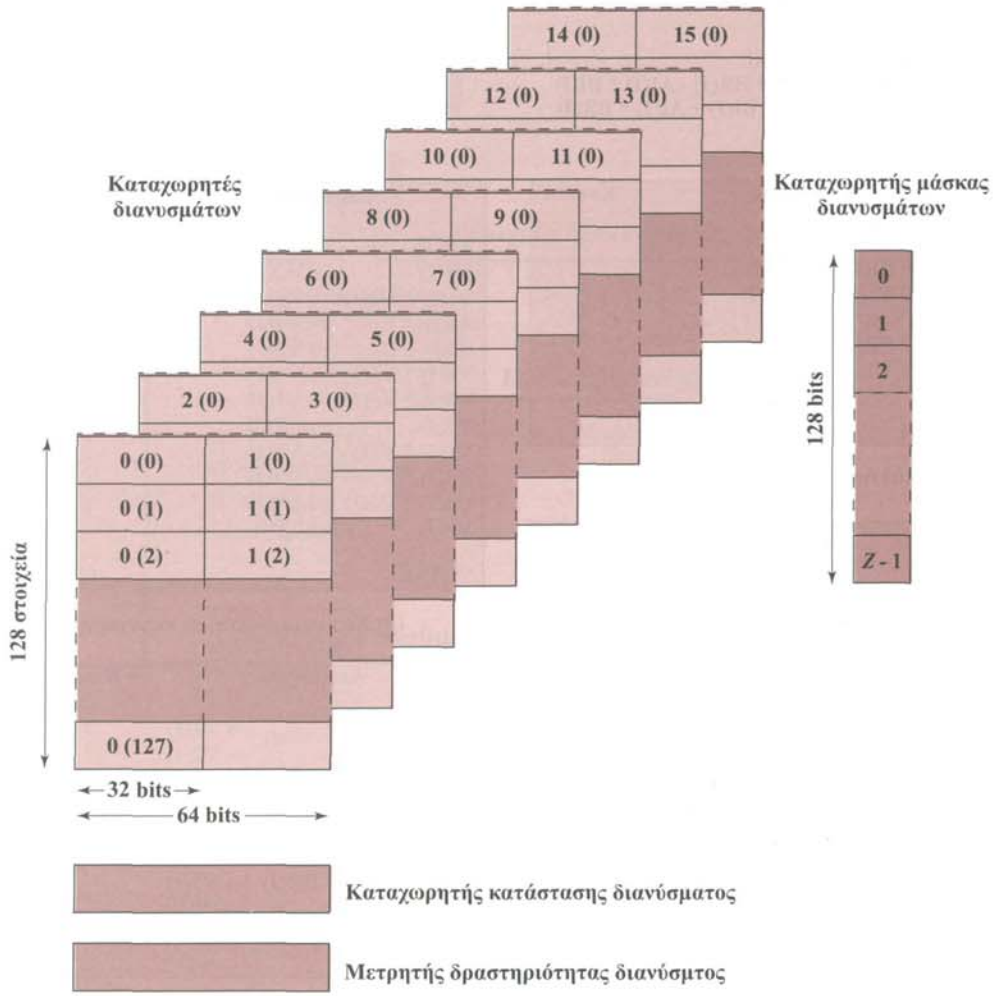
Vi = Καταχωρητές διανυσμάτων
AR, BR, AI, BI = Παράγοντες στη μνήμη
Ti = Προσωρινές θέσεις στη μνήμη

Σχήμα 17.20: Εναλλακτικά προγράμματα υπολογισμού διανυσμάτων

φορές. Το βασικό πλεονέκτημα της χρήσης καταχωρητών διανύσματος είναι ότι η πράξη αποσυνδέεται από την αργή κύρια μνήμη και εκτελείται κυρίως με τη βοήθεια των ταχύτερων καταχωρητών.

Η αύξηση της ταχύτητας η οποία είναι εφικτή όταν χρησιμοποιούνται καταχωρητές, απεικονίζεται στο Σχήμα 17.20. Η ρουτίνα η οποία είναι γραμμένη σε γλώσσα FORTRAN, πολλαπλασιάζει το διάνυσμα A με το διάνυσμα B, παράγοντας το διάνυσμα C, όπου κάθε διάνυσμα έχει ένα πραγματικό (AR, BR, CR) και ένα φανταστικό μέρος (AI, BI, CI). Ο υπολογιστής 3090 έχει τη δυνατότητα εκτέλεσης μίας προσπέλασης μνήμης ανά κύκλο επεξεργαστή ή ρολογιού (είτε για εγγραφή είτε για ανάγνωση). Επίσης, διαθέτει καταχωρητές οι οποίοι μπορούν να υποστηρίξουν δύο προσπελάσεις για ανάγνωση και μία για εγγραφή ανά κύκλο, και παράγει ένα αποτέλεσμα ανά κύκλο, στην αριθμητική της μονάδα. Ας υποθέσουμε τη χρήση εντολών οι οποίες προσδιορίζουν δύο παράγοντες προέλευσης και ένα αποτέλεσμα.[4] Το τμήμα (α) του Σχήματος δείχνει ότι, όταν οι εντολές είναι από τη μνήμη

[4] Στην αρχιτεκτονική 370/390, οι μοναδικές εντολές τριών παραγόντων (εντολές καταχωρητών και αποθήκευσης, RS), προσδιορίζουν δύο παράγοντες μέσα σε καταχωρητές και έναν μέσα στη μνήμη. Στο τμήμα (α) του παραδείγματος, υποθέτουμε την ύπαρξη εντολών τριών παραγόντων όπου όλοι οι παράγοντες βρίσκονται στην κύρια μνήμη. Αυτό γίνεται για λόγους σύγκρισης και, στην πραγματικότητα, αυτή η μορφή εντολής θα μπορούσε να επιλεχθεί για την αρχιτεκτονική επεξεργασίας διανυσμάτων.



Σχήμα 17.21: Καταχωρητές που χρησιμοποιούνται από το μηχανισμό επεξεργασίας διανυσμάτων της μηχανής IBM 3090

προς τη μνήμη, κάθε επανάληψη του υπολογισμού απαιτεί συνολικά 18 κύκλους. Με μία αμιγή αρχιτεκτονική από καταχωρητή σε καταχωρητή, η οποία απεικονίζεται στο τμήμα (β), ο αντίστοιχος χρόνος μειώνεται σε 12 κύκλους. Φυσικά, όταν η πράξη εκτελείται από καταχωρητή σε καταχωρητή, οι ποσότητες των διανυσμάτων θα πρέπει να φορτώνονται σε καταχωρητές διανύσματος πριν από τον υπολογισμό και να αποθηκεύονται στη μνήμη εκ των υστέρων. Για μεγάλα διανύσματα, αυτή η σταθερή επιβάρυνση είναι σχετικά μικρή. Το Σχήμα 17.20γ δείχνει ότι, η δυνατότητα καθορισμού μέσα σε μία εντολή, τόσο παραγόντων που βρίσκονται στη μνήμη όσο και παραγόντων που βρίσκονται στους καταχωρητές, μειώνει περαιτέρω το χρόνο εκτέλεσης σε 10 κύκλους ανά επανάληψη. Η τελευταία αυτή μορφή εντολής συμπεριλαμβάνεται στην αρχιτεκτονική επεξεργασίας διανυσμάτων. [5]

Το Σχήμα 17.21 απεικονίζει τους καταχωρητές οι οποίοι αποτελούν τμήμα του μηχανισμού επεξεργασίας διανυσμάτων της μηχανής IBM 3090. Υπάρχουν 16 καταχωρητές διανύσματος 32 bit. Οι καταχωρητές αυτοί, είναι επίσης δυνατόν να συνδεθούν σχηματίζοντας 8 καταχωρητές μήκους 64 bits. Κάθε στοιχείο του καταχωρητή μπορεί να αποθηκεύσει μία ακέραια τιμή, ή έναν αριθμό κινητής υποδιαστολής. Επομένως, οι καταχωρητές διανύσματος χρησιμοποιούνται για ακεραίους 32 και 64

[5] Οι σύνθετες εντολές, οι οποίες περιγράφονται στη συνέχεια, επιφέρουν περαιτέρω μείωση.

bit ή για αριθμούς κινητής υποδιαστολής 32 ή 64 bit.

Η αρχιτεκτονική ορίζει ότι κάθε καταχωρητής περιέχει από 8 μέχρι 512 στοιχεία. Η επιλογή του πραγματικού μήκους συνεπάγεται ένα σχεδιαστικό συμβιβασμό. Ο χρόνος που απαιτείται για την εκτέλεση μίας διανυσματικής πράξης, αποτελείται ουσιαστικά από την επιβάρυνση που οφείλεται στην αρχικοποίηση της διασωλήνωσης και στο γέμισμα των καταχωρητών, συν ένα κύκλο ανά στοιχείο του διανύσματος. Επομένως, η χρήση ενός μεγάλου πλήθους στοιχείων μέσα στον καταχωρητή, μειώνει το χρόνο έναρξης που απαιτείται για τον υπολογισμό. Ωστόσο, αυτό το γεγονός θα πρέπει να αντισταθμίζεται με τον επιπλέον χρόνο ο οποίος απαιτείται για την αποθήκευση και ανάκτηση των καταχωρητών διανύσματος, όταν γίνεται μεταφορά από μία διεργασία σε μία άλλη, και με τα πρακτικά όρια του κόστους και του χώρου. Αυτά τα ζητήματα οδήγησαν στη χρησιμοποίηση 128 στοιχείων ανά καταχωρητή, στην τρέχουσα υλοποίηση της μηχανής 3090.

Τρεις επιπλέον καταχωρητές είναι αναγκαίοι στο μηχανισμό επεξεργασίας διανυσμάτων. Ο καταχωρητής μάσκας διανύσματος, περιέχει τα bits μάσκας τα οποία χρησιμοποιούνται για να επιλεχθούν τα στοιχεία των καταχωρητών διανύσματος τα οποία θα υποστούν επεξεργασία προκειμένου να υλοποιηθεί μία πράξη. Ο καταχωρητής κατάστασης διανύσματος, ο οποίος περιέχει πεδία ελέγχου, όπως είναι ο μετρητής διανύσματος που προσδιορίζει πόσα στοιχεία των καταχωρητών διανύσματος θα υποστούν επεξεργασία. Ο μετρητής δραστηριότητας διανύσματος παρακολουθεί το χρόνο που δαπανάται για την εκτέλεση των διανυσματικών εντολών.

ΣΥΝΘΕΤΕΣ ΕΝΤΟΛΕΣ Όπως περιγράφηκε προηγουμένως, η εκτέλεση των εντολών είναι δυνατόν να επικαλυφθεί χρησιμοποιώντας την αλυσίδωση, προκειμένου να βελτιωθεί η απόδοση. Οι σχεδιαστές του μηχανισμού επεξεργασίας διανυσμάτων της IBM επέλεξαν να μην συμπεριλάβουν αυτή τη δυνατότητα, για πολλούς λόγους. Η αρχιτεκτονική System/370 θα πρέπει να επεκταθεί ώστε να διαχειριστεί τις πολύπλοκες διακοπές (συμπεριλαμβανομένων των συνεπειών τους στη διαχείριση της ιδεατής μνήμης), ενώ αντίστοιχες αλλαγές απαιτούνται και στο λογισμικό. Ένα πιο βασικό ζήτημα, ήταν το κόστος που θα προέκυπτε από τη συμπερίληψη στο μηχανισμό, των επιπλέον ελέγχων και μονοπατιών προσπέλασης των καταχωρητών που χρειάζονται για τη γενικευμένη αλυσίδωση.

Αντί αυτών, παρέχονται τρεις λειτουργίες οι οποίες συνδυάζουν σε μία εντολή (έναν κωδικό εντολής), τις πιο συνηθισμένες ακολουθίες που υπάρχουν στους υπολογισμούς διανυσμάτων, δηλαδή τον πολλαπλασιασμό ο οποίος ακολουθείται από πρόσθεση, ή αφαίρεση, ή άθροισμα. Για παράδειγμα, η εντολή MULTIPLY-AND-ADD (πολλαπλασιασμού και πρόσθεσης), η οποία είναι εντολή από καταχωρητή σε καταχωρητή, προσκομίζει ένα διάνυσμα από τη μνήμη, το πολλαπλασιάζει με ένα διάνυσμα που βρίσκεται σε έναν καταχωρητή, και προσθέτει το γινόμενο με ένα τρίτο διάνυσμα που βρίσκεται σε έναν καταχωρητή. Χρησιμοποιώντας τις σύνθετες εντολές MULTIPLY-AND-ADD και MULTIPLY-AND-SUBTRACT (πολλαπλασιασμού και αφαίρεσης) στο παράδειγμα του Σχήματος 17.20, ο συνολικός χρόνος εκτέλεσης της επανάληψης θα μειωθεί από 10 σε 8 κύκλους.

Σε αντίθεση με την αλυσίδωση, οι σύνθετες εντολές δεν απαιτούν τη χρήση επιπλέον καταχωρητών προσωρινής αποθήκευσης των ενδιάμεσων αποτελεσμάτων, ενώ απαιτούν μία λιγότερη προσπέλαση σε καταχωρητή. Για παράδειγμα, θεωρείστε την παρακάτω αλυσίδα:

$$A \rightarrow VR1$$
$$VR1 + VR2 \rightarrow VR1$$

Σε αυτή την περίπτωση, απαιτούνται δύο λειτουργίες αποθήκευσης στον καταχωρητή διανύσματος VR1. Στην αρχιτεκτονική της IBM, υπάρχει μία εντολή πρόσθεσης (ADD) από καταχωρητή σε καταχωρητή. Με αυτή την εντολή, μόνον το άθροισμα τοποθετείται στον R1. Επίσης, η σύνθετη εντολή αποφεύγει την ανάγκη απεικόνισης της ταυτόχρονης εκτέλεσης ενός πλήθους εντολών στην περιγραφή της κατάστασης μηχανής. Αυτό το γεγονός απλοποιεί την αποθήκευση και την ανάκληση της κατάστασης από το λειτουργικό σύστημα, αλλά και την διαχείριση των διακοπών.

ΤΟ ΣΥΝΟΛΟ ΕΝΤΟΛΩΝ Ο Πίνακας 17.3 συνοψίζει τις αριθμητικές και λογικές πράξεις οι οποίες

Μηχανισμός επεξεργασίας διανυσμάτων της μηχανής IBM 3090. Αριθμητικές και λογικές εντολές.

	Τύποι δεδομένων						
	Κινητή υποδιαστολή						
Πράξη	**Μεγάλου Μήκους**	**Μικρού Μήκους**	**Δυαδική ή λογική**	**Θέσεις παραγόντων**			
Πρόσθεση	FL	FS	BI	V+V → V	V+S → V	Q+V → V	Q+S → V
Αφαίρεση	FL	FS	BI	V-V→ V	V-S → V	Q-V→ V	Q-S → V
Πολλαπλασιασμός	FL	FS	BI	V × V→ V	V × V→ V	Q × V→ V	Q × S→ V
Διαίρεση	FL	FS	-	V/V→ V	V/S→ V	Q/V→ V	Q/S→ V
Σύγκριση	FL	FS	BI	V· V→ V	V· S→ V	Q· V→ V	Q· S→ V
Πολλαπλασιασμός και πρόσθεση	FL	FS	-		V+ V × S → V	V+ Q × V → V	V+ Q × S → V
Πολλαπλασιασμός και αφαίρεση	FL	FS	-		V- V × S → V	V- Q × S → V	V- Q × S → V
Πολλαπλασιασμός και συσσώρευση	FL	FS	-	P+·V→ V	P+·S→ V		
Συμπλήρωμα	FL	FS	BI	-V→ V			
Θετική απόλυτη τιμή	FL	FS	BI	|V| → V			
Αρνητική απόλυτη τιμή	FL	FS	BI	-|V| → V			
Μέγιστη τιμή	FL	FS	-			Q · V → Q	
Απόλυτη μέγιστη τιμή	FL	FS	-			Q · V → Q	
Ελάχιστη τιμή	FL	FS	-			Q · V → Q	
Λογική αριστερή ολίσθηση	-	-	LO	·V→ V			
Λογική δεξιά ολίσθηση	-	-	LO	·V→ V			
Λογικό ΚΑΙ	-	-	LO	V& V → V	V& S → V	Q& V → V	Q& S → V
Λογικό Ή	-	-	LO	V |V→ V	V |S→ V	Q |V→ V	Q |S→ V
Λογικό Αποκλειστικό Ή	-	-	LO	V⊕V→ V	V⊕S→ V	Q⊕V→ V	Q⊕S→ V

Εξήγηση:

Τύποι Δεδομένων

FL Κινητής υποδιαστολής μεγάλου μήκους
FS Κινητής υποδιαστολής μικρού μήκους
BI Δυαδικού ακεραίου
LO Λογικός

Θέσεις Παραγόντων

V Καταχωρητής διανυσμάτων
S Μνήμη
Q Βαθμωτή (γενικός καταχωρητής ή καταχωρητής κινητής υποδιαστολής)
P Μερικά αθροίσματα στον καταχωρητή διανυσμάτων
- Ειδική πράξη

ορίζονται στην αρχιτεκτονική επεξεργασίας διανυσμάτων. Επιπλέον, υπάρχουν εντολές φόρτωσης από τη μνήμη προς καταχωρητές και εντολές αποθήκευσης από καταχωρητές προς τη μνήμη. Παρατηρείστε ότι πολλές από τις εντολές χρησιμοποιούν μία μορφή τριών παραγόντων. Επίσης, πολλές εντολές έχουν ένα πλήθος παραλλαγών, αναλόγως με τη θέση στην οποία βρίσκονται οι παράγοντες. Ένας παράγοντας προέλευσης μπορεί να είναι ένας καταχωρητής διανύσματος (V), η μνήμη (S), ή ένας βαθμωτός καταχωρητής (Q). Ο προορισμός είναι πάντοτε ένας καταχωρητής διανύσματος, με εξαίρεση την πράξη της σύγκρισης, όπου το αποτέλεσμα μεταβαίνει στον καταχωρητή μάσκας διανύσματος. Με όλες αυτές τις παραλλαγές, το συνολικό πλήθος των κωδικών εντολής (διαφορετικές εντολές), είναι ίσο με 171. *Ωστόσο, αυτός ο αρκετά μεγάλος αριθμός, δεν είναι ακριβός ως προς την υλοποίηση, όπως θα φανταζόταν κάποιος.* Από τη στιγμή που η μηχανή παρέχει τις αριθμητικές μονάδες και τα μονοπάτια δεδομένων για να τροφοδοτεί τις διασωληνώσεις διανυσμάτων με παράγοντες από τη μνήμη, τους βαθμωτούς καταχωρητές, και τους καταχωρητές διανύσματος, το μεγαλύτερο μέρος του κόστους έχει καλυφθεί. Η αρχιτεκτονική έχει τη δυνατότητα, με μικρή διαφορά κόστους, να παρέχει ένα πλούσιο σύνολο από παραλλαγές όσον αφορά τη χρήση αυτών των καταχωρητών και δομών διασωλήνωσης.

Οι περισσότερες εντολές του Πίνακα 17.3 είναι αυτοεξηγούμενες. Οι δύο εντολές άθροισης απαιτούν περαιτέρω εξήγηση. Η πράξη συσσώρευσης προσθέτει τα στοιχεία ενός διανύσματος (ACCUMULATE) ή τα στοιχεία του γινομένου δύο διανυσμάτων (MULTIPLY AND ACCUMULATE). Αυτές οι εντολές παρουσιάζουν ένα ενδιαφέρον σχεδιαστικό πρόβλημα. Επιθυμούμε να εκτελέσουμε αυτή την πράξη με όσο το δυνατόν μεγαλύτερη ταχύτητα, εκμεταλλευόμενοι πλήρως την αριθμητική και λογική μονάδα που χρησιμοποιεί διασωλήνωση. Η δυσκολία βρίσκεται στο γεγονός ότι το άθροισμα των δύο αριθμών που τοποθετούνται στη διασωλήνωση δεν είναι διαθέσιμο, παρά μόνον μετά το πέρας αρκετών κύκλων. Επομένως, δεν είναι δυνατόν να προστεθεί ένα τρίτο στοιχείο του διανύσματος στο άθροισμα των δύο πρώτων στοιχείων, έως ότου τα δύο αυτά στοιχεία έχουν περάσει από ολόκληρη τη διασωλήνωση. Για να ξεπεραστεί αυτό το πρόβλημα, τα στοιχεία του διανύσματος προστίθενται με τέτοιο τρόπο, ώστε να παράγουν 4 μερικά αθροίσματα. Πιο συγκεκριμένα, τα στοιχεία 0, 4, 8, 12, ...124 προστίθενται με αυτή τη σειρά για να παράγουν το μερικό άθροισμα 0, τα στοιχεία 1, 5, 9, 13,.....125 προστίθενται για να παράγουν το μερικό άθροισμα 1, τα στοιχεία 2, 6, 10, 14,.....126 προστίθενται για να παράγουν το μερικό άθροισμα 2, και τέλος, τα στοιχεία 3, 7, 11, 15,.....127 προστίθενται για να παράγουν το μερικό άθροισμα 3. Καθένα από αυτά τα μερικά αθροίσματα μπορεί να προχωρήσει μέσα στη διασωλήνωση με τη μέγιστη ταχύτητα, επειδή η καθυστέρηση σε αυτή είναι το πολύ 4 κύκλοι. Χρησιμοποιείται ένας ξεχωριστός καταχωρητής διανύσματος, στον οποίο αποθηκεύονται τα μερικά αθροίσματα. Όταν ολοκληρωθεί η επεξεργασία όλων των στοιχείων του αρχικού διανύσματος, τα 4 μερικά αθροίσματα προστίθενται παράγοντας το τελικό αποτέλεσμα. Η απόδοση αυτής της δεύτερης φάσης δεν είναι τόσο σημαντική, επειδή συμπεριλαμβάνει μόνον 4 στοιχεία.

17.8 ΠΡΟΤΕΙΝΟΜΕΝΗ ΜΕΛΕΤΗ

Στην αναφορά [CATA94] παρουσιάζονται οι αρχές των πολυεπεξεργαστών και εξετάζονται λεπτομερώς οι συμμετρικοί πολυπεπεξεργαστές που βασίζονται στην αρχιτεκτονική SPARC. Επίσης, οι συμμετρικοί πολυεπεξεργαστές εξετάζονται με κάποιες λεπτομέρειες στις αναφορές [STON93] και [HWAN93].

Το άρθρο [MILE00] αποτελεί μία γενική θεώρηση των αλγορίθμων και τεχνικών συνοχής της κρυφής μνήμης για συστήματα πολυεπεξεργαστών, με έμφαση στην απόδοση. Μία άλλη μελέτη πάνω σε ζητήματα συνοχής της κρυφής μνήμης σε συστήματα πολυεπεξεργαστών, είναι εκείνη της αναφοράς [LILJ93]. Στο άρθρο [TOMA93] περιέχονται ανατυπώσεις από πολλά σημαντικά άρθρα πάνω στο αντικείμενο.

Η αναφορά [UNGE02] αποτελεί μία εξαιρετική μελέτη πάνω στις έννοιες των επεξεργαστών πολλαπλών νημάτων και chip πολυπεπεξεργαστών. Στο άρθρο [UNGE03] υπάρχει εκτενής μελέτη προτεινόμενων και σημερινών επεξεργαστών πολλαπλών νημάτων, οι οποίοι χρησιμοποιούν άμεση πολυνημάτωση.

Μία λεπτομερής εξέταση των συστάδων υπάρχει στα άρθρα [BUYY99a] και [BUYY99b]. Στο βιβλίο [WEYG01] υπάρχει μία λιγότερο τεχνική μελέτη των συστάδων, με καλά σχόλια πάνω σε διάφορα εμπορικά προϊόντα. Στο άρθρο [DESA05] περιγράφεται η αρχιτεκτονική φετών εξυπηρετών της IBM. Καλές περιγραφές σχετικά με τους υπολογισμούς διανυσμάτων υπάρχουν στις αναφορές [STON93] και [HWAN93].

BUYY99a Buyya, R. *High-Performance Cluster Computing: Architectures and Systems.* Upper Saddle River, NJ: Prentice Hall, 1999.

BUYY99b Buyya, R. *High-Performance Cluster Computing: Programming and Applications* Upper Saddle River, NJ: Prentice Hall, 1999.

CATA94 Catanzaro, B. *Multiprocessor System Architectures.* Mountain View, CA: Sunsoft Press, 1994.

DESA05 Desai,D., et al."BladeCenter System Overview." *IBM Journal of Research and Development,* November 2005.

HWAN93 Hwang, K *Advanced Computer Architecture.* New York: McGraw-Hill, 1993.

LILJ93 Lilja, D. "Cache Coherence in Large-Scale Shared-Memory Multiprocessors: Issues and Comparisons."*ACM Computing Surveys,* September 1993.

MILE00 Milenkovic, A. "Achieving High Performance in Bus-Based Shared-Memory Multiprocessors." *IEEE Concurrency,* July-September 2000.

STON93 Stone, H. *High-Performance Computer Architecture.* Reading, MA: Addison- Wesley, 1993.

TOMA93 Tomasevic, M., and Milutinovic,V. *The Cache Coherence Problem in Shared- Memory Multiprocessors: Hardware Solutions.* Los Alamitos, CA: IEEE Computer Society Press, 1993.

UNGE02 Ungerer, T.; Rubic, B.; and Silc, J. "Multithreaded Processors." *The Computer Journal,* No. 3, 2002.

UNGE03 Ungerer,T.; Rubic, B.; and Silc, J. " A Survey of Processors with Explicit Multithreading." *ACM Computing Surveys,* March, 2003.

WEYG01 Weygant, P. *Clusters for High Availability.* Upper Saddle River, NJ: Prentice Hall, 2001.

Προτεινόμενοι Διαδικτυακοί Τόποι:

- **IEEE Computer Society Task Force on Cluster Computing:** Ένας διεθνής τόπος συζητήσεων με σκοπό την προώθηση της έρευνας και εκπαίδευσης με αντικείμενο τις συστάδες υπολογιστών.

17.9 ΒΑΣΙΚΟΙ ΟΡΟΙ, ΕΡΩΤΗΣΕΙΣ ΕΠΑΝΑΛΗΨΗΣ ΚΑΙ ΠΡΟΒΛΗΜΑΤΑ

Βασικοί Όροι

αδιάκριτο πρωτόκολλο	μονοεπεξεργαστής	πρωτόκολλο καταλόγου
βλάβη με επιστροφή	ομοιόμορφη προσπέλαση μνήμης (UMA)	συμμετρικός πολυεπεξεργαστής (SMP)
βλάβη με μεταφορά	παθητική αναμονή	συνοχή κρυφής μνήμης
ενεργός αναμονή	πολυεπεξεργαστής	συστάδα
μη ομοιόμορφη προσπέλαση μνήμης (NUMA)	πρωτόκολλο MESI	
μηχανισμός επεξεργασίας διανυσμάτων		

Ερωτήσεις Επανάληψης

17.1. Να παραθέσετε και να ορίσετε σύντομα τρεις τύπους οργάνωσης ενός υπολογιστικού συστήματος.

17.2. Ποιά είναι τα βασικά χαρακτηριστικά ενός συμμετρικού πολυεπεξεργαστή;

17.3. Ποιά είναι ορισμένα από τα εν δυνάμει πλεονεκτήματα του συμμετρικού πολυεπεξεργαστή σε σχέση με έναν μονοεπεξεργαστή;

17.4. Ποιά είναι ορισμένα από τα βασικά σημεία της σχεδίασης του λειτουργικού συστήματος ενός συμμετρικού πολυεπεξεργαστή;

17.5. Ποιά είναι η διαφορά ανάμεσα στις τεχνικές διατήρησης της συνοχής της κρυφής μνήμης, οι οποίες βασίζονται στο υλικό και στο λογισμικό;

17.6. Τι σημαίνει κάθε μία από τις τέσσερις καταστάσεις του πρωτοκόλλου MESI;

17.7. Ποιά είναι μερικά από τα βασικά ωφέλη της συσταδοποίησης;

17.8. Ποιά είναι η διαφορά ανάμεσα στη βλάβη με επιστροφή και στη βλάβη με μεταφορά;

17.9. Ποιές είναι οι διαφορές ανάμεσα στις υλοποιήσεις UMA, NUMA, και CC-NUMA;

Προβλήματα

17.1 Έστω ότι a είναι το ποσοστό του κώδικα ενός προγράμματος ο οποίος είναι δυνατόν να εκτελεστεί ταυτόχρονα από n επεξεργαστές ενός συστήματος. Έστω επίσης ότι το υπόλοιπο μέρος του κώδικα θα πρέπει να εκτελεστεί ακολουθιακά, από έναν μόνο επεξεργαστή. Κάθε επεξεργαστής έχει ρυθμό εκτέλεσης ίσο με x MIPS.

α. Να εξάγετε μία έκφραση για τον ενεργό ρυθμό MIPS ως προς τα n, a, και x, όταν χρησιμοποιείται το σύστημα αποκλειστικά για την εκτέλεση αυτού του προγράμματος.

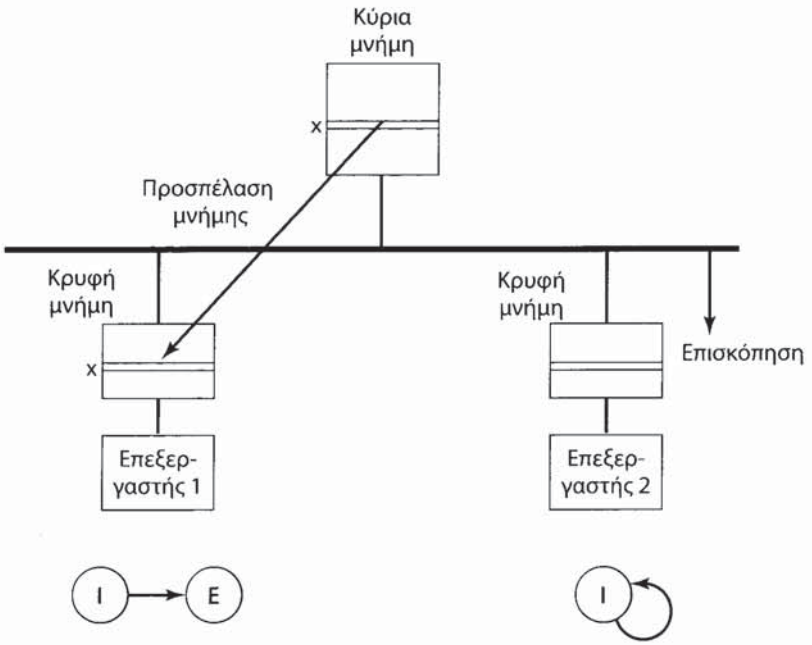
β. Αν $n = 16$ και $x = 4$ MIPS, να προσδιορίσετε την τιμή του a, η οποία θα δώσει απόδοση συστήματος ίση με 40 MIPS.

17.2 Ένας πολυεπεξεργαστής αποτελούμενος από 8 επεξεργαστές έχει συνδεδεμένες σε αυτόν 20 ταινίες. Υπάρχει ένα μεγάλο πλήθος εργασιών οι οποίες έχουν υποβληθεί στο σύστημα, και κάθε μία απαιτεί το πολύ 4 ταινίες για να ολοκληρώσει την εκτέλεση. Έστω ότι κάθε εργασία ξεκινά να εκτελείται με τρεις μόνον ταινίες για μία μεγάλη περίοδο, πριν απαιτήσει και την τέταρτη ταινία για μικρό χρονικό διάστημα, λίγο πριν από την ολοκλήρωσή της. Επίσης, έστω ότι υπάρχει μία ατελείωτη ροή εργασιών αυτής της μορφής.

α. Υποθέστε ότι ο χρονοδρομολογητής του λειτουργικού συστήματος δεν θα ξεκινήσει την εργασία, εκτός αν υπάρχει διαθεσιμότητα τεσσάρων ταινιών. Όταν ξεκινά μία εργασία, εκχωρούνται αμέσως τέσσερις ταινίες, οι οποίες δεν απελευθερώνονται παρά μόνον όταν τελειώσει η εργασία. Ποιό είναι το μέγιστο πλήθος εργασιών οι οποίες εκτελούνται ταυτόχρονα; Ποιό είναι το μέγιστο και ελάχιστο πλήθος ταινιών οι οποίες είναι πιθανό να μείνουν αδρανείς ως αποτέλεσμα αυτής της στρατηγικής;

β. Να προτείνετε μία εναλλακτική πολιτική με την οποία θα βελτιωθεί ο βαθμός χρήσης των ταινιών και ταυτόχρονα, θα αποφεύγονται τα αδιέξοδα του συστήματος. Ποιό είναι το μέγιστο πλήθος εργασιών οι οποίες εκτελούνται ταυτόχρονα; Ποιό είναι το όριο του πλήθους των αδρανών ταινιών;

17.3 Μπορείτε να προβλέψετε κάποιο πρόβλημα στην προσέγγιση της κρυφής μνήμης με μία μόνον εγγραφή, σε ένα σύστημα πολυεπεξεργαστών βασισμένο σε δίαυλο; Αν ναι, να προτείνετε μία λύση.



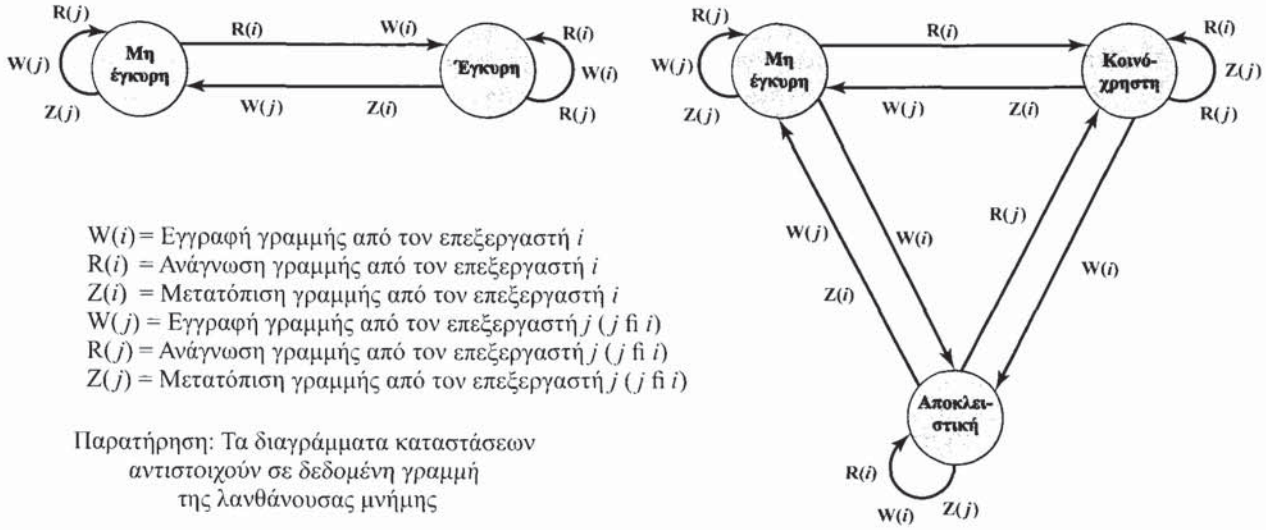
Παράδειγμα του πρωτοκόλλου MESI: Ο επεξεργαστής 1 διαβάζει τη γραμμή x

17.4 Θεωρείστε μία κατάσταση στην οποία δύο επεξεργαστές μίας διάταξης συμμετρικών πολυεπεξεργαστών, απαιτεί ανά διαστήματα την προσπέλαση της ίδιας γραμμής δεδομένων από τη μνήμη. Αμφότεροι οι επεξεργαστές διαθέτουν μία κρυφή μνήμη και χρησιμοποιούν το πρωτόκολλο MESI. Αρχικά, και οι δύο κρυφές μνήμες διαθέτουν ένα μη έγκυρο αντίγραφο της γραμμής. Το Σχήμα 17.22 απεικονίζει τις συνέπειες της ανάγνωσης μίας γραμμής x από τον επεξεργαστή P1. Αν αυτή είναι η αρχή μίας σειράς προσπελάσεων, να σχεδιάσετε τα επόμενα διαγράμματα για την παρακάτω ακολουθία ενεργειών:

1. Ο P2 διαβάζει τη γραμμή x.
2. Ο P1 γράφει στη γραμμή x (για λόγους ευκρίνειας, ονομάστε τη γραμμή της κρυφής μνήμης του επεξεργαστή P1 x').
3. Ο P1 γράφει στη γραμμή x (ονομάστε τη γραμμή της κρυφής μνήμης του επεξεργαστή P1 x'').
4. Ο P2 διαβάζει τη γραμμή x.

17.5 Το Σχήμα 17.23 απεικονίζει τα διαγράμματα κατάστασης δύο πιθανών πρωτοκόλλων συνοχής της κρυφής μνήμης. Να εξάγετε και να εξηγήσετε κάθε πρωτόκολλο, και να το συγκρίνετε με το πρωτόκολλο MESI.

17.6 Θεωρείστε ένα συμμετρικό πολυεπεξεργαστή, ο οποίες διαθέτει κρυφές μνήμες Επιπέδου 1 και 2 και χρησιμοποιεί το πρωτόκολλο MESI. Όπως περιγράφηκε στην Ενότητα 17.3, μία από τις τέσσερις καταστάσεις σχετίζεται με κάθε γραμμή της κρυφής μνήμης Επιπέδου 2. Είναι αναγκαίες και οι τέσσερις καταστάσεις για κάθε γραμμή, στην κρυφή μνήμη Επιπέδου 1; Αν ναι, γιατί; Αν όχι, να εξηγήσετε ποια ή ποιες καταστάσεις είναι δυνατόν να εξαλειφθούν.



Σχήμα 17.23: Δύο πρωτόκολλα συνοχής της κρυφής μνήμης

17.7 Μία αρχική έκδοση του μεγάλου υπολογιστικού συστήματος της IBM S/390 G4, χρησιμοποιούσε τρία επίπεδα κρυφής μνήμης. Όπως συμβαίνει και με τη μηχανή z990, μόνον το πρώτο επίπεδο βρισκόταν στο chip του επεξεργαστή [ονομαζόταν μονάδα επεξεργαστή (Processor Unit-PU)]. Η κρυφή μνήμη Επιπέδου 2 ήταν επίσης παρόμοια με εκείνη του συστήματος z990. Η κρυφή μνήμη Επιπέδου 3 βρισκόταν σε ένα ξεχωριστό chip, το οποίο ενεργούσε ως ελεγκτής μνήμης και βρισκόταν ανάμεσα στις κρυφές μνήμες Επιπέδου 2 και τις κάρτες μνήμης. Ο Πίνακας 17.4 δείχνει την απόδοση μίας διάταξης τριών επιπέδων κρυφής μνήμης για το σύστημα IBM S/390. Ο σκοπός αυτού του προβλήματος είναι να προσδιοριστεί αν η συμπερίληψη ενός τρίτου επιπέδου κρυφής μνήμης έχει αξία. Να προσδιορίσετε την επιβάρυνση προσπέλασης (μέσο πλήθος κύκλων της PU), για ένα σύστημα το οποίο διαθέτει κρυφή μνήμη μόνον Επιπέδου 1, και να κανονικοποιήσετε αυτή την τιμή στην τιμή 1.0. Στη συνέχεια, να προσδιορίσετε την κανονικοποιημένη επιβάρυνση προσπέλασης όταν χρησιμοποιούνται κρυφές μνήμες Επιπέδου 1 και 2, και την επιβάρυνση προσπέλασης όταν χρησιμοποιούνται και τα τρία επίπεδα κρυφής μνήμης. Παρατηρείστε τη βελτίωση που παρουσιάζεται σε κάθε περίπτωση και εκφράστε την άποψή σας για τη χρησιμοποίηση της κρυφής μνήμης Επιπέδου 3.

Πίνακας 17.4: Τυπικός ρυθμός ευστοχίας της κρυφής μνήμης στη διάταξη συμμετρικού πολυεπεξεργαστή S/390 [MAK97]

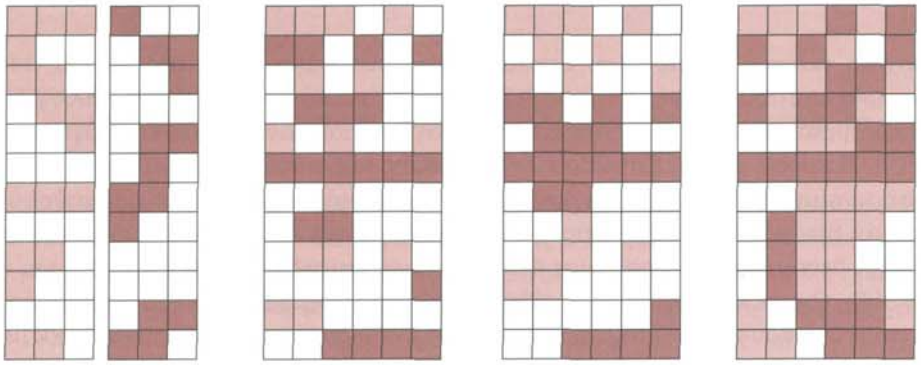
Υποσύστημα Μνήμης	Επιβάρυνση Προσπέλασης (Κύκλοι της PU)	Μέγεθος Κρυφής Μνήμης	Ρυθμός Ευστοχίας (%)
Κρυφή μνήμη Επιπέδου 1	1	32 KB	89
Κρυφή μνήμη Επιπέδου 2	5	256 KB	5
Κρυφή μνήμη Επιπέδου 3	14	2 MB	3
Μνήμη	32	8 GB	3

17.8 **α.** Θεωρείστε έναν μονοεπεξεργαστή με ξεχωριστές κρυφές μνήμες δεδομένων και εντολών, με λόγους ευστοχίας H_d και H_i, αντίστοιχα. Ο χρόνος προσπέλασης από τον επεξεργαστή στην κρυφή μνήμη είναι c κύκλοι ρολογιού, και ο χρόνος μεταφοράς ενός μπλοκ ανάμεσα στην κύρια και στην κρυφή μνήμη είναι b κύκλοι ρολογιού. Έστω ότι f_i είναι ένα ποσοστό

προσπελάσεων της μνήμης για εντολές, και f_d είναι ένα ποσοστό βρώμικων γραμμών της κρυφής μνήμης δεδομένων, ανάμεσα σε γραμμές οι οποίες έχουν αντικατασταθεί. Υποθέστε ότι χρησιμοποιείται μία πολιτική εγγραφής προς τα πίσω και προσδιορίστε τον ενεργό χρόνο προσπέλασης της μνήμης με όρους των παραμέτρων που μόλις ορίστηκαν.

β. Να εξετάσετε τώρα ένα συμμετρικό πολυεπεξεργαστή βασισμένο σε δίαυλο. Στο σύστημα αυτό, κάθε επεξεργαστής έχει τα χαρακτηριστικά τα οποία περιγράφηκαν στο ερώτημα (α). Κάθε επεξεργαστής θα πρέπει να διαχειρίζεται τις μη έγκυρες γραμμές της κρυφής μνήμης, εκτός από τις λειτουργίες ανάγνωσης και εγγραφής. Αυτό το γεγονός επηρρεάζει τον ενεργό χρόνο προσπέλασης της μνήμης. Έστω ότι f_{inv} είναι το ποσοστό των αναφορών δεδομένων τα οποία προκαλούν την αποστολή σημάτων ένδειξης μη έγκυρων γραμμών προς άλλες κρυφές μνήμες. Ο επεξεργαστής ο οποίος στέλνει το σήμα απαιτεί t κύκλους ρολογιού για να ολοκληρώσει τη λειτουργία. Οι άλλοι επεξεργαστές δεν εμπλέκονται σε αυτή τη λειτουργία. Να προσδιορίσετε τον ενεργό χρόνο προσπέλασης της μνήμης.

17.9 Ποιά εναλλακτική μορφή οργάνωσης προτείνει κάθε τμήμα του Σχήματος 17.24;



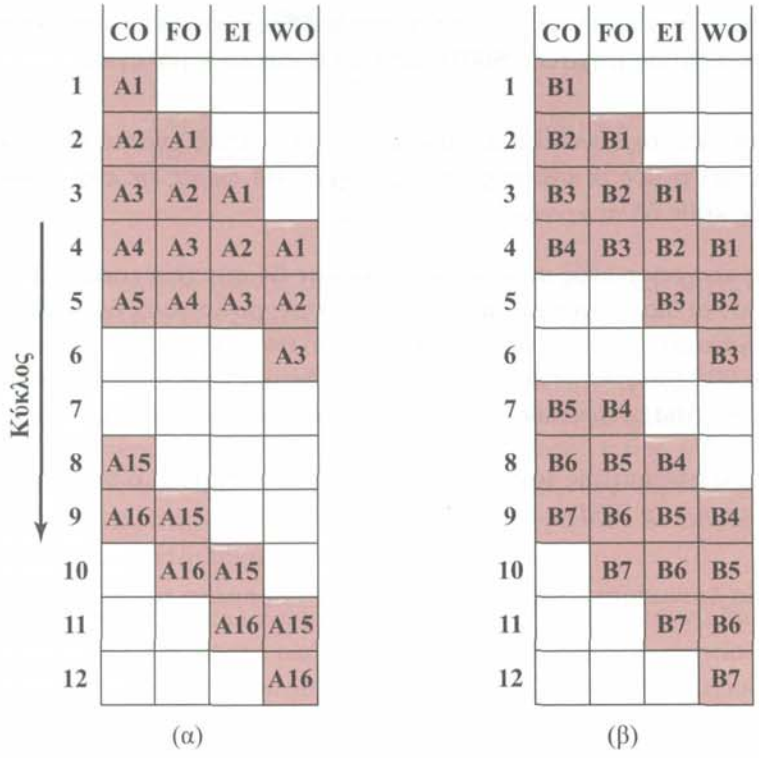
Σχήμα 17.24: Διάγραμμα για το πρόβλημα 18.9

17.10 Στο Σχήμα 17.8, ορισμένα από τα διαγράμματα δείχνουν οριζόντιες γραμμές, οι οποίες είναι μερικώς συμπληρωμένες. Σε άλλες περιπτώσεις, υπάρχουν γραμμές οι οποίες είναι εντελώς κενές. Αυτές αναπαριστούν δύο διαφορετικές μορφές απώλειας της αποτελεσματικότητας. Εξηγήστε.

17.11 Θεωρείστε την απεικόνιση μίας λειτουργίας διασωλήνωσης η οποία παρουσιάστηκε στο Σχήμα 12.13 και δίνεται εκ νέου στο Σχήμα 17.25α. Σε αυτή την απεικόνιση, αγνοούμε τα στάδια ανάκλησης και αποκωδικοποίησης και παρουσιάζουμε την εκτέλεση ενός νήματος Α. Το Σχήμα 17.25β απεικονίζει την εκτέλεση ενός διαφορετικού νήματος Β. Στις δύο αυτές περιπτώσεις, χρησιμοποιείται ένας απλός επεξεργαστής με διασωλήνωση.

α. Να δείξετε ένα διάγραμμα έκδοσης εντολών, παρόμοιο με εκείνο του Σχήματος 17.8α, για καθένα από τα δύο νήματα.

β. Υποθέστε ότι τα δύο νήματα πρόκειται να εκτελεστούν παράλληλα σε ένα chip πολυεπεξεργαστών, όπου καθένας από τους δύο επεξεργαστές του chip χρησιμοποιεί μία απλή διασωλήνωση. Να δείξετε ένα διάγραμμα έκδοσης εντολών το οποίο είναι παρόμοιο με το Σχήμα 17.8κ. Επίσης, να δείξετε ένα διάγραμμα εκτέλεσης της διασωλήνωσης, παρόμοιο με αυτό του Σχήματος 17.25.



Σχήμα 17.25: Δύο νήματα εκτέλεσης

γ. Υποθέστε μία υπερβαθμωτή αρχιτεκτονική η οποία υποστηρίζει δύο νήματα. Να επαναλάβετε το ερώτημα (β) για μία υλοποίηση φυλλωμένης υπερβαθμωτής πολυνημάτωσης, υποθέτοντας ότι δεν υπάρχουν εξαρτήσεις δεδομένων. *Παρατήρηση:* Δεν υπάρχει μοναδική απάντηση. Θα πρέπει να κάνετε υποθέσεις σχετικά με το λανθάνοντα χρόνο και τις προτεραιότητες.

δ. Να επαναλάβετε το ερώτημα (γ) για μία υλοποίηση παρεμποδισμένης υπερβαθμωτής πολυνημάτωσης.

ε. Να επαναλάβετε για μία αρχιτεκτονική συμμετρικών πολυεπεξεργαστών, η οποία υποστηρίζει 4 νήματα.

17.12 Το παρακάτω τμήμα κώδικα θα πρέπει να εκτελεστεί 64 φορές για να αποτιμηθεί η παρακάτω αριθμητική διανυσματική έκφραση: D(I)=A(I)+B(I)× C(I), όπου 0≤I≤63.

```
Load R1, B(I)       /R1← Memory(α+I)
Load R2, C(I)       /R2← Memory(β+I)
Multiply R1, R2     /R1← (R1) × (R2)
Load R3, A(I)       /R3← Memory(γ+I)
Add R3, R1          /R3← (R3) + (R1)
Load D1, R3         /Memory(θ+I)← (R3)/
```

όπου R1, R2, και R3 είναι καταχωρητές του επεξεργαστή, και $\alpha, \beta, \gamma, \theta$ είναι οι διευθύνσεις έναρξης των πινάκων B(I), C(I), A(I), και D(I), αντίστοιχα. Υποθέστε ότι κάθε λειτουργία αποθήκευσης (Store) και φόρτωσης (Load) απαιτεί τέσσερις κύκλους ρολογιού, ενώ η πράξη της

πρόσθεσης απαιτεί δύο κύκλους. Τέλος, ο πολλαπλασιαστής είτε του μονοεπεξεργαστή ή του απλού επεξεργαστή μίας μηχανής SIMD, απαιτεί 8 κύκλους ρολογιού.

α. Να υπολογίσετε το συνολικό πλήθος των κύκλων επεξεργαστή οι οποίοι απαιτούνται για να εκτελεστεί αυτός ο κώδικας επαναληπτικά 64 φορές σε ένα μονοεπεξεργαστή SISD, αγνοώντας κάθε άλλη χρονική καθυστέρηση.

β. Θεωρείστε τη χρήση ενός υπολογιστή SIMD με 64 στοιχεία επεξεργασίας, προκειμένου να εκτελεστούν οι διανυσματικές πράξεις σε έξι συγχρονισμένες διανυσματικές εντολές, πάνω σε διανυσματικά δεδομένα αποτελούμενα από 64 στοιχεία. Η ταχύτητα του ρολογιού είναι ίδια με εκείνη του μονοεπεξεργαστή SIMD. Να υπολογίσετε το συνολικό χρόνο εκτέλεσης της μηχανής SIMD, αγνοώντας τις μεταδόσεις εντολών και άλλες καθυστερήσεις.

γ. Ποιά είναι η αύξηση της ταχύτητας την οποία επιτυγχάνει ο υπολογιστής SIMD σε σχέση με τον υπολογιστή SISD;

17.13 Να παράγετε μία διανυσματοποιημένη έκδοση του παρακάτω προγράμματος:

```
      DO 20 I = 1, N
      B(I, 1) = 0
      DO 10 J = 1,M
      A(I) =A(I) + B(I, J) × C(I, J)
10    CONTINUE
      D(I) = E(I) + A(I)
20    CONTINUE
```

17.14 Ένα πρόγραμμα εφαρμογής εκτελείται σε μία συστάδα αποτελούμενη από 9 υπολογιστές. Ένα μετροπρόγραμμα χρειάστηκε χρόνο T για να εκτελεστεί σε αυτή τη συστάδα. Επιπλέον, βρέθηκε ότι 25% του T, ήταν χρόνος στον οποίο η εφαρμογή έτρεχε ταυτόχρονα και στις 9 μηχανές. Κατά τον υπόλοιπο χρόνο, η εφαρμογή έτρεχε σε μία μόνον μηχανή.

α. Να υπολογίσετε την αύξηση της ταχύτητας κάτω από τη συνθήκη η οποία αναφέρθηκε, σε σύγκριση με την περίπτωση εκτέλεσης του προγράμματος σε έναν απλό υπολογιστή. Επίσης, να υπολογίσετε το a, το οποίο αντιστοιχεί στο ποσοστό του κώδικα που έχει παραλληλοποιηθεί (έχει προγραμματιστεί ή μεταγλωττιστεί κατά τέτοιο τρόπο ώστε να εκτελείται στη συστάδα) στο προηγούμενο πρόγραμμα.

β. Έστω ότι έχουμε τη δυνατότητα να χρησιμοποιήσουμε με ενεργό τρόπο 17 υπολογιστές αντί των 9, στο παραλληλοποιημένο κομμάτι του κώδικα. Να υπολογίσετε την αύξηση της ταχύτητας.

17.15 Το παρακάτω πρόγραμμα το οποίο είναι γραμμένο σε γλώσσα FORTRAN πρόκειται να εκτελεστεί σε έναν υπολογιστή, ενώ η παράλληλη έκδοσή του πρόκειται να εκτελεστεί σε μία συστάδα 32 υπολογιστών:

Έστω ότι οι γραμμές 2 και 4 απαιτούν δύο χρόνους κύκλου μηχανής, συμπεριλαμβανομένων όλων των δραστηριοτήτων του επεξεργαστή, αλλά και των προσπελάσεων της μνήμης. Αγνοήστε τυχόν επιβαρύνσεις οι οποίες οφείλονται σε προτάσεις ελέγχου βρόχων (γραμμές 1, 3, 5) και κάθε άλλη επιβάρυνση του συστήματος και τυχόν διαμάχες πόρων.

```
L1:      DO 10 I = 1, 1024
L2:         SUM(I) = 0
L3:         DO 20 J = 1, I
L4: 20          SUM(I) = SUM(I) + I
L5: 10   CONTINUE
```

α. Ποιός είναι ο συνολικός χρόνος εκτέλεσης (σε χρόνους κύκλων μηχανής) του προγράμματος, αν αυτό εκτελεστεί σε έναν απλό υπολογιστή;

β. Να διαχωρίσετε τις επαναλήψεις του βρόχου I ανάμεσα στους 32 υπολογιστές, με τον ακόλουθο τρόπο: Ο υπολογιστής 1 εκτελεί τις 32 πρώτες επαναλήψεις (I=1 ως 32), ο υπολογιστής 2 εκτελεί τις επόμενες 32 επαναλήψεις, κ.ο.κ. Ποιός είναι ο συνολικός χρόνος εκτέλεσης και ποιός ο παράγοντας αύξησης της ταχύτητας σε σύγκριση με το ερώτημα (α); (Παρατηρείστε ότι το υπολογιστικό φορτίο, το οποίο καθορίζεται από το βρόχο J δεν σταθμίζεται ανάμεσα στους υπολογιστές).

γ. Να εξηγήσετε τον τρόπο αλλαγής της παραλληλοποίησης, ώστε να διευκολυνθεί η ισορροπημένη παράλληλη εκτέλεση ολόκληρου του υπολογιστικού φορτίου σε 32 υπολογιστές. Με τον όρο ισορροπημένο φορτίο εννούμε ότι σε κάθε υπολογιστή εκχωρείται ίσο πλήθος προσθέσεων όσον αφορά και τους δύο βρόχους.

δ. Ποιός είναι ο ελάχιστος χρόνος εκτέλεσης ο οποίος προκύπτει από την παράλληλη εκτέλεση σε 32 υπολογιστές; Ποιά είναι η αύξηση της ταχύτητας η οποία προκύπτει σε σχέση με την περίπτωση εκτέλεσης σε έναν απλό υπολογιστή;

17.16 Θεωρείστε τις δύο παρακάτω εκδοχές ενός προγράμματος πρόσθεσης δύο διανυσμάτων:

```
L1:      DO                          DOALL K = 1,M
L2:         A(I) = B(I) + C(I)          DO 10 I = L(K-1) + 1, KL
L3: 10   CONTINUE                        A(I) = B(I) + C(I)
L4:      SUM =0                  10      CONTINUE
L5:      DO 20 J = 1, N                  SUM(K) = 0
L6:         SUM = SUM +A(J)              DO 20 J = 1, L
L7: 20   CONTINUE                        SUM(K) =SUM(K) +A(L(K-1) +J)
                                 20      CONTINUE
                                      ENDALL
```

α. Το πρόγραμμα το οποίο βρίσκεται στο αριστερό μέρος, εκτελείται σε ένα μονοεπεξεργαστή. Έστω ότι κάθε γραμμή του κώδικα *L2*, *L4*, και *L6* απαιτεί έναν κύκλο ρολογιού του επεξεργαστή, για να ολοκληρώσει την εκτέλεσή του. Για λόγους απλούστευσης, δεν λαμβάνουμε υπόψη το χρόνο που απαιτούν οι άλλες γραμμές του κώδικα. Αρχικά, όλοι οι πίνακες είναι ήδη φορτωμένοι στην κύρια μνήμη και το σύντομο κομμάτι του προγράμματος βρίσκεται στην κρυφή μνήμη εντολών. Πόσοι κύκλοι ρολογιού απαιτούνται για να εκτελεστεί το πρόγραμμα;

β. Το πρόγραμμα το οποίο βρίσκεται στο δεξιό μέρος, είναι γραμμένο ώστε να εκτελείται σε έναν πολυεπεξεργαστή αποτελούμενο από M επεξεργαστές. Διαιρούμε τις λειτουργίες οι οποίες αφορούν βρόχους σε M ενότητες, με $L = N/M$ στοιχεία ανά ενότητα. Η εντολή

DOALL δηλώνει ότι οι M ενότητες εκτελούνται παράλληλα. Το αποτέλεσμα αυτού του προγράμματος είναι ότι παράγονται M μερικά αθροίσματα. Έστω ότι είναι αναγκαίοι k κύκλοι ρολογιού για την υλοποίηση κάθε επικοινωνίας μεταξύ των επεξεργαστών του συστήματος μέσω της κοινόχρηστης μνήμης, και επομένως, ότι η πρόσθεση κάθε μερικού αθροίσματος απαιτεί k κύκλους. Ένα δυαδικό δένδρο αθροιστή l επιπέδων, έχει τη δυνατότητα να συγχωνεύσει όλα τα μερικά αθροίσματα, όπου $l = \log_2 M$. Πόσοι κύκλοι είναι απαραίτητοι για να παραχθεί το τελικό άθροισμα;

γ. Έστω ότι $N = 2^{20}$ στοιχεία του πίνακα και ότι $M = 256$. Ποιά είναι η αύξηση της ταχύτητας αν χρησιμοποιηθεί ο πολυεπεξεργαστής; Υποθέστε ότι $k = 200$. Τι ποσοστό της θεωρητικής αύξησης κατά 256 αποτελεί αυτή η αύξηση;