Κεφάλαιο **10**

Παράλληλες Αρχιτεκτονικές

10.1 ΕΙΣΑΓΩΓΗ

Είναι γνωστό ότι οι αρχικοί υπολογιστές αναπτύχθηκαν και χρησιμοποιήθηκαν κυρίως για αριθμητικές εφαρμογές (επίλυση διαφορικών εξισώσεων κλπ.). Τα τεχνολογικά δεδομένα της εποχής εκείνης δεν επέτρεπαν τη χρήση των υπολογιστών ούτε για πιο απλές εφαρμογές (λόγω του υψηλού κόστους των τότε υπολογιστών), ούτε για πιο πολύπλοκες (λόγω της χαμηλής ταχύτητας εκτέλεσης). Η αλλαγή όμως των δεδομένων στην τεχνολογία των υπολογιστών επέτρεψε τόσο τη ραγδαία πτώση του κόστους του υλικού, όσο και τη σημαντική αύξηση της ταχύτητας των υπολογιστών. Αυτό είχε ως συνέπεια τη διεύρυνση του φάσματος των προβλημάτων που μπορούσαν να επιλυθούν από υπολογιστές. Αυτή η εξέλιξη επιταχύνθηκε με την εισαγωγή των μικροεπεξεργαστών και της τεχνολογίας VLSI. Επειδή όμως η πίεση για ακόμα ισχυρότερους υπολογιστές εξακολουθούσε να αυξάνει, οι σχεδιαστές υπολογιστικών συστημάτων έστρεψαν την προσοχή τους και σε ένα νέο πεδίο: την **παράλληλη επεξεργασία** (parallel processing).

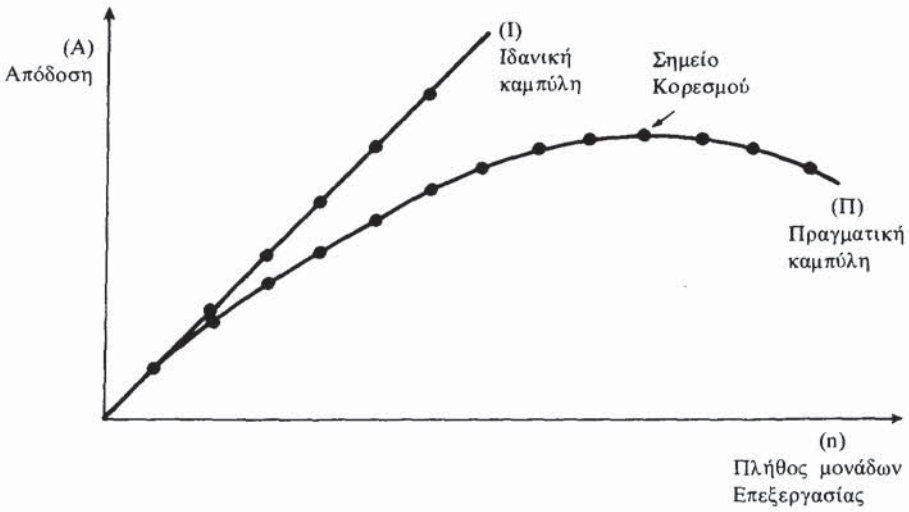
Μέχρι την προηγούμενη δεκαετία, τα υπολογιστικά συστήματα στήριζαν όλη τους την απόδοση στην ύπαρξη μιας ισχυρής ΚΜΕ που εκτελούσε ακολουθιακά τις εντολές του κάθε προγράμματος. Η μόνη δυνατότητα «παράλληλης επεξεργασίας» ήταν μέσω της ταυτόχρονης λειτουργίας της ΚΜΕ και των καναλιών Ε/Ε. Η εισαγωγή του παραλληλισμού στην εκτέλεση των εντολών με τη χρήση πολλών μονάδων επεξεργασίας, συνοδεύθηκε από την εμφάνιση αρκετών νέων προβλημάτων. Τα κυριότερα από αυτά αφορούν:

- Στον τρόπο επικοινωνίας και συγχρονισμού των μονάδων επεξεργασίας
- Στον έλεγχο των ταυτόχρονων αναφορών στην κοινή μνήμη, ή στις συσκευές Ε/Ε
- Στον τρόπο ανάθεσης των προγραμμάτων στις μονάδες επεξεργασίας
- Στον τρόπο περιγραφής των παράλληλων προγραμμάτων.

Για όλα αυτά τα προβλήματα έχουν διαμορφωθεί αρκετές απόψεις. Γενικά όμως, η σύγχρονη τάση είναι η ανάπτυξη συστημάτων που να χρησιμοποιούν, στο μέγιστο δυνατό βαθμό, τις δυνατότητες παράλληλης επεξεργασίας των αλγόριθμών. Αυτά τα συστήματα αναφέρονται συχνά και ως **υπερυπολογιστές** (supercomputers) ή ως **υπολογιστές μαζικού παραλληλισμού** (massively parallel computers).

Η απόδοση ενός υπολογιστικού συστήματος μπορεί να μετρηθεί με διάφορους δείκτες. Ο πλέον διαδεδομένος δείκτης απόδοσης είναι το πλήθος των εκτελούμενων πράξεων κινητής υποδιαστολής ανά δευτερόλεπτο, FLOPS (FLoating-point Operations Per Second). Ένας άλλος δείκτης απόδοσης είναι το πλήθος των εντολών που εκτελούνται στη μονάδα του χρόνου, όπως η μονάδα MIPS (Million Instructions Per Second), δηλαδή 10^6 εντολές ανά δευτερόλεπτο.

Στο Σχ. 10.1 φαίνεται ένα διάγραμμα (Ι) που παριστάνει την ιδανική μεταβολή της απόδοσης (Α) ενός υπολογιστή (π.χ. ταχύτητα εκτέλεσης



Σχ. 10.1. Η απόδοση ενός παράλληλου συστήματος ως συνάρτηση του αριθμού των μονάδων επεξεργασίας.

εντολών), σε συνάρτηση με τον αριθμό (n) των μονάδων επεξεργασίας που χρησιμοποιεί. Στην πραγματικότητα, όμως, η εισαγωγή νέων μονάδων επεξεργασίας συχνά προξενεί καθυστερήσεις στις άλλες μονάδες, με

αποτέλεσμα η συνολική απόδοση του συστήματος να μην αυξάνεται γραμμικά, αλλά σύμφωνα με τη μορφή της καμπύλης (Π) του Σχ. 10.1. Σε ορισμένα συστήματα παράλληλης επεξεργασίας μπορεί να παρατηρηθεί το φαινόμενο του κορεσμού (saturation), όπου η εισαγωγή επιπλέον μονάδων επεξεργασίας μειώνει, αντί να αυξάνει, τη συνολική απόδοση του συστήματος.

Στη συνέχεια αυτού του κεφαλαίου δίνονται οι αρχές των πιο σημαντικών τύπων παράλληλων υπολογιστών, που στηρίζονται στη χρήση συμβατικών επεξεργαστών αποθηκευμένου προγράμματος (Von Neumann) και αναλύονται μερικές αντιπροσωπευτικές μηχανές. Κατόπιν παρουσιάζονται οι σπουδαιότερες μη-συμβατικές (non-Von Neumann) αρχιτεκτονικές υπολογιστών και τέλος, αναλύεται η κατηγορία των παράλληλων αρχιτεκτονικών ειδικού σκοπού.

10.2 ΣΥΜΒΑΤΙΚΟΙ ΠΑΡΑΛΛΗΛΟΙ ΥΠΟΛΟΓΙΣΤΕΣ

Βασικό χαρακτηριστικό των συμβατικών παράλληλων υπολογιστών είναι ότι, γενικά, αποτελούνται από ένα σύνολο μονάδων επεξεργασίας, κάθε μία από τις οποίες λειτουργεί με βάση το κλασικό μοντέλο του Von Neumann, δηλαδή το μοντέλο του αποθηκευμένου προγράμματος. Στο μοντέλο αυτό οι εντολές εκτελούνται ακολουθιακά με τη βοήθεια του μετρητή προγράμματος. Έτσι, κάθε μονάδα επεξεργασίας εκτελεί μία ακολουθία εντολών πάνω σε μία ακολουθία δεδομένων. Ανάλογα με τον τρόπο παροχής των εντολών και των δεδομένων στις μονάδες επεξεργασίας, διακρίνονται τέσσερις κατηγορίες μηχανών, εκ των οποίων η μία είναι ακολουθιακή ενώ οι υπόλοιπες τρείς έχουν δυνατότητες παράλληλης επεξεργασίας. Αυτές είναι:

α) Μηχανές Μοναδικής Εντολής, Μοναδικών Δεδομένων: **ΜΕΜΔ** (Single Instruction Single Data, SISD). Στις μηχανές αυτές μια μονάδα επεξεργασίας εκτελεί ακολουθιακά τις εντολές (μία προς μία) πάνω σε μία ακολουθία δεδομένων. Πρόκειται για το κλασικό μοντέλο του Von Neumann, στο οποίο ανήκει και ο υπολογιστής TRN που περιγράφηκε στα προηγούμενα κεφάλαια.

β) Μηχανές Μοναδικής Εντολής, Πολλαπλών Δεδομένων: **ΜΕΠΔ** (Single Instruction Multiple Data, SIMD). Στις μηχανές αυτές οι μονάδες επεξεργασίας εκτελούν μία κοινή ακολουθία εντολών πάνω σε διαφορετικά δεδομένα. Οι μηχανές αυτής της κατηγορίας αναφέρονται και ως **μηχανές μητρώου** (array machines).

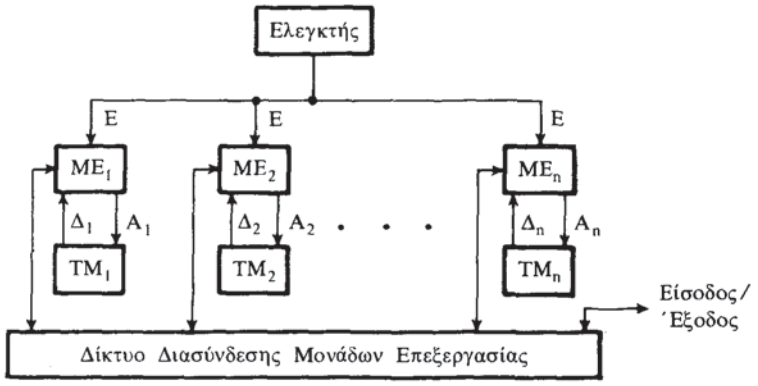
γ) Μηχανές Πολλαπλών Εντολών Μοναδικών Δεδομένων: **ΠΕΜΔ** (Multiple Instruction Single Data, MISD). Στις μηχανές αυτές οι μονάδες επεξεργασίας διατάσσονται σε μία αλυσιδωτή μορφή. Κάθε μονάδα επεξεργασίας εκτελεί ανεξάρτητες εντολές (λειτουργίες) ενώ τα δεδομένα μιας μονάδας επεξεργασίας είναι το αποτέλεσμα της προηγούμενης. Για το λόγο

αυτό, οι μηχανές αυτού του τύπου αναφέρονται συχνότατα ως **μηχανές αγωγού*** (pipeline machines).

δ) Μηχανές Πολλαπλών Εντολών, Πολλαπλών Δεδομένων: **ΠΕΠΔ** (Multiple Instruction Multiple Data, MIMD). Στις μηχανές αυτές, οι μονάδες επεξεργασίας είναι ελεύθερες να εκτελούν, ανεξάρτητα, οποιαδήποτε εντολή με οποιαδήποτε δεδομένα. Για το λόγο αυτό, οι μηχανές αυτού του τύπου αναφέρονται επίσης και ως **μηχανές πολυεπεξεργασίας** (multiprocessor machines).

10.2.1 Μηχανές Μητρώου

Ένα απλοποιημένο μοντέλο της αρχιτεκτονικής των μηχανών μητρώου με n μονάδες επεξεργασίας (ΜΕ) φαίνεται στο Σχ. 10.2. Η εντολή (Ε) που



Σχ. 10.2. Απλοποιημένο μοντέλο μηχανής μητρώου.

εκτελούν οι ΜΕ είναι κοινή και παρέχεται, σε αποκωδικοποιημένη μορφή, από τον ελεγκτή της μηχανής. Τα δεδομένα (Δ) σε κάθε ΜΕ είναι ανεξάρτητα και οδηγούν στον υπολογισμό των αποτελεσμάτων (Α). Οι ΜΕ έχουν συνήθως δυνατότητα τοπικής μνήμης (ΤΜ), τύπου RAM, όπου συνήθως αποθηκεύονται τα δεδομένα (Δ) και τα αποτελέσματα (Α) της εκτέλεσης των εντολών. Επίσης, συχνά υπάρχει η δυνατότητα επικοινωνίας μεταξύ των ΜΕ. Αυτό επιτυγχάνεται μέσω του δικτύου διασύνδεσης, που προσφέρει δυνατότητες επικοινωνίας μεταξύ των ΜΕ του συστήματος. Μπορεί, για παράδειγμα, οι ΜΕ να διαταχθούν ως ένας δισδιάστατος πίνακας, όπου κάθε ΜΕ έχει συνδέσμους επικοινωνίας με τις τέσσερις γειτονικές ΜΕ. Διατάξεις αυτής της μορφής μπορούν να χρησιμοποιηθούν για ειδικές εφαρμογές, όπως π.χ.για την παράλληλη επεξεργασία εικόνων. Στην τελευταία περίπτωση κάθε ΜΕ μπορεί να κρατάει και να επεξεργάζεται τα δεδομένα μιας περιοχής της εικόνας.

Σε οποιαδήποτε περίπτωση, είναι απαραίτητος ένας μηχανισμός εισό-

---

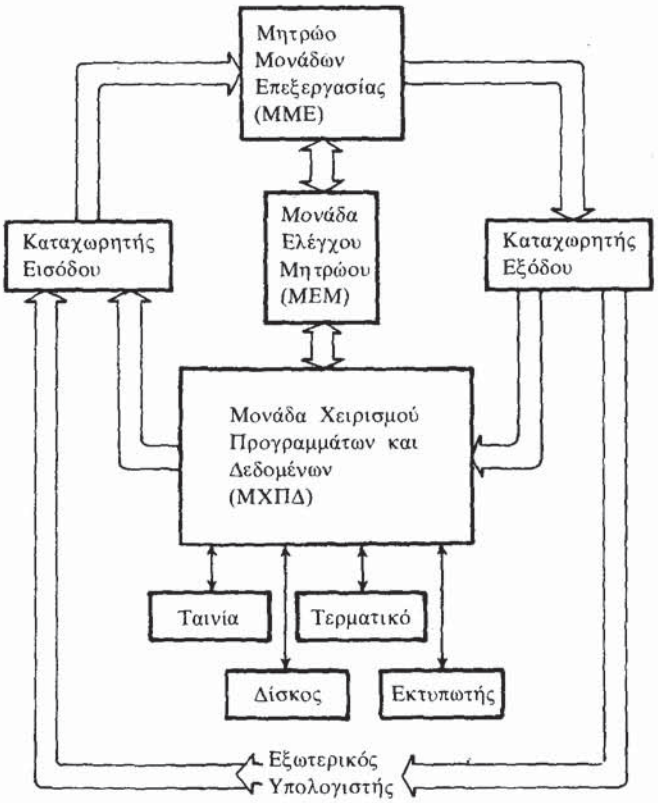
* Πολλές φορές η κατηγορία ΠΕΜΔ δεν χρησιμοποιείται και οι μηχανές αγωγού κατατάσσονται στην κατηγορία ΜΕΜΔ.

δου/εξόδου, που θα επιτρέπει τόσο τη φόρτωση των αρχικών δεδομένων στις ΜΕ, όσο και την έξοδο των τελικών αποτελεσμάτων.

Από τους σπουδαιότερους υπολογιστές μητρώου (ΜΕΠΔ) είναι ο ILLIAC-IV, ο BSP, ο DAP, ο STARAN, ο PEPE και ο MPP. Στη συνέχεια περιγράφεται ένας αντιπροσωπευτικός υπολογιστής τύπου ΜΕΠΔ, ο MPP.

Παράδειγμα: Ο Υπολογιστής MPP

Ο υπολογιστής MPP (Massively Parallel Processor) σχεδιάσθηκε κυρίως για την εξυπηρέτηση εφαρμογών επεξεργασίας εικόνας· για το λόγο αυτό χρησιμοποιεί μεγάλο αριθμό από ΜΕ (16.384), που είναι διατεταγμένες σε μορφή δισδιάστατου μητρώου διαστάσεων 128 × 128. Η γενική αρχιτεκτονική του συστήματος MPP φαίνεται στο Σχ. 10.3. Όλες οι ΜΕ συναποτε-



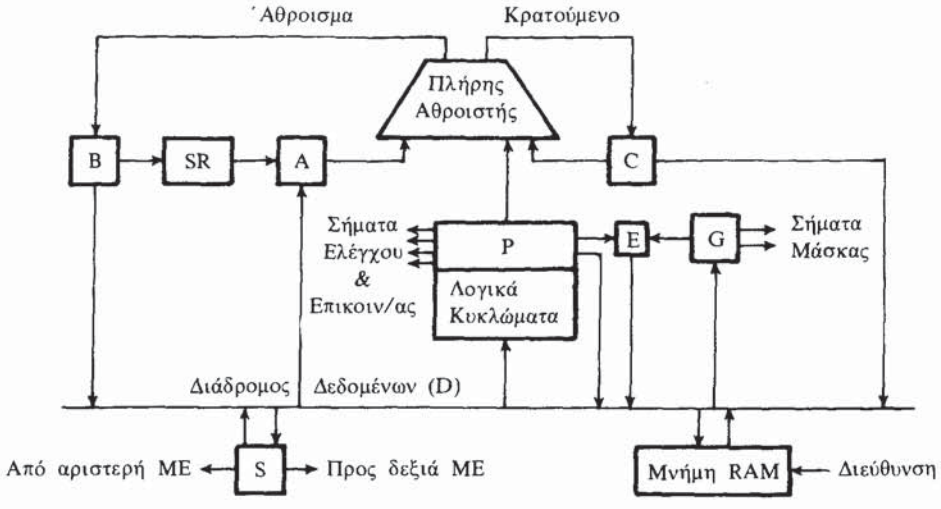
Σχ. 10.3. Γενική αρχιτεκτονική του συστήματος MPP.

λούν το μητρώο μονάδων επεξεργασίας (ΜΜΕ) και εκτελούν εντολές σύμφωνα με το μοντέλο ΜΕΠΔ. Για να υπάρχει δυνατότητα ανοχής βλαβών σε ΜΕ, το ΜΜΕ διαθέτει τέσσερις επιπλέον στήλες με εφεδρικές ΜΕ. Έτσι, οι φυσικές διαστάσεις του ΜΜΕ είναι 128 × 132 ΜΕ. Όταν παρουσιασθεί βλάβη σε κάποια στήλη του ΜΜΕ, τότε ειδικά κυκλώματα απομονώνουν

αυτήν τη στήλη, διατηρώντας έτσι τις λογικές διαστάσεις του ΜΜΕ σε 128 × 128 ΜΕ. Η Μονάδα Ελέγχου Μητρώου (ΜΕΜ) είναι ένας μικροπρογραμματιζόμενος υπολογιστής που εκτελεί τα προγράμματα των εφαρμογών και ελέγχει τη λειτουργία του όλου συστήματος. Η Μονάδα Χειρισμού Προγραμμάτων και Δεδομένων (ΜΧΠΔ) είναι ένας βοηθητικός μίνι-υπολογιστής, που ελέγχει τη ροή των δεδομένων στο σύστημα, φορτώνει τα δεδομένα στη ΜΕΜ, εκτελεί ορισμένες διαγνωστικές ρουτίνες και τέλος, χρησιμοποιείται ως μέσο για την ανάπτυξη προγραμμάτων εφαρμογής. Ο καταχωρητής εισόδου και ο καταχωρητής εξόδου χρησιμοποιούνται για την είσοδο και έξοδο δεδομένων στο ΜΜΕ. Τα Δεδομένα εισάγονται στις ΜΕ της αριστερότερης στήλης του ΜΜΕ και εξάγονται από τις ΜΕ της δεξιότερης στήλης. Οι καταχωρητές εισόδου και εξόδου διαθέτουν 128 bits, ένα για κάθε ΜΕ. Οι περιφερειακές συσκευές περιλαμβάνουν μονάδες μαγνητικής ταινίας και δίσκου, έναν εκτυπωτή και ένα τερματικό.

Οι ΜΕ είναι απλοί επεξεργαστές τύπου ψηφιοφέτας (bit-slice), για την εκτέλεση αριθμητικών πράξεων μεταβλητού μήκους. Κάθε ΜΕ του ΜΜΕ επικοινωνεί με τις τέσσερις γειτονικές ΜΕ (βόρεια, νότια, ανατολική και δυτική). Οι ΜΕ που βρίσκονται στις τέσσερις πλευρές του ΜΜΕ μπορεί είτε να είναι ανοικτές είτε να συνδέονται μεταξύ τους (η επάνω πλευρά με την κάτω και η δεξιά με την αριστερή). Επίσης, μπορεί οι γραμμές του ΜΜΕ να δημιουργούν μία ελικοειδή διάταξη, όπου η δεξιά ΜΕ μιας γραμμής συνδέεται με την αριστερή ΜΕ της επόμενης γραμμής. Η έλικα μπορεί να είναι είτε κλειστή είτε ανοικτή, αναλόγως με το εάν συνδέεται η αριστερή ΜΕ της πρώτης γραμμής με τη δεξιά ΜΕ της τελευταίας. Όλες αυτές οι τοπολογίες είναι προγραμματιζόμενες και καθορίζονται από ειδικό καταχωρητή της ΜΕΜ.

Η αρχιτεκτονική μιας ΜΕ φαίνεται στο Σχ. 10.4. Οι καταχωρητές A, B,



Σχ. 10.4. Η αρχιτεκτονική της ΜΕ του MPP.

C, P, G και S αποθηκεύουν πληροφορίες του ενός bit ενώ ο πλήρης αθροιστής εκτελεί αριθμητικές πράξεις με σειριακό τρόπο (ένα bit την κάθε φορά). Ο καταχωρητής ολίσθησης SR μπορεί να αποθηκεύει δεδομένα των 2, 6, 10, 14, 18, 22, 26 ή 30 bits. Ο καταχωρητής G χρησιμοποιείται για την αποθήκευση ενός bit μάσκας, για τον έλεγχο της λειτουργίας της ΜΕ. Το κύκλωμα Ε στέλνει στην έξοδό του το περιεχόμενο του P μόνον όταν ο G περιέχει το "1". Για την πράξη της πρόσθεσης, τα bits του πρώτου ορίσματος προωθούνται ακολουθιακά, μέσω του καταχωρητή SR, στον καταχωρητή A. Τα αντίστοιχα bits του άλλου ορίσματος προωθούνται, μέσω του διαδρόμου D, στον καταχωρητή P. Ο πλήρης αθροιστής υπολογίζει κάθε φορά το άθροισμα και το κρατούμενο που προκύπτει από την πρόσθεση των δυαδικών ψηφίων του A, του P και του C. Το bit του αθροίσματος οδηγείται στον καταχωρητή B, ενώ το bit του κρατούμενου στον καταχωρητή C, για να χρησιμοποιηθεί στο επόμενο βήμα της πρόσθεσης. Με τον τρόπο αυτό το τελικό αποτέλεσμα του αθροίσματος δύο αριθμών μπορεί να συσσωρευθεί στον καταχωρητή ολισθήσεως SR. Η πράξη της αφαίρεσης υλοποιείται κατά ανάλογο τρόπο, με τη χρησιμοποίηση αριθμητικής συμπληρώματος του δύο. Οι πράξεις του πολλαπλασιασμού και της διαίρεσης μπορούν να υλοποιηθούν με βάση την πράξη της πρόσθεσης (ή της αφαίρεσης).

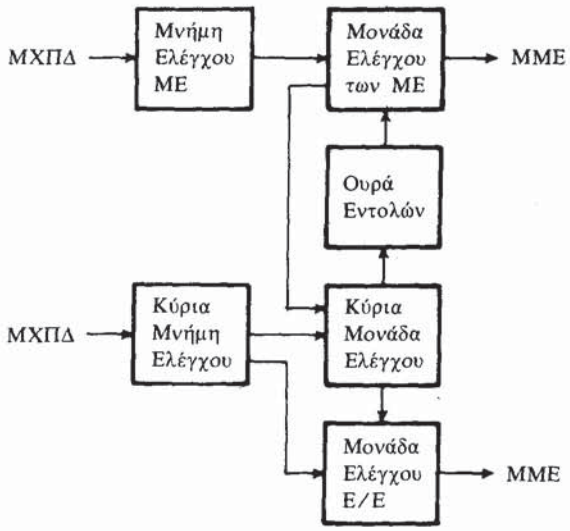
Ο καταχωρητής S χρησιμοποιείται για την είσοδο (ή έξοδο) δεδομένων από (ή προς) την αριστερή (ή δεξιά) ΜΕ. Κάθε ΜΕ περιέχει και κυκλώματα μνήμης RAM διαστάσεων 1024 × 1 bit. Η προσπέλαση στα δεδομένα της RAM γίνεται με βάση τη διεύθυνση που προσδιορίζεται από τη μονάδα ελέγχου μητρώου (MEM). Όπως μπορεί να φανεί και από το Σχ. 10.4, η αρχιτεκτονική της ΜΕ επιτρέπει πολλές ταυτόχρονες λειτουργίες· για παράδειγμα, είναι δυνατόν, ενώ γίνεται ανάγνωση από τη μνήμη RAM να γίνεται ταυτοχρόνως και μεταφορά δεδομένων από την αριστερή ΜΕ στον καταχωρητή S.

Η αρχιτεκτονική της Μονάδας Ελέγχου Μητρώου (MEM) φαίνεται στο Σχ. 10.5· αποτελείται από τρία βασικά τμήματα που λειτουργούν ταυτοχρόνως:

- Την κύρια μονάδα ελέγχου
- Τη μονάδα ελέγχου των ΜΕ
- Τη μονάδα ελέγχου Ε/Ε.

Η κύρια μονάδα ελέγχου εκτελεί εντολές που βρίσκονται στην κύρια μνήμη ελέγχου. Οι εντολές που αναφέρονται σε **βαθμωτές** (scalar) λειτουργίες εκτελούνται από την ίδια τη MEM. Βαθμωτές λειτουργίες είναι εκείνες οι λειτουργίες που παράγουν ως αποτέλεσμα έναν απλό αριθμό, σε αντιδιαστολή με τις **διανυσματικές** (vector) λειτουργίες που παράγουν ως αποτέλεσμα ένα σύνολο από αριθμούς (διάνυσμα, vector). Οι εντολές που αναφέρονται σε διανυσματικές λειτουργίες προωθούνται από την κύρια μονάδα ελέγχου, μέσω της ουράς εντολών, προς τη μονάδα ελέγχου των ΜΕ. Η μονάδα αυτή εκτελεί όλες τις διανυσματικές εντολές (δηλαδή τις εντολές που αναφέρονται στις ΜΕ του ΜΜΕ). Η εκτέλεση μιας διανυσματικής

εντολής συνίσταται στην παραγωγή της απαραίτητης ακολουθίας σημάτων ελέγχου προς τις ΜΕ. Η μνήμη ελέγχου των ΜΕ περιέχει ορισμένες προκαθορισμένες ρουτίνες, για την εκτέλεση ολοκληρωμένων λειτουργιών επί των δεδομένων που βρίσκονται αποθηκευμένα στις ΜΕ του ΜΜΕ. Η κύρια μονάδα ελέγχου της ΜΕΜ οδηγεί επίσης και τη μονάδα ελέγχου Ε/Ε, η οποία ελέγχει τη μεταφορά των δεδομένων από και προς το ΜΜΕ.



Σχ. 10.5. Διάγραμμα ενοτήτων της Μονάδας Ελέγχου Μητρώου (ΜΕΜ).

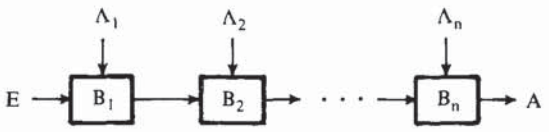
Είναι φανερό ότι τα τρία προαναφερθέντα τμήματα της ΜΕΜ μπορούν να λειτουργούν ταυτοχρόνως, συμβάλλοντας έτσι στην αποτελεσματική λειτουργία του όλου συστήματος.

Όπως είναι αναμενόμενο, η μηχανή MPP παρουσιάζει εξαιρετικές επιδόσεις κατά την εκτέλεση πράξεων με πίνακες (arrays). Με τη χρήση ρολογιού των 10 MHz για τις ΜΕ, μπορούν να εκτελεσθούν μέχρι 6.553×10^6 προσθέσεις το δευτερόλεπτο, για ακέραιους αριθμούς των 8 bits. Κατά την πράξη του εσωτερικού γινομένου δύο πινάκων, είναι δυνατόν να εκτελεσθούν μέχρι 1.861×10^6 πολλαπλασιασμοί το δευτερόλεπτο, με ακεραίους των 8 bits που παράγουν γινόμενα των 16 bits. Τέλος, κατά την εκτέλεση του πολλαπλασιασμού των στοιχείων ενός πίνακα επί έναν ακέραιο αριθμό, είναι δυνατόν να εκτελεσθούν μέχρι 2.340×10^6 πολλαπλασιασμοί το δευτερόλεπτο.

Συνοψίζοντας παρατηρούμε ότι η μηχανή MPP διευκολύνει την ταχεία εκτέλεση αλγορίθμων για τους οποίους απαιτούνται πολλαπλές αριθμητικές πράξεις επί πινάκων μεγάλων διαστάσεων. Αντίθετα, για αλγορίθμους στους οποίους απαιτούνται, ως επί το πλείστον, βαθμωτές λειτουργίες, η μηχανή MPP δεν είναι η πλέον ενδεδειγμένη, επειδή δεν εξασφαλίζει μεγάλο βαθμό παραλληλισμού.

10.2.2 Μηχανές Αγωγού

Στις μηχανές αγωγού (pipeline machines) κάθε κύκλος εκτέλεσης εντολής διαιρείται σε έναν αριθμό από διαδοχικές φάσεις, που ονομάζονται **κύκλοι αγωγού** (pipeline cycles), κάθε μία από τις οποίες εκτελείται σε μια διαφορετική βαθμίδα (stage) της ΚΜΕ. Στο Σχ. 10.6 φαίνεται ένα απλοποιη-



Σχ. 10.6. Απλοποιημένο διάγραμμα αγωγού.

μένο μοντέλο των μηχανών αγωγού. Κάθε βαθμίδα B_i του αγωγού εκτελεί μία συγκεκριμένη λειτουργία Λ_i στα δεδομένα που παράγονται από την προηγούμενη βαθμίδα B_{i-1}. Το αποτέλεσμα της B_i χρησιμοποιείται ως δεδομένο για την επόμενη βαθμίδα B_{i+1} του αγωγού. Οι εντολές Ε (ή τα δεδομένα) εισάγονται στον αγωγό μέσω της πρώτης βαθμίδας (B_1), ενώ τα τυχόν αποτελέσματα Α λαμβάνονται από την τελευταία βαθμίδα B_n.

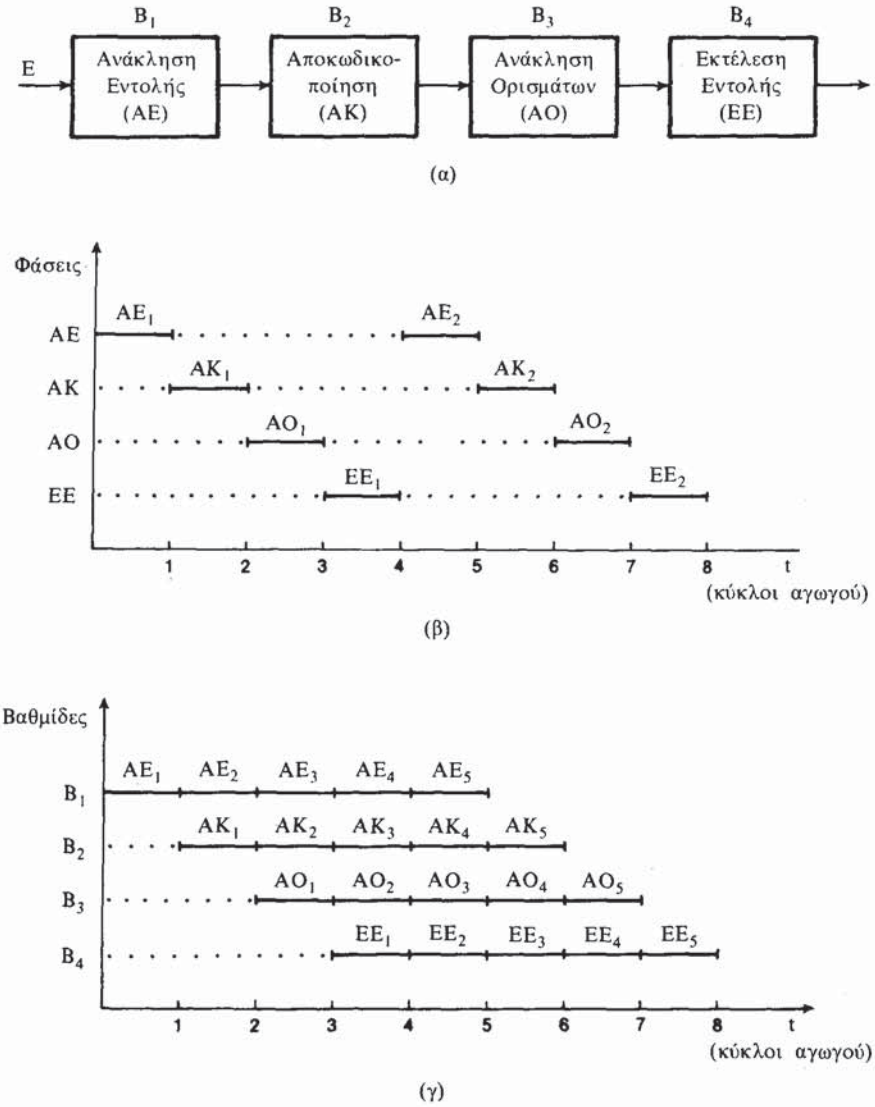
Εάν δεν υπήρχε η δυνατότητα των πολλαπλών βαθμίδων, τότε οι εντολές θα εκτελούνταν με ρυθμό 1 εντολή/κύκλο εντολής. Στην περίπτωση του αγωγού, ο ρυθμός εκτέλεσης των εντολών είναι 1 εντολή/κύκλο αγωγού (με την προϋπόθεση ότι η καθυστέρηση όλων των βαθμίδων είναι σταθερή). Ο κύκλος αγωγού είναι ο χρόνος που απαιτείται για την ολοκλήρωση της λειτουργίας της κάθε βαθμίδας και ασφαλώς είναι πολύ μικρότερος του κύκλου εντολής.

Στο Σχ. 10.7 (α) φαίνεται η διάταξη ενός αγωγού τεσσάρων βαθμίδων για την εκτέλεση των τεσσάρων φάσεων (κύκλων) μιας εντολής, ως εξής:

- Φάση ανάκλησης εντολής (ΑΕ)
- Φάση αποκωδικοποίησης εντολής (ΑΚ)
- Φάση ανάκλησης ορίσματος (ΑΟ)
- Φάση εκτέλεσης εντολής (ΕΕ).

Έστω ότι οι τέσσερις αυτές φάσεις απαιτούν χρόνο ίσο με τον κύκλο αγωγού, οπότε ο κύκλος εντολής είναι ίσος με τέσσερις κύκλους αγωγού. Στο Σχ. 10.7 (β) φαίνεται ένα διάγραμμα χρονισμού για την εκτέλεση τριών εντολών χωρίς τη χρήση αρχιτεκτονικής αγωγού. Έτσι, σε κάθε κύκλο εντολής (που είναι ίσος με τέσσερις κύκλους αγωγού) ολοκληρώνεται μία μόνον εντολή. Αντίθετα, στην περίπτωση αγωγού τεσσάρων βαθμίδων υπάρχει η δυνατότητα ταυτόχρονης εκτέλεσης των τεσσάρων διαδοχικών φάσεων των εντολών. Με τον τρόπο αυτό, όπως φαίνεται στο Σχ. 10.7 (γ), επιτυγχάνεται χρονική επικάλυψη των διαφόρων φάσεων, με αποτέλεσμα, μετά τη χρονική στιγμή $t=4$, οι εντολές να ολοκληρώνονται με ρυθμό 1 εντολή/κύκλο αγωγού. Οι αγωγοί που χρησιμοποιούνται για την επεξερ-

γασία των διαφόρων φάσεων των εντολών (Σχ. 10.7) αναφέρονται ως **αγωγοί εντολών** (instruction pipelines).

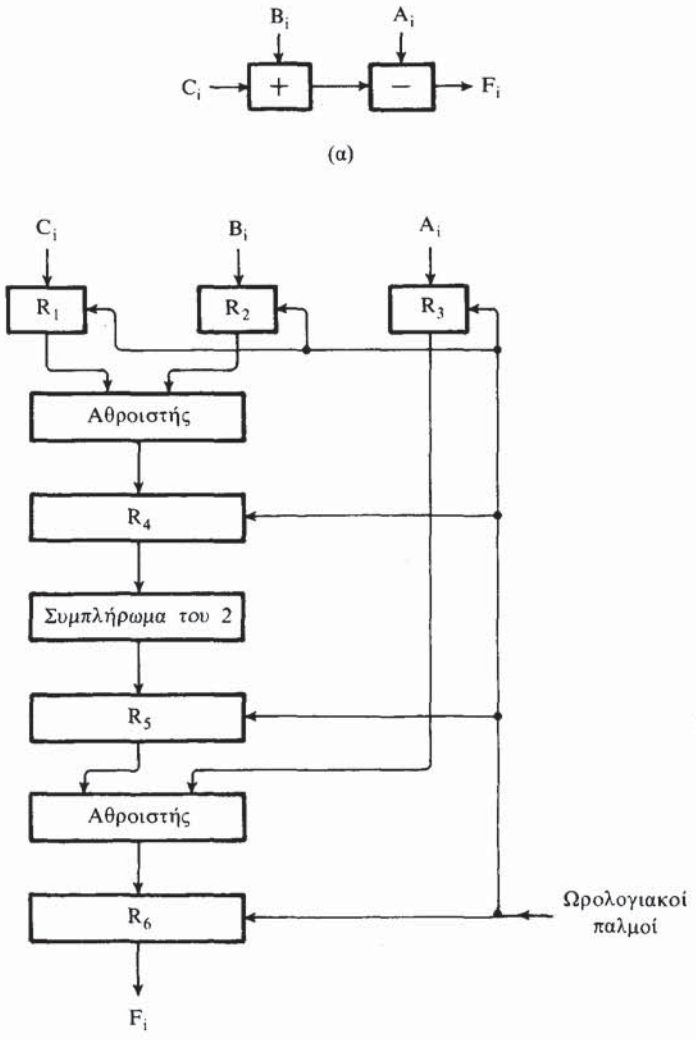


Σχ. 10.7. Οι τέσσερις φάσεις εκτέλεσης μιας εντολής.

Αγωγοί που χρησιμοποιούνται για τον κατά φάσεις υπολογισμό αριθμητικών εκφράσεων αναφέρονται ως **αριθμητικοί αγωγοί** (arithmetic pipelines). Ένας αγωγός αυτού του τύπου, για τον υπολογισμό της έκφρασης

$$F_i = A_i - (B_i + C_i)$$

φαίνεται στο Σχ. 10.8 (α). Μία φυσική υλοποίηση αυτού του αγωγού φαίνεται στο Σχ. 10.8 (β), όπου χρησιμοποιούνται 6 καταχωρητές ($R_1 - R_6$)



Σχ. 10.8. *Παράδειγμα αριθμητικού αγωγού· λογική περιγραφή (α) και φυσική υλοποίηση (β).*

για την αποθήκευση ενδιάμεσων αποτελεσμάτων. Στον Πίνακα 10.1 δίνονται τα περιεχόμενα των καταχωρητών $R_1 - R_6$ σε συνάρτηση με τους ωρολογιακούς παλμούς, για την εκτέλεση της εντολής:

```
FOR     I:=1 TO 4 DO
        F[I]:=A[I]-(B[I]+C[I]);
```

ΠΙΝΑΚΑΣ 10.1: Η λειτουργία του αγωγού του Σχ. 10.8 για 4 σειρές δεδομένων.

α/α Ωρολογιακού παλμού	R_1	R_2	R_3	R_4	R_5	R_6
1	C_1	B_1	—	—	—	—
2	C_2	B_2	—	B_1+C_1	—	—
3	C_3	B_3	A_1	B_2+C_2	$-(B_1+C_1)$	—
4	C_4	B_4	A_2	B_3+C_3	$-(B_2+C_2)$	F_1
5	—	—	A_3	B_4+C_4	$-(B_3+C_3)$	F_2
6	—	—	A_4	—	$-(B_4+C_4)$	F_3
7	—	—	—	—	—	F_4

Τέτοιου είδους εντολές, που περικλείουν την επαναληπτική εκτέλεση πολλών ταυτόσημων και ανεξάρτητων πράξεων, αναφέρονται ως **διανυσματικές εντολές** (vector instructions). Κάθε διανυσματική εντολή μπορεί να εκτελεσθεί παράλληλα σε έναν αριθμητικό αγωγό. Ωστόσο, μετά την ολοκλήρωση μιας διανυσματικής εντολής,ο αριθμητικός αγωγός πρέπει να αναδιατάσσεται, έτσι ώστε να αποκτά τη μορφή που απαιτείται για την εκτέλεση της επόμενης διανυσματικής εντολής.

Ένας αγωγός με n βαθμίδες μπορεί να επιταχύνει την εκτέλεση των προγραμμάτων μέχρι n φορές. Στην πραγματικότητα, όμως, η βέλτιστη αυτή επίδοση επιτυγχάνεται σπανίως, επειδή οι καθυστερήσεις των επί μέρους βαθμίδων δεν είναι ίσες μεταξύ τους. Ένας άλλος παράγοντας που οδηγεί σε καθυστερήσεις του αγωγού εντολών, είναι η ύπαρξη εντολών «άλματος υπό συνθήκη» (conditional jump), όπου η επόμενη εκτελέσιμη εντολή προσδιορίζεται μόνον μετά την ολοκλήρωση της εκτέλεσης της εν λόγω εντολής. Στην περίπτωση όπου η συνθήκη του άλματος ικανοποιείται, ο αγωγός πρέπει να «εκκενωθεί» από τις άλλες εντολές και να αρχίσει να εκτελεί εντολές από εκείνη τη θέση μνήμης η οποία ορίζεται στην εντολή «άλματος υπό συνθήκη». Άλλοι παράγοντες που επιβραδύνουν τη λειτουργία ενός αγωγού είναι οι ταυτόχρονες αναφορές στη μνήμη καθώς και οι διάφορες εξαρτήσεις των δεδομένων (όταν π.χ. μία εντολή πρόκειται να χρησιμοποιήσει το αποτέλεσμα της προηγούμενής της). Για τον περιορισμό της πιθανότητας «συγκρούσεων» στη μνήμη, χρησιμοποιείται συχνά η τεχνική της διαφύλλωσης της μνήμης (memory interleaving), που περιγράφεται στην § 8.6.3.

Οι μηχανές αγωγού είναι σχετικά απλές στη σύλληψη και στην κατασκευή. Όμως, η επίτευξη υψηλού ρυθμού απόδοσης εξαρτάται καθοριστικά από την ύπαρξη ή όχι αρκετών διανυσματικών εντολών. Η αναγνώριση διανυσματικών εντολών σε προγράμματα γλωσσών υψηλού επιπέδου γίνεται από ειδικούς μεταγλωττιστές (compilers).

Από τους σημαντικότερους σύγχρονους υπολογιστές που χρησιμο-

ποιούν διατάξεις αγωγού για την επιτάχυνση της εκτέλεσης των προγραμμάτων, είναι ο CDC Cyber 205, ο Fujitsu VP-200 και ο Cray-1. Στη συνέχεια θα εξετάσουμε τα σπουδαιότερα αρχιτεκτονικά χαρακτηριστικά του Cray-1.

Παράδειγμα: Ο υπολογιστής Cray-1

Ο Cray-1 είναι ένας από τους πλέον εξελιγμένους σύγχρονους υπολογιστές που χρησιμοποιεί 12 διατάξεις αγωγών, για την επίτευξη υψηλών ταχυτήτων εκτέλεσης (μέχρι 160×10^6 FLOPS). Ο υπολογιστής αυτός χρησιμοποιήθηκε ως βάση για την ανάπτυξη ενός ακόμη ισχυρότερου υπολογιστή, του Cray X-MP.

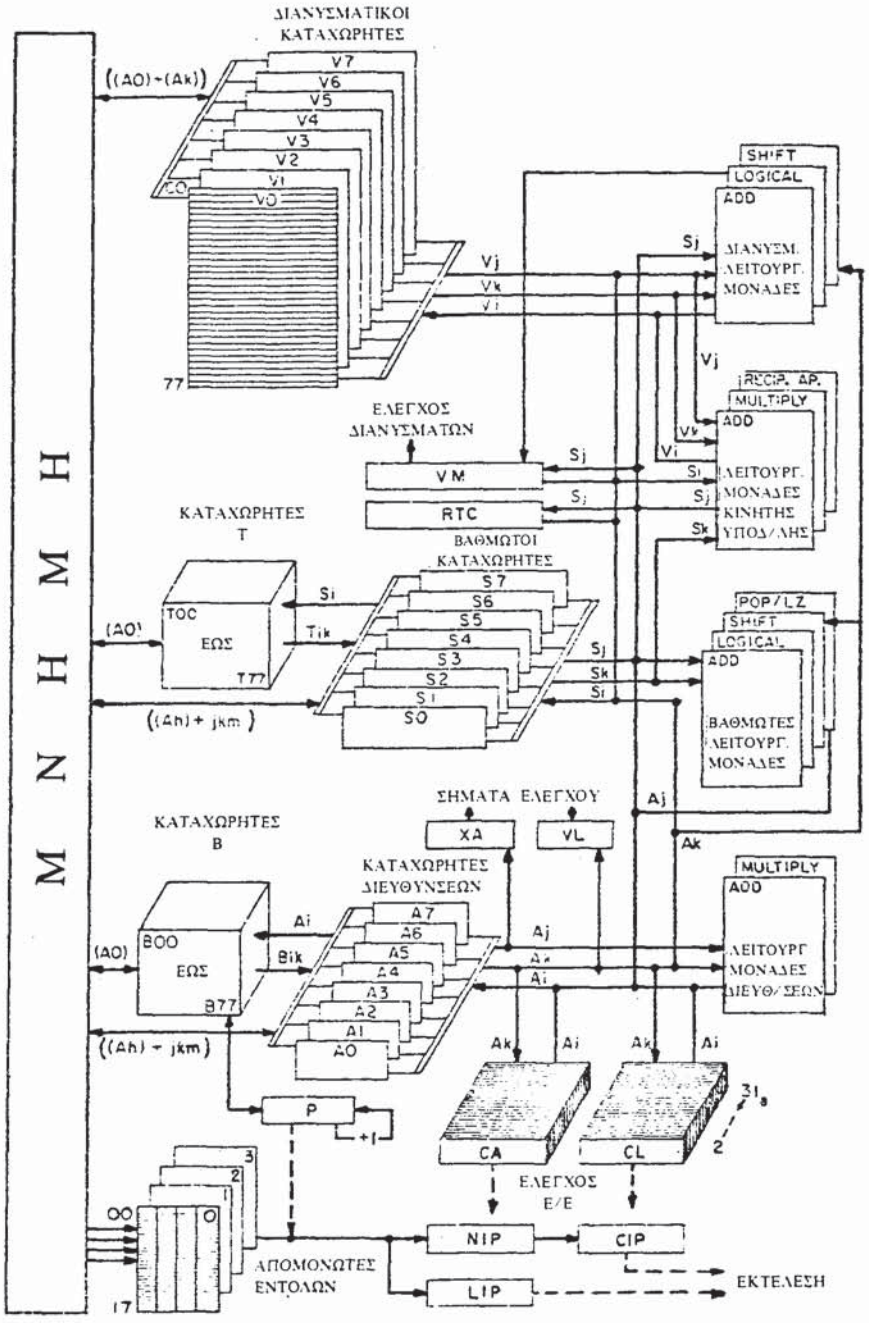
Ο Cray-1, ως σύστημα, αποτελείται από τρία τμήματα:

- Το **τμήμα επεξεργασίας**, που περιλαμβάνει τους καταχωρητές, τους αγωγούς (που ονομάζονται λειτουργικές μονάδες) και τους απομονωτές εντολών.
- Το **τμήμα μνήμης**, που περιλαμβάνει 0,25 M έως 1,0 M λέξεις των 64 bits. Για την επίτευξη γρήγορων αναφορών στη μνήμη χρησιμοποιήθηκαν κυκλώματα διπολικής (bipolar) τεχνολογίας.
- Το **τμήμα εισόδου/εξόδου**, που διαθέτει 12 κανάλια εισόδου και 12 κανάλια εξόδου. Τα κανάλια αυτά συνδέονται με συσκευές περιφερειακής μνήμης καθώς και με άλλους συμβατικούς υπολογιστές. Οι ταχύτητες μεταφοράς δεδομένων μέσω των καναλιών μπορεί να είναι μέχρι 80 Mbytes/sec.

Ένα διάγραμμα του τμήματος επεξεργασίας του Cray-1 φαίνεται στο Σχ. 10.9. Οι εντολές των προς εκτέλεση προγραμμάτων διαβάζονται σε τέσσερις ειδικούς απομονωτές εντολών (instruction buffers), κάθε ένας από τους οποίους περιέχει 64 εντολές. Ο καταχωρητής P (22 bits) είναι ο μετρητής προγράμματος του συστήματος και υποδεικνύει τον επόμενο κώδικα εντολής που πρέπει να φορτωθεί στον καταχωρητή επόμενης εντολής (NIP). Ο καταχωρητής NIP (16 bits) κρατά τον κώδικα εντολής πριν αυτός φορτωθεί στον καταχωρητή παρούσας εντολής (CIP). Ο CIP (16 bits) κρατά τον κώδικα της εντολής που πρόκειται να εκτελεσθεί από τη μηχανή. Εάν ο κώδικας της εντολής είναι 32 bits, τότε ο καταχωρητής CIP κρατάει τα 16 πιο σημαντικά bits της εντολής, ενώ ο καταχωρητής LIP κρατά τα υπόλοιπα 16 bits. Άλλοι καταχωρητές, όπως ο καταχωρητής μάσκας διανύσματος VM, ο καταχωρητής διεύθυνσης ανταλλαγής BA, ο καταχωρητής διεύθυνσης ορίου LA, ο καταχωρητής σημαίας F και ο καταχωρητής κατάστασης M, χρησιμοποιούνται για τον υπολογισμό τελικών διευθύνσεων στη μνήμη, για την προγραμματιζόμενη αδρανοποίηση τμημάτων του συστήματος και, εν γένει, για τον έλεγχο της εκτέλεσης των προγραμμάτων.

Στον Cray-1 υπάρχουν τρεις βασικοί τύποι καταχωρητών:

α) Οι καταχωρητές διευθύνσεων A (address registers)



Σχ. 10.9. Το τμήμα επεξεργασίας του Cray-1.
(Ανατυπώνεται με άδεια της Cray Research Inc.).

β) Οι βαθμωτοί καταχωρητές S (scalar registers), δηλαδή καταχωρητές που αποθηκεύουν ένα μόνο στοιχείο
γ) Οι διανυσματικοί καταχωρητές V (vector registers), δηλαδή καταχωρητές που ο καθένας τους αποθηκεύει πολλά στοιχεία.

Αναλυτικότερα, υπάρχουν 8 καταχωρητές A που καθένας τους αποθηκεύει διευθύνσεις των 24 bits. Χρησιμοποιούνται από τις εντολές του συστήματος κατά την αναφορά στη μνήμη, κατά τη δεικτοδότηση, κατά την αναφορά στα κανάλια E/E, καθώς και για τον έλεγχο των επαναλήψεων (loop control). Οι καταχωρητές A επικοινωνούν με την κύρια μνήμη είτε απ' ευθείας, είτε μέσω των 64 καταχωρητών B.

Οι 8 βαθμωτοί καταχωρητές S αποθηκεύουν δεδομένα των 64 bits. Οι καταχωρητές αυτοί χρησιμοποιούνται από τις εντολές του συστήματος ως σημεία προέλευσης ή προορισμού των ορισμάτων ή των αποτελεσμάτων τους. Τα περιεχόμενα των καταχωρητών S μεταφέρονται από (ή προς) την κύρια μνήμη είτε απ' ευθείας, είτε μέσω των 64 καταχωρητών T, που καθένας τους έχει 64 bits. Τόσο οι καταχωρητές T όσο και οι καταχωρητές B μεταφέρουν τα δεδομένα τους από ή προς τη μνήμη κατά ενότητες (blocks), χρησιμοποιώντας εντολές του τύπου «ανάγνωση ενότητας» ή «εγγραφή ενότητας». Έτσι μεγιστοποιείται η ταχύτητα μετάδοσης δεδομένων μεταξύ μνήμης και καταχωρητών T ή B. Εξάλλου, τα δεδομένα μπορούν να προσπελασθούν ταχύτερα από τους καταχωρητές T και B απ' ό,τι από τη μνήμη του συστήματος.

Οι διανυσματικοί καταχωρητές V χρησιμοποιούνται για την αποθήκευση διανυσματικών δεδομένων, δηλαδή δεδομένων που αποτελούνται από πολλά στοιχεία. Κάθε καταχωρητής V έχει 64 στοιχεία, κάθε ένα από τα οποία είναι ένα βαθμωτό δεδομένο των 64 bits. Οι εντολές που χρησιμοποιούν ως όρισμα κάποιον καταχωρητή V, χρησιμοποιούν διαδοχικά τα στοιχεία του καταχωρητή αυτού και δημιουργούν το αποτέλεσμα σε κάποιον άλλον καταχωρητή V. Η επαναληπτική αυτή λειτουργία συνεχίζεται μέχρις ότου ο αριθμός των εκτελεσθέντων λειτουργιών γίνει ίσος με το περιεχόμενο του καταχωρητή μήκους διανύσματος VL. Οι καταχωρητές V επικοινωνούν απ' ευθείας με τη μνήμη, η δε μεταφορά των δεδομένων γίνεται κατά ομάδες, με τη χρήση ειδικών εντολών.

Η πραγματική επεξεργασία των δεδομένων, ως αποτέλεσμα ανάλογων εντολών, γίνεται από τις 12 λειτουργικές μονάδες (functional units) του συστήματος. Κάθε λειτουργική μονάδα εκτελεί μια συγκεκριμένη λειτουργία και έχει την οργάνωση αγωγού πολλαπλών βαθμίδων. Όπως φαίνεται και στο Σχ. 10.9, οι 12 λειτουργικές μονάδες του Cray-1 χωρίζονται σε τέσσερις ομάδες που είναι:

α) Οι **λειτουργικές μονάδες διευθύνσεων** (address functional units), που υλοποιούν αριθμητικές λειτουργίες σε διευθύνσεις. Συγκεκριμένα, υπάρχει μία λειτουργική μονάδα για την πράξη της πρόσθεσης (Add) και μία άλλη για την πράξη του πολλαπλασιασμού (Multiply). Το αποτέλεσμα αυτών των λειτουργιών οδηγείται πάντα σε κάποιον καταχωρητή A.
β) Οι **λειτουργικές μονάδες βαθμωτών πράξεων** (scalar functional units), που υλοποιούν απλές αριθμητικές πράξεις με βαθμωτά ορίσματα. Συγκεκρι-

μένα, υπάρχει μία λειτουργική μονάδα για την πρόσθεση (Add), μία για την ολίσθηση (Shift), μία για λογικές (logical) πράξεις και μία για τη μέτρηση του αριθμού των "1" στο όρισμα ή για τη μέτρηση των "0", που ακολουθούνται από "1" στο όρισμα (Population-leading zero count). Τα αποτελέσματα αυτών των λειτουργικών μονάδων επιστρέφουν πάντα σε κάποιον καταχωρητή S, εκτός από την τελευταία λειτουργική μονάδα, που μεταφέρει το αποτέλεσμά της (7 bits) σε έναν καταχωρητή A.

γ) Οι **λειτουργικές μονάδες πράξεων κινητής υποδιαστολής** (floating point functional units), που υλοποιούν αριθμητικές πράξεις (βαθμωτές ή διανυσματικές), σε αριθμητικά δεδομένα που παριστάνονται στο σύστημα κινητής υποδιαστολής (KY). Συγκεκριμένα, υπάρχει η λειτουργική μονάδα της πρόσθεσης (Add), του πολλαπλασιασμού (Multiply) και της αντιστροφής (Reciprocal Approximation).

δ) Οι **λειτουργικές μονάδες διανυσματικών πράξεων** (vector functional units), που υλοποιούν διανυσματικές λειτουργίες σε αριθμητικά δεδομένα τα οποία προέρχονται από δύο καταχωρητές V ή από έναν καταχωρητή V και έναν καταχωρητή S. Τα αριθμητικά δεδομένα παριστάνονται σε μορφή συμπληρώματος του δύο. Ειδικότερα, υπάρχει μία λειτουργική μονάδα για διανυσματική πρόσθεση (Add), μία για λογικές λειτουργίες (Logical) και μία για λειτουργίες ολίσθησης (Shift). Σε όλες τις περιπτώσεις, το αποτέλεσμα μιας διανυσματικής λειτουργίας οδηγείται σε έναν καταχωρητή V.

Στον Πίνακα 10.2 δίνονται για κάθε λειτουργική μονάδα, οι καταχωρη-

ΠΙΝΑΚΑΣ 10.2 Χαρακτηριστικά των λειτουργικών μονάδων του Cray-1

Λειτουργικές Μονάδες	Χρησιμοποιούμενοι καταχωρητές	Καθυστέρηση (σε κύκλους ρολογιού)
Διευθύνσεων		
Πρόσθεσης	A	2
Πολλαπλασιασμού	A	6
Βαθμωτών πράξεων		
Πρόσθεσης	S	3
Ολίσθησης	S	2 ή 3
Λογικών πράξεων	S	1
Μετρήσεων	S	3
Πράξεων κινητής υποδιαστολής		
Πρόσθεσης	V, S	6
Πολλαπλασιασμού	V, S	7
Αντιστροφής	V, S	14
Διανυσματικών πράξεων		
Πρόσθεσης	V, S	3
Ολίσθησης	V, S	4
Λογικών πράξεων	V, S	2

τές που αυτή χρησιμοποιεί, καθώς και το «μήκος» του αντίστοιχου αγωγού, δηλαδή η καθυστέρηση που αυτός προκαλεί.

Πρέπει να σημειωθεί ότι κάθε λειτουργική μονάδα λειτουργεί ανεξάρτητα από τις άλλες. Έτσι, είναι δυνατόν πολλές λειτουργικές μονάδες να εργάζονται παράλληλα, με την προϋπόθεση βέβαια ότι δεν χρησιμοποιούν κοινούς καταχωρητές. Κάθε λειτουργική μονάδα (αγωγός) λαμβάνει τα ορίσματά της συνήθως από δύο καταχωρητές προέλευσης και στέλνει τα αποτελέσματά της σε κάποιον καταχωρητή προορισμού. Έτσι, οι λειτουργικές μονάδες χρησιμοποιούνται ως επί το πλείστον από εντολές των τριών διευθύνσεων. Συνολικά το ρεπερτόριο εντολών του Cray-1 αποτελείται από 128 εντολές με 10 τύπους διανυσματικών εντολών και 13 τύπους βαθμωτών εντολών.

Οι αναγκαίες προϋποθέσεις για να μπορούν δύο εντολές του Cray-1 να εκτελεσθούν παράλληλα είναι οι εξής:
α) Οι εντολές δεν απαιτούν χρήση της ίδιας λειτουργικής μονάδας και,
β) οι εντολές δεν χρησιμοποιούν κοινούς καταχωρητές.

Να σημειωθεί ότι στον Cray-1 δύο εντολές που διαβάζουν δεδομένα από έναν κοινό καταχωρητή δεν μπορούν να εκτελούνται παράλληλα. Αυτή η επιλογή έγινε για λόγους απλότητας των κυκλωμάτων ελέγχου του συστήματος.

Για να δώσουμε μία εικόνα του τρόπου προγραμματισμού και λειτουργίας του Cray-1, ας θεωρήσουμε ότι έχουμε για εκτέλεση την εντολή:

```
FOR I:=1 TO N DO
A[I]:=W*B[I]+Z+3;
```

Εάν υποθέσουμε ότι $N \leqslant 64$, και ότι όλα τα αριθμητικά δεδομένα (εκτός από τους ακέραιους I και N) είναι αριθμοί κινητής υποδιαστολής, τότε η εν λόγω εντολή Pascal μεταφράζεται (από ειδικό μεταγλωττιστή) στις εξής εντολές του Cray-1:

```
1: S1 ← W;   Αρχικοποίηση S1
2: S2 ← Z;   Αρχικοποίηση S2
3: S3 ← 3;   Αρχικοποίηση S3
4: VL ← N;   Αρχικοποίηση καταχωρητή VL
5: V0 ← B;   Φόρτωση διανύσματος B στον V0
6: S4 ← S2 + S3;   Υπολογισμός του Z + 3
7: V1 ← V0 * S1;   Πολλαπλασιασμός διανύσματος B με τη σταθερή τιμή W
8: V2 ← V1 + S4;   Υπολογισμός τελικού διανύσματος
9: A ← V2;   Επιστροφή της τιμής του A στη μνήμη.
```

Οι εντολές 1-5 είναι εντολές αναφοράς στη μνήμη, για την κατάλληλη αρχικοποίηση των αντίστοιχων καταχωρητών του Cray-1. Η εντολή 6 προσδιορίζει την πρόσθεση δύο βαθμωτών καταχωρητών και εκτελείται στη λειτουργική μονάδα πρόσθεσης κινητής υποδιαστολής. Οι εντολές 7 και 8 είναι διανυσματικές και εκτελούνται στις λειτουργικές μονάδες πολλαπλασιασμού και πρόσθεσης κινητής υποδιαστολής. Η εντολή 7 μπορεί

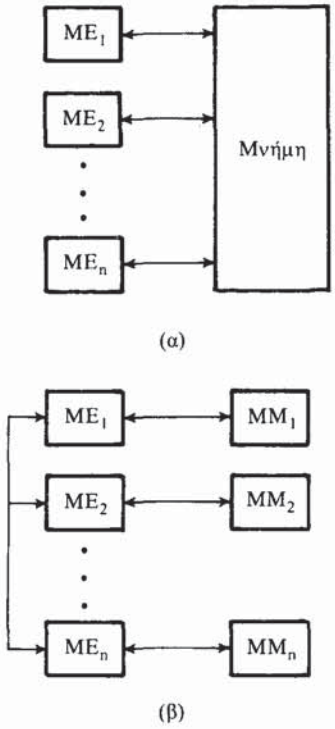
να εκτελεσθεί ταυτοχρόνως με την εντολή 6, επειδή και οι δύο χρησιμοποιούν διαφορετικούς καταχωρητές και διαφορετικές λειτουργικές μονάδες. Η εντολή 8, όμως, δεν μπορεί να εκτελεσθεί ταυτοχρόνως με την εντολή 6 επειδή χρησιμοποιεί την ίδια λειτουργική μονάδα. Επίσης, η εντολή 8 δεν μπορεί να εκτελεσθεί ταυτοχρόνως με την εντολή 7 επειδή ο χρησιμοποιούμενος από την εντολή 8 καταχωρητής V_1, λαμβάνει τιμή από την εντολή 7. Ειδικά για τέτοιες περιπτώσεις, όπου μία διανυσματική εντολή χρησιμοποιεί το διανυσματικό αποτέλεσμα της προηγούμενης, χρησιμοποιείται ένας ειδικός μηχανισμός εκτέλεσης, που αναφέρεται ως **αλυσιδοποίηση** (chaining). Στην περίπτωση αυτή, η εντολή 8 δεν περιμένει την ολοκλήρωση της εντολής 7 για να αρχίσει να εκτελείται. Αντίθετα, η εντολή 8 ξεκινά την εκτέλεσή της αμέσως μόλις η εντολή 7 παράξει το πρώτο στοιχείο του καταχωρητή V_1. Έτσι, επιτυγχάνεται μεγάλη χρονική επικάλυψη στην εκτέλεση των δύο αυτών εντολών, με αποτέλεσμα το πρόγραμμα να εκτελείται με μεγάλη ταχύτητα. Η τεχνική της αλυσιδοποίησης μπορεί να χρησιμοποιηθεί για το συνδυασμό μέχρι 5 εντολών του Cray-1. Η εντολή 9, τέλος, καταγράφει στη μνήμη την τελική τιμή του αποτελέσματος (A).

10.2.3 Μηχανές Πολυεπεξεργασίας

Βασικό χαρακτηριστικό των μηχανών πολυεπεξεργασίας (multiprocessor machines) είναι ότι οι χρησιμοποιούμενες μονάδες επεξεργασίας (ΜΕ) είναι αυτόνομες ΚΜΕ που έχουν τη δυνατότητα να εκτελούν ανεξάρτητα προγράμματα, χρησιμοποιώντας ανεξάρτητα, επίσης, δεδομένα. Επειδή στις μηχανές πολυεπεξεργασίας (Π/Ε) οι ΜΕ μπορούν να εκτελούν οποιαδήποτε εντολή με οποιαδήποτε δεδομένα, οι μηχανές αυτές αναφέρονται και ως μηχανές πολλαπλών Εντολών-Πολλαπλών Δεδομένων, ΠΕΠΔ (Multiple Instruction Multiple Data, MIMD).

Τα δεδομένα και οι εντολές των ΜΕ λαμβάνονται από την κεντρική μνήμη της μηχανής, που συνήθως αποτελείται από αρκετές ανεξάρτητες μονάδες μνήμης (ΜΜ). Όταν οι ΜΜ του συστήματος είναι απ' ευθείας προσπελάσιμες από οποιαδήποτε ΜΕ, όπως φαίνεται στο Σχ. 10.10 (α), η διάταξη αναφέρεται ως **στενά συνδεδεμένο σύστημα** (tightly coupled system). Αντίθετα, όταν κάθε ΜΜ είναι προσπελάσιμη από μία μόνο ΜΕ (Σχ. 10.10 (β)), η διάταξη αναφέρεται ως **χαλαρά συνδεδεμένο σύστημα** (loosely coupled system). Ωστόσο, τα πραγματικά συστήματα Π/Ε συνήθως δεν ανήκουν πλήρως ούτε στη μία ούτε στην άλλη κατηγορία. Έτσι, τα χαλαρά συνδεδεμένα συστήματα συνήθως διαθέτουν και ένα μικρό μέρος κοινής μνήμης, ενώ, στα στενά συνδεδεμένα συστήματα κάθε ΜΕ διαθέτει και μία τοπική μνήμη για την αποθήκευση προγραμμάτων ή δεδομένων που χρησιμοποιούνται πιο συχνά. Στην περίπτωση των στενά συνδεδεμένων συστημάτων οι ΜΕ μπορούν να ανταλλάσσουν πληροφορίες μέσω της κοινής μνήμης, ενώ στην περίπτωση των χαλαρά συνδεδεμένων συστημάτων η επικοινωνία μεταξύ των ΜΕ γίνεται μέσω ειδικών γραμμών που συνδέουν όλες τις ΜΕ.

Οι ΜΕ ενός συστήματος Π/Ε είναι ισοδύναμες και συχνά είναι όμοιοι επεξεργαστές, που ως σύστημα λειτουργούν παράλληλα με σκοπό τη διεκπεραίωση ενός γενικότερου υπολογιστικού προβλήματος. Κάθε σύστημα Π/Ε ελέγχεται κατά κανόνα από ένα μοναδικό λειτουργικό σύστημα.

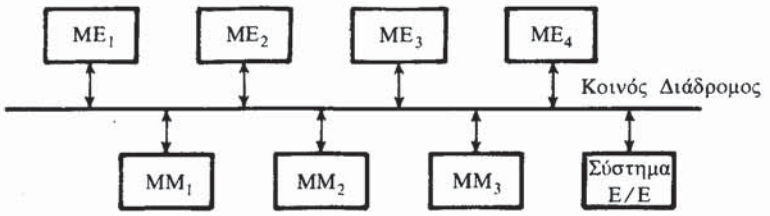


Σχ. 10.10. Στενά συνδεδεμένο σύστημα Π/Ε (α) και χαλαρά συνδεδεμένο σύστημα Π/Ε (β).

Συνήθως όλες οι ΜΕ ενός συστήματος Π/Ε βρίσκονται σε σχετικά μικρές αποστάσεις, έτσι ώστε να υπάρχει η δυνατότητα ταχείας ανταλλαγής δεδομένων μεταξύ των διαφόρων ΜΕ. Σε ορισμένες ειδικές εφαρμογές όμως, οι ΜΕ τοποθετούνται σε απομακρυσμένα σημεία και επικοινωνούν μεταξύ τους με σειριακό τρόπο, μέσω γραμμών επικοινωνίας. Τα συστήματα αυτά μπορούν να θεωρηθούν ως επέκταση των χαλαρά συνδεδεμένων συστημάτων Π/Ε και αναφέρονται συχνά ως **κατανεμημένα συστήματα** (distributed systems).

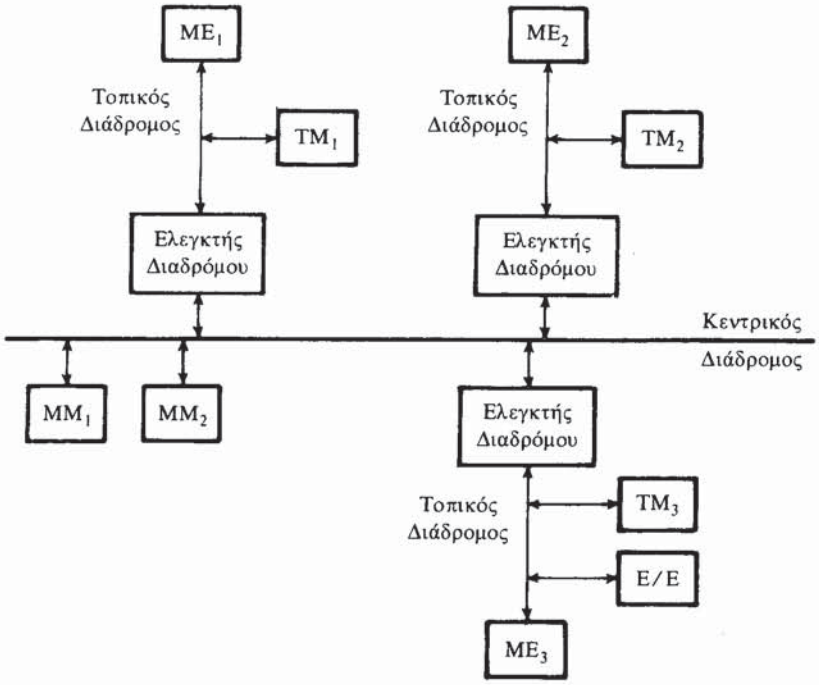
Τα στενά συνδεδεμένα συστήματα Π/Ε προσφέρουν σε όλες τις ΜΕ τη δυνατότητα απ' ευθείας αναφοράς στη μνήμη του συστήματος. Αυτό μπορεί να επιτευχθεί με διάφορους τρόπους. Ένας τρόπος είναι με τη χρήση ενός **κοινού διαδρόμου** (common bus), όπως φαίνεται στο Σχ. 10.11. Η υλοποίηση ενός τέτοιου συστήματος είναι σχετικά απλή. Ωστόσο, επειδή ο κοινός διάδρομος μπορεί να χρησιμοποιείται κάθε στιγμή από μία

μόνο ΜΕ, είναι πιθανό να παρουσιάζονται αρκετές «συγκρούσεις» στη χρήση του (διαδρόμου) από τις ΜΕ. Για το λόγο αυτό πρέπει να υπάρχει στο λογικό των ΜΕ ειδικός μηχανισμός για την επίλυση αυτού του προβλήματος. Αυτός ο μηχανισμός αναφέρεται ως διαιτησία διαδρόμου (bus arbitration) και εξετάζεται στην § 9.8.2.



Σχ. 10.11. Σύστημα Π/Ε με κοινό διάδρομο.

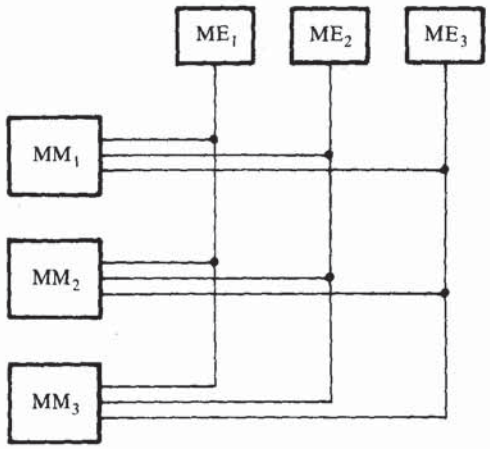
Τα συστήματα Π/Ε με κοινό διάδρομο δεν μπορούν να χρησιμοποιηθούν για απεριόριστο αριθμό ΜΕ, επειδή ο κοινός διάδρομος μπορεί να εξυπηρετήσει πεπερασμένο αριθμό προσπελάσεων στη μονάδα του χρόνου. Η σύνδεση υπερβολικού αριθμού ΜΕ οδηγεί στον κορεσμό του διαδρόμου, με αποτέλεσμα πολλές ΜΕ να παραμένουν αδρανείς αναμένοντας την



Σχ. 10.12. Σύστημα Π/Ε με πολλαπλούς διαδρόμους.

απελευθέρωση του κοινού διαδρόμου. Μια συνήθης τεχνική για τον περιορισμό των προσπελάσεων στην κοινή μνήμη είναι η χρησιμοποίηση πολλαπλών διαδρόμων (Σχ. 10.12), όπου κάθε ΜΕ διαθέτει μία τοπική μνήμη (ΤΜ), στην οποία αποθηκεύονται οι συχνότερα χρησιμοποιούμενες ρουτίνες και δεδομένα.

Ένας άλλος τρόπος υλοποίησης των στενά συνδεδεμένων συστημάτων Π/Ε είναι η χρήση μνημών πολλαπλών εισόδων (multiport memories), όπως φαίνεται στο Σχ. 10.13. Στα συστήματα αυτά οι μονάδες μνήμης (ΜΜ) διαθέτουν πολλές εισόδους, κάθε μία από τις οποίες συνδέεται αποκλειστικά με μία ΜΕ. Έτσι, είναι δυνατόν δύο ΜΕ να εκτελούν ταυτοχρόνως εντολές ανάγνωσης (ή εγγραφής) στη μνήμη. Για την περίπτωση όμως αναφοράς στην ίδια ΜΜ υπάρχει ειδικός μηχανισμός στο υλικό των ΜΜ, για τη διαδοχική εξυπηρέτηση οποιωνδήποτε ταυτόχρονων αιτήσεων. Η σειρά εξυπηρέτησης των ταυτόχρονων αναφορών στην ίδια ΜΜ προσδιορίζεται συνήθως από τις σχετικές προτεραιότητες που έχουν οι διάφορες είσοδοι της κάθε ΜΜ. Έτσι, οι ΜΕ δεν ασχολούνται καθόλου με το πρόβλημα της επίλυσης των «συγκρούσεων» στη μνήμη.

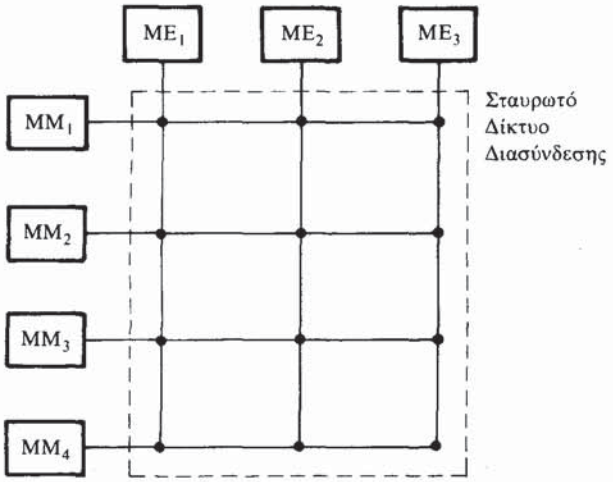


Σχ. 10.13. Σύστημα Π/Ε με μνήμες πολλαπλών εισόδων.

Το κυριότερο μειονέκτημα αυτών των συστημάτων είναι (α) το υψηλό κόστος των μνημών πολλαπλών εισόδων και (β) η μειωμένη επεκτασιμότητά τους (π.χ. είναι πολύ δύσκολο να συνδέσει κανείς μία τέταρτη ΜΕ σε σύστημα που διαθέτει ΜΜ τριών εισόδων).

Ένας τελευταίος τρόπος υλοποίησης στενά συνδεδεμένων συστημάτων Π/Ε παρουσιάζεται με τη χρήση ενός **σταυρωτού δικτύου διασύνδεσης** (crossbar switch), όπως φαίνεται στο Σχ. 10.14. Οι κόμβοι αυτού του δικτύου είναι προγραμματιζόμενοι διακόπτες και επιτρέπουν την επικοινωνία οποιασδήποτε ΜΕ με οποιαδήποτε ΜΜ. Με αυτόν τον τρόπο οι ΜΕ μπορούν να λειτουργούν παράλληλα, χρησιμοποιώντας συγκεκριμένες ΜΜ, ανάλογα με την κατάσταση των διακοπτών του δικτύου διασύνδεσης.

Ο μεγαλύτερος περιορισμός στην ανάπτυξη τέτοιων συστημάτων είναι το μεγάλο κόστος του δικτύου διασύνδεσης, η πολυπλοκότητα του οποίου αυξάνει εκθετικά καθώς αυξάνεται ο αριθμός των ΜΕ και των ΜΜ. Ως χαρακτηριστικές μηχανές Π/Ε αναφέρονται οι εξής: S-1 (Lawrence Livermore National Laboratory), Cm* και C.mmp (Carnegie Mellon University), IBM 3081, Cray X-MP και Cray-2.



Σχ. 10.14. Σύστημα Π/Ε με σταυρωτό δίκτυο διασύνδεσης.

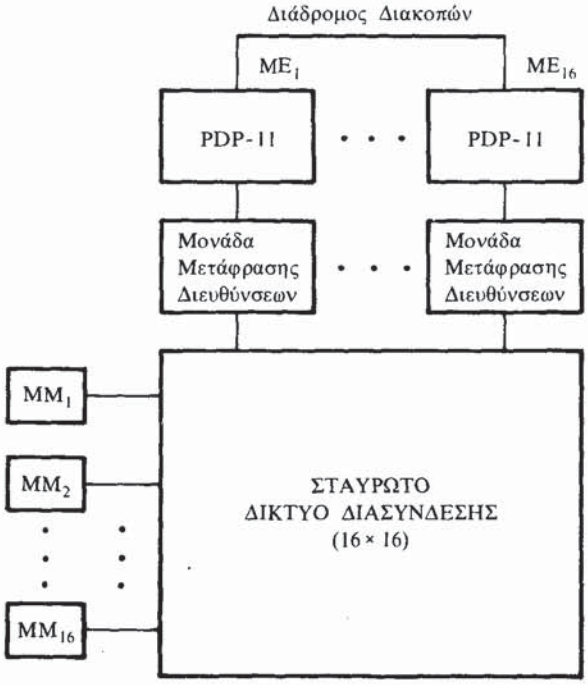
Τα τελευταία χρόνια εμφανίσθηκαν στην αγορά ορισμένοι νέοι μικροεπεξεργαστές, ειδικά σχεδιασμένοι για την ανάπτυξη συστημάτων Π/Ε. Έτσι δίνεται η δυνατότητα για τη σχεδίαση συστημάτων Π/Ε, με την καταλληλότερη, για κάθε περίπτωση, τοπολογία. Για την απλοποίηση των προβλημάτων συγχρονισμού των ΜΕ, συχνά αποφεύγεται η χρήση κοινών μονάδων μνήμης. Στις περιπτώσεις αυτές, η επικοινωνία μεταξύ των ΜΕ γίνεται μόνον μέσω προκαθορισμένων γραμμών επικοινωνίας (όπως στα χαλαρά συνδεδεμένα συστήματα). Χαρακτηριστικό παράδειγμα τέτοιων μικροεπεξεργαστών είναι ο Transputer της INMOS. Κάθε Transputer έχει μία εξαιρετικά απλή αρχιτεκτονική και εκτελεί περιορισμένο αριθμό εντολών. Τα προγράμματα και τα δεδομένα είναι αποθηκευμένα στην τοπική μνήμη κάθε Transputer. Η επικοινωνία με τους άλλους Transputers γίνεται μέσω σειριακών συνδέσμων επικοινωνίας (communication links). Κάθε Transputer διαθέτει (στην ίδια ψηφίδα) τέσσερις συνδέσμους εισόδου και τέσσερις συνδέσμους εξόδου. Κατά τη μεταφορά δεδομένων μέσω των συνδέσμων Ε/Ε, ακολουθείται ειδικό πρωτόκολλο επικοινωνίας για την εξασφάλιση της ορθής μετάδοσης των δεδομένων. Οι λειτουργίες Ε/Ε εκτελούνται ανεξάρτητα από τη λειτουργία της ΚΜΕ του Transputer. Έτσι, αποφεύγεται η επιβάρυνση της ΚΜΕ για τις λειτουργίες Ε/Ε. Τα συστήματα Π/Ε με βάση τον Transputer είναι ιδανικά για την παράλληλη

εκτέλεση αλγορίθμων, που περιγράφονται από ένα σύνολο ταυτόχρονων διεργασιών (concurrent processes). Για το λόγο αυτό έχει αναπτυχθεί ειδική γλώσσα υψηλού επιπέδου (η OCCAM) για την περιγραφή αλγορίθμων και για τον προγραμματισμό των Transputers ενός συστήματος Π/Ε.

Στη συνέχεια περιγράφεται μια χαρακτηριστική μηχανή Π/Ε με σταυρωτό δίκτυο διασύνδεσης, η C.mmp.

Παράδειγμα: Η μηχανή C.mmp

Η μηχανή C.mmp, που αναπτύχθηκε στο Πανεπιστήμιο Carnegie Mellon, χρησιμοποιεί ένα σταυρωτό δίκτυο διασύνδεσης διαστάσεων 16 × 16 (Σχ. 10.15). Οι 16 ΜΕ που χρησιμοποιούνται στο σύστημα είναι διάφορα



Σχ. 10.15. Η αρχιτεκτονική του συστήματος Π/Ε C.mmp.

μοντέλα του υπολογιστή PDP-11. Κάθε μία από τις 16 ΜΜ περιέχει 2M bytes. Επιπλέον, κάθε ΜΕ διαθέτει και μία τοπική μνήμη (8K bytes), όπου αποθηκεύονται κώδικες και δεδομένα για τις ρουτίνες εξυπηρέτησης διακοπών. Επειδή ο χώρος διευθύνσεων του PDP-11 είναι πολύ μικρότερος από το χώρο διευθύνσεων των 16 ΜΜ, χρησιμοποιείται για κάθε ΜΕ μία μονάδα μετάφρασης διευθύνσεων, για να υπάρχει δυνατότητα κάλυψης όλου του χώρου διευθύνσεων του συστήματος C.mmp. Εξάλλου, οι διάφορες ΜΕ του συστήματος έχουν τη δυνατότητα άμεσης επικοινωνίας μεταξύ τους, μέσω ενός διαδρόμου διακοπών. Έτσι, μία ΜΕ μπορεί να

διακόπτει κάποια άλλη για να την ειδοποιήσει σχετικά με κάποιο γεγονός ή για να της προωθήσει ορισμένα δεδομένα. Κάθε ΜΕ (PDP-11) του συστήματος C.mmp διαθέτει (εκτός από την τοπική μνήμη) και ορισμένες μονάδες Ε/Ε. Έτσι, δεν υπάρχει η δυνατότητα συμμερισμού (sharing) των διαφόρων μονάδων Ε/Ε του συστήματος από τις ΜΕ, με αποτέλεσμα το όλο σύστημα να μην είναι απολύτως συμμετρικό ως προς τις δυνατότητες των ΜΕ.

Η καθυστέρηση που εισάγεται από το δίκτυο διασύνδεσης και από τις μονάδες μετάφρασης διευθύνσεων είναι, για κάθε αναφορά στην κοινή μνήμη, περίπου 1 μsec. Αυτό είναι αρκετά υψηλό για τα σημερινά τεχνολογικά δεδομένα και οπωσδήποτε θέτει σοβαρά εμπόδια στην εμπορική χρήση παρόμοιων συστημάτων.

Ο συνολικός έλεγχος του συστήματος C.mmp γίνεται από το κατανεμημένο λειτουργικό σύστημα Hydra. Αυτό το λειτουργικό σύστημα φροντίζει (εκτός από την επικοινωνία με τους χρήστες) για τον προγραμματισμό του σταυρωτού δικτύου διασύνδεσης, για την ανάθεση των εργασιών (jobs) στις ΜΕ, για τη διαχείριση της κοινής μνήμης καθώς και για τη διαχείριση των μονάδων Ε/Ε. Με αυτό το λειτουργικό σύστημα καλύπτονται και οι διάφορες ασυμμετρίες που υπάρχουν ως προς τις δυνατότητες των ΜΕ του συστήματος.

10.3 ΜΗ ΣΥΜΒΑΤΙΚΕΣ ΠΑΡΑΛΛΗΛΕΣ ΜΗΧΑΝΕΣ

Σε όλους τους τύπους παράλληλων μηχανών που εξετάσθηκαν προηγουμένως, ο παραλληλισμός ελεγχόταν από έναν ή περισσότερους «μετρητές προγράμματος», οι οποίοι προσδιόριζαν την ακριβή ροή ελέγχου του παράλληλου προγράμματος. Αυτός ο μηχανισμός εκτέλεσης των (παράλληλων) προγραμμάτων είναι λογικά ισοδύναμος με το μηχανισμό εκτέλεσης στους συμβατικούς μη παράλληλους υπολογιστές (τύπου Von Neumann). Ένα άλλο χαρακτηριστικό των συμβατικών μοντέλων υπολογισμού είναι η ύπαρξη κεντρικής μνήμης για την αποθήκευση εντολών και δεδομένων. Η μεταφορά δεδομένων από ή προς τις μονάδες επεξεργασίας γίνεται μέσω του διαδρόμου του συστήματος. Όπως αναφέρθηκε και προηγουμένως, ο διάδρομος αυτός είναι ένα περιοριστικό στοιχείο στη χρησιμοποίηση πολλών μονάδων επεξεργασίας. Για το λόγο αυτό τα συστήματα πολυεπεξεργασίας σπανίως χρησιμοποιούν περισσότερες από 10 ανεξάρτητες μονάδες επεξεργασίας. Το γεγονός αυτό ενίσχυσε την άποψη ότι η αρχιτεκτονική των συμβατικών συστημάτων έχει φθάσει στα όριά της.

Ο βασικός στόχος της σύγχρονης έρευνας στο χώρο της αρχιτεκτονικής υπολογιστών είναι η πλήρης εκμετάλλευση του παραλληλισμού των αλγορίθμων. Κινητήρια δύναμη αυτής της έρευνας αποτέλεσε η εισαγωγή της τεχνολογίας VLSI, με τη βοήθεια της οποίας έγινε δυνατή η ανάπτυξη μεγάλων και αξιόπιστων συστημάτων υλικού σε μικρό χώρο και με μικρό κόστος. Γενικά, η τεχνολογία VLSI ευνοεί την ανάπτυξη αρχιτεκτονικών με τις παρακάτω ιδιότητες:

α) Οι τύποι των στοιχειωδών μονάδων (κυττάρων) της αρχιτεκτονικής να είναι λίγοι και σχετικά απλοί.
β) Οι διαδρομές των γραμμών ελέγχου των κυττάρων να είναι σχετικά απλές και κανονικές, δημιουργώντας έτσι ένα δίκτυο κυττάρων με, ως επί το πλείστον, τοπικές συνδέσεις και με περιορισμένο αριθμό συνδέσεων μεγάλου μήκους.

Στα πλαίσια αυτά, πολλοί είναι οι νέοι τύποι αρχιτεκτονικών, που αποβλέπουν στην πλήρη (μαζική) εκμετάλλευση του παραλληλισμού των αλγορίθμων. Αυτές οι μηχανές θεωρούνται ως εναλλακτικές προτάσεις για τους υπολογιστές της επόμενης (πέμπτης) γενιάς. Από τις μηχανές αυτές, οι οποίες, στο σύνολό τους, βρίσκονται ακόμη σε ερευνητικό στάδιο, θα εξετασθούν στη συνέχεια οι δύο κυριότερες κατηγορίες: οι μηχανές ροής δεδομένων και οι μηχανές αναγωγής. Η εξέταση των μηχανών αυτών θα συνδυασθεί με την εξέταση των μεθόδων προγραμματισμού τους, επειδή οι αρχιτεκτονικές αυτών των μηχανών έχουν μεγάλη συνάφεια με τους αντίστοιχους τρόπους προγραμματισμού.

10.3.1 Μηχανές Ροής Δεδομένων

Στις **μηχανές ροής δεδομένων** (ΜΡΔ, dataflow machines) οι εντολές μηχανής έχουν συναρτησιακή φύση, με την έννοια ότι το αποτέλεσμα (έξοδος) μιας εντολής είναι συνάρτηση μόνον των ορισμάτων (εισόδων) της. Επί πλέον, κάθε εντολή δεν δημιουργεί φαινόμενα παρενεργειών (side effects) στις μεταβλητές του προγράμματος επειδή, εκτός από τις μεταβλητές εξόδου της εντολής, δεν επηρεάζεται οποιαδήποτε άλλη μεταβλητή.

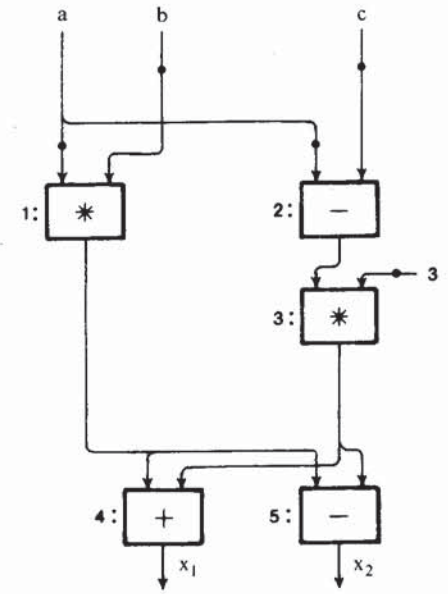
Ο μηχανισμός εκτέλεσης των εντολών στις ΜΡΔ είναι πολύ απλός. Μία εντολή εκτελείται μόνον όταν όλα τα δεδομένα εισόδου τής εντολής (ορίσματα) είναι διαθέσιμα. Έτσι ανατρέπεται η αρχή του κεντρικού ελέγχου στην εκτέλεση των εντολών, επειδή τα κριτήρια για την εκτέλεση μιας εντολής είναι «τοπικά» και δεν έχουν σχέση με τη γενικότερη κατάσταση του προγράμματος. Συνεπώς, η εκτέλεση των εντολών ενός προγράμματος ελέγχεται από την παρουσία ή όχι των δεδομένων σε αντιδιαστολή με τις συμβατικές μηχανές, όπου η εκτέλεση των εντολών ελέγχεται από τον μετρητή προγράμματος. Συνεπώς, στις ΜΡΔ μπορεί να επιτευχθεί ασύγχρονη εκτέλεση των εντολών μηχανής, οδηγώντας έτσι σε υψηλό βαθμό παραλληλισμού.

Γράφοι Ροής Δεδομένων

Τα προγράμματα μιας ΜΡΔ περιγράφονται συχνά με τη βοήθεια των γράφων ροής δεδομένων (dataflow graphs). Οι κόμβοι αυτών των γράφων αντιστοιχούν στις εντολές του προγράμματος, ενώ οι ακμές αντιστοιχούν σε δεδομένα. Ένας απλός γράφος ροής δεδομένων για τον υπολογισμό των τιμών:

$$x_{1,2} = \alpha * b \pm 3 * (\alpha - c)$$

φαίνεται στο Σχ. 10.16. Κάθε ακμή του γράφου αντιστοιχεί σε κάποιο δεδομένο. Η ύπαρξη μιας μεταβλητής δηλώνεται με μία ένδειξη (token) στην αντίστοιχη ακμή. Μία εντολή ενεργοποιείται μόνον όταν υπάρχουν ενδείξεις σε όλες τις ακμές των εισόδων της. Αποτέλεσμα της εκτέλεσης μιας εντολής είναι η διαγραφή των ενδείξεων από τις ακμές εισόδου και η δημιουργία νέων ενδείξεων για τις ακμές εξόδου της εντολής. Προφανώς, ένα πρόγραμμα ΜΡΔ ολοκληρώνεται όταν παραχθούν οι ενδείξεις για όλες τις εξόδους του προγράμματος. Στην αρχική κατάσταση, ο γράφος του Σχ. 10.16 έχει ενδείξεις για τις εισόδους a, b και c καθώς και για τη σταθερά

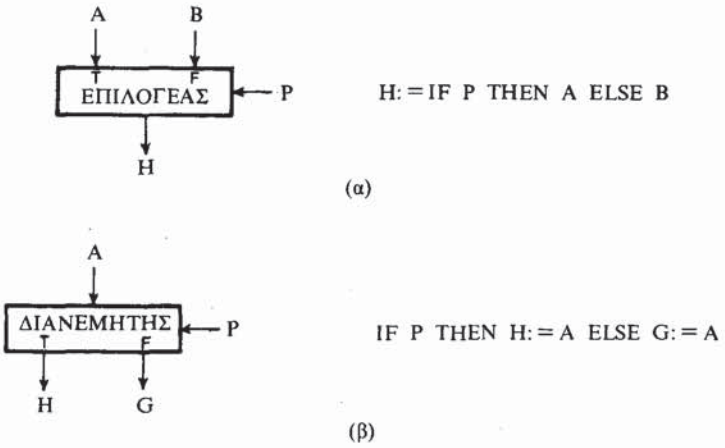


Σχ. 10.16. Ένας απλός γράφος ροής δεδομένων.

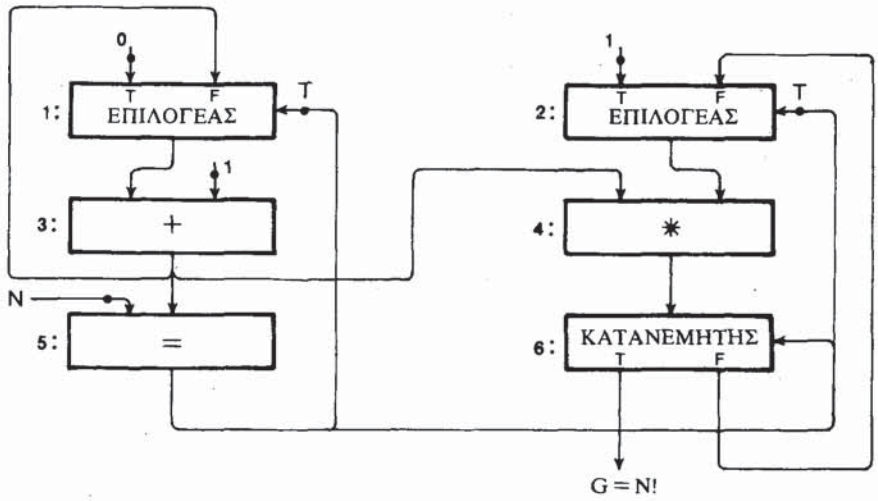
"3". Στην κατάσταση αυτή, μόνον οι εντολές 1 και 2 είναι εκτελέσιμες· αποτέλεσμα αυτών των εντολών είναι η δημιουργία των τιμών $a * b$ και $a - c$ από τους κόμβους (εντολές) 1 και 2, αντιστοίχως. Έτσι, μετά την εκτέλεση των εντολών 1 και 2, η εντολή 3 καθίσταται εκτελέσιμη, επειδή όλες οι είσοδοί της έχουν από μία ένδειξη. Η διαδικασία αυτή συνεχίζεται έως ότου υπολογισθούν οι τιμές των x_1 και x_2. Είναι φανερό ότι ο παραλληλισμός στην εκτέλεση των εντολών είναι μεταβλητός και εξαρτάται από τη μορφολογία του γράφου ροής δεδομένων.

Εκτός από αριθμητικές εντολές, είναι δυνατόν να χρησιμοποιηθούν και εντολές ελέγχου, όπως αυτές του Σχ. 10.17. Με τη βοήθεια αυτών των εντολών είναι δυνατή η περιγραφή επαναληπτικών αλγορίθμων και γενικότερα, αλγορίθμων μεγάλης πολυπλοκότητας. Στο Σχ. 10.18 φαίνεται ένας γράφος ροής δεδομένων για τον υπολογισμό του N!, για $N > 0$. Το

αποτέλεσμα G = N! υπολογίζεται με διαδοχικές επαναλήψεις. Το τέλος του υπολογισμού ελέγχεται από την εντολή 5, η οποία δίνει έξοδο T (αληθές) μόνον όταν οι δύο είσοδοί της είναι ίσες· διαφορετικά, η έξοδός της είναι F (ψευδές).



Σχ. 10.17. Εντολές για τον έλεγχο της ροής δεδομένων.



Σχ. 10.18. Γράφος ροής δεδομένων για τον υπολογισμό του N!.

Οι γράφοι ροής δεδομένων αποτελούν μία χαμηλού επιπέδου περιγραφή των αλγορίθμων, αντίστοιχη της συμβολικής γλώσσας των συμβατικών μηχανών. Για τη διευκόλυνση του έργου των προγραμματιστών έχουν αναπτυχθεί αρκετές νέες γλώσσες υψηλού επιπέδου (όπως η VAL, η ID, η

LAPSE και η VALID), με βασικό κριτήριο την αποτελεσματική μετάφρασή τους σε γράφο ροής δεδομένων και την ταχεία εκτέλεσή τους σε κάποια ΜΡΔ. Οι γλώσσες αυτές μοιάζουν εξωτερικά με συμβατικές γλώσσες υψηλού επιπέδου (π.χ. Pascal). Το κυριότερο χαρακτηριστικό τους όμως, είναι ότι οι μεταβλητές που χρησιμοποιούν δεν μπορούν να αλλάζουν τιμή. Για το λόγο αυτό οι γλώσσες αυτές αναφέρονται και ως **γλώσσες μοναδικής ανάθεσης** (single assignment languages).

Στις γλώσσες μοναδικής ανάθεσης επιτρέπεται η δημιουργία και η επεξεργασία δομών δεδομένων, με τρόπο ανάλογο της Pascal. Επίσης είναι δυνατή η δημιουργία διαδικασιών (procedures) για την ιεραρχική ανάπτυξη των προγραμμάτων.

Αρχιτεκτονική ΜΡΔ

Μία ΜΡΔ εκτελεί προγράμματα που είναι κωδικοποιημένα σε μορφή γράφων ροής δεδομένων. Κάθε κόμβος του γράφου αντιστοιχεί σε ένα πλαίσιο (template),στο οποίο αποθηκεύονται ο κώδικας της εντολής, τα ορίσματά της, και η διεύθυνση του πλαισίου για το οποίο προορίζονται τα αποτελέσματα του εν λόγω πλαισίου. Η μορφή των πλαισίων για το γράφο ροής δεδομένων του Σχ. 10.16 φαίνεται στον Πίνακα 10.3. Στο πεδίο «ορίσματα» φαίνονται οι τιμές των ορισμάτων τής κάθε εντολής, ενώ στο

ΠΙΝΑΚΑΣ 10.3. Η μορφή των πλαισίων για το γράφο του Σχ. 10.16.

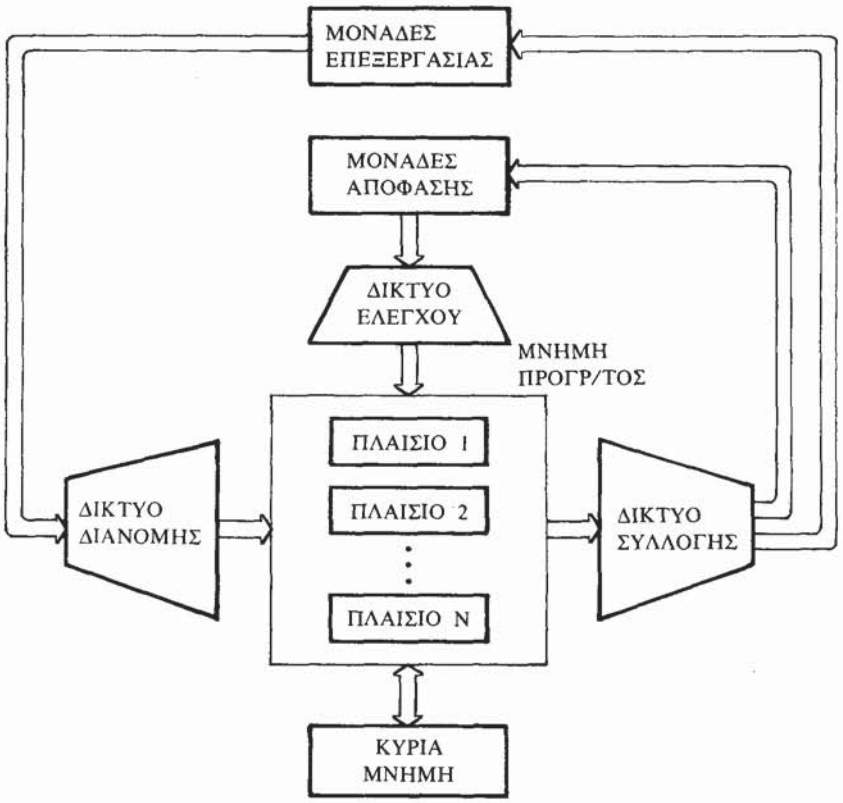
α/α	Λειτουργία	Ορίσματα* A	B	Έξοδοι* A	B
1:	*	α	b	4A	5A
2:	—	α	c	3A	—
3:	*	—	3	4B	5B
4:	+	—	—	x_1	—
5:	—	—	—	x_2	—

* A: η αριστερή είσοδος του κόμβου
B: η δεξιά είσοδος του κόμβου

πεδίο «έξοδοι» φαίνονται οι θέσεις των πλαισίων στα οποία προωθούνται τα αποτελέσματά της. Για παράδειγμα, το αποτέλεσμα της εντολής 3 (Πίνακας 10.3) θα οδηγηθεί στη δεύτερη είσοδο της εντολής 4 (4B) και στη δεύτερη είσοδο της εντολής 5 (5B).

Στο Σχ. 10.19 φαίνεται ένα απλοποιημένο διάγραμμα ενοτήτων για μία ΜΡΔ. Στη μνήμη προγράμματος φορτώνεται (από την κύρια μνήμη) το προς εκτέλεση πρόγραμμα, μαζί με τα δεδομένα εισόδου. Όταν η εντολή ενός πλαισίου είναι εκτελέσιμη (δηλαδή, όταν υπάρχουν δεδομένα σε όλες τις εισόδους του πεδίου ορισμάτων του πλαισίου), τότε αποστέλλεται στο

δίκτυο συλλογής ένα πακέτο με τον κώδικα της εντολής, τα ορίσματά της και τις διευθύνσεις των πλαισίων προορισμού τού αποτελέσματος. Το δίκτυο συλλογής προωθεί το πακέτο αυτό είτε σε κάποια μονάδα επεξεργα-



Σχ. 10.19. Απλοποιημένο διάγραμμα ΜΡΔ.

σίας (εάν πρόκειται για αριθμητική εντολή), είτε σε κάποια μονάδα απόφασης (εάν πρόκειται για λογική εντολή). Οι μονάδες επεξεργασίας (όπως και οι μονάδες απόφασης), αφού εκτελέσουν την εντολή, αποστέλλουν στο δίκτυο διανομής (ή στο δίκτυο ελέγχου) ένα πακέτο που περιέχει τα αποτελέσματα της εντολής, καθώς και τη διεύθυνση του πλαισίου για το οποίο προορίζεται το αποτέλεσμα. Η μονάδα διανομής (ή η μονάδα ελέγχου) φροντίζει για την αποθήκευση του αποτελέσματος στον κατάλληλο καταχωρητή του προβλεπόμενου πλαισίου.

Το αποτέλεσμα μιας εντολής είναι πιθανό να ενεργοποιήσει την εκτέλεση κάποιων άλλων εντολών, μέχρις ότου ολοκληρωθεί ο υπολογισμός των αποτελεσμάτων του προγράμματος· αυτά συνήθως οδηγούνται στην κύρια μνήμη για περαιτέρω επεξεργασία ή αποθήκευση.

Η περιγραφή που δόθηκε μέχρι τώρα αφορούσε μηχανές ροής δεδομέ-

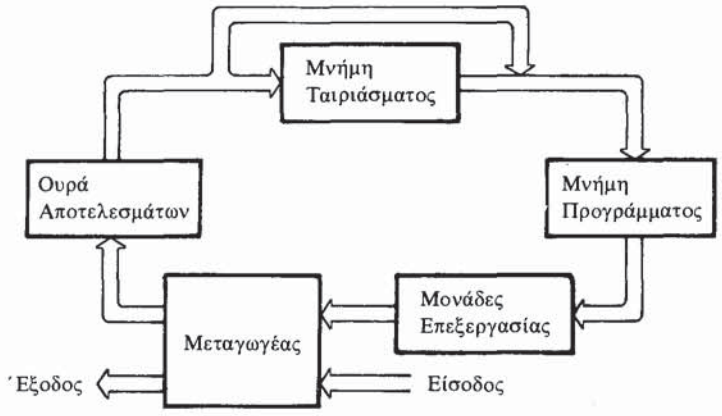
νων, που δεν επιτρέπουν να υπάρχουν περισσότερες από μία ενδείξεις (δεδομένα) σε κάθε ακμή του γράφου ροής δεδομένων. Μηχανές αυτού του τύπου αναφέρονται και ως **στατικές μηχανές ροής δεδομένων** (static dataflow machines).

Εάν επιτραπεί η ταυτόχρονη ύπαρξη περισσότερων από μία ενδείξεων για κάθε ακμή του γράφου, τότε κάθε δεδομένο πρέπει να περιέχει και επί πλέον πληροφορίες αναγνώρισης (ετικέττες, tags), έτσι ώστε να γίνεται ορθός συνδυασμός των δεδομένων που χρησιμοποιούνται από τις εντολές. Με τη χρήση των ετικεττών είναι δυνατόν μία εντολή να εκτελείται ταυτοχρόνως σε πολλές μονάδες επεξεργασίας· τα ορίσματα όμως της κάθε εκτέλεσης οφείλουν να έχουν κοινή ετικέττα. Μηχανές που χρησιμοποιούν ενδείξεις με ετικέττες (tagged tokens) αναφέρονται και ως **δυναμικές μηχανές ροής δεδομένων** (dynamic dataflow machines).

Πολλές ερευνητικές ομάδες και εταιρίες έχουν ενεργά προγράμματα για την ανάπτυξη μηχανών ροής δεδομένων. Τα σπουδαιότερα από αυτά είναι η μηχανή του MIT, η μηχανή του Manchester, η μηχανή DDM1 (Utah), η μηχανή LAU (Toulouse), η μηχανή EDDY καθώς και η ΜΡΔ της NTT. Στη συνέχεια εξετάζονται τα βασικά αρχιτεκτονικά χαρακτηριστικά της μηχανής του Manchester.

Παράδειγμα: Η Μηχανή του Manchester

Η μηχανή που αναπτύχθηκε στο Πανεπιστήμιο του Manchester είναι μια τυπική περίπτωση δυναμικής ΜΡΔ. Η γενική αρχιτεκτονική του συστήματος αυτού φαίνεται στο Σχ. 10.20. Οι λειτουργίες εισόδου/εξόδου



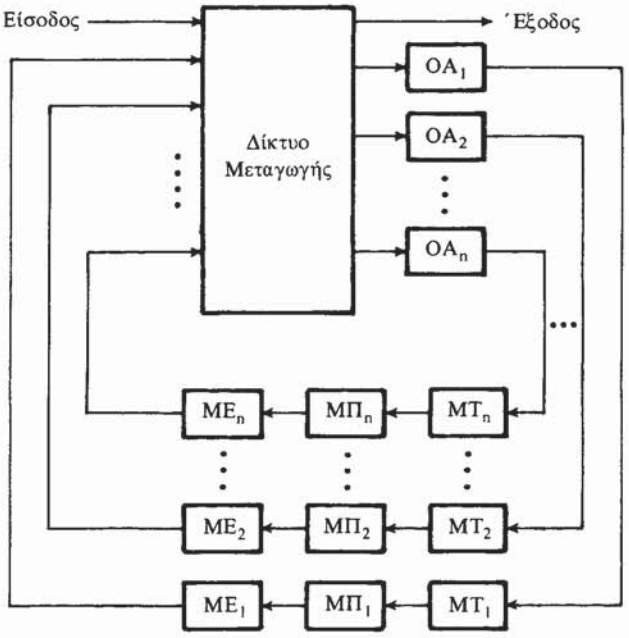
Σχ. 10.20. Γενική αρχιτεκτονική της ΜΡΔ του Manchester.

γίνονται μέσω του μεταγωγέα, ο οποίος αναγνωρίζει ποιά από τα δεδομένα των εισόδων του προορίζονται για τη ΜΡΔ και ποιά για τον εξωτερικό κόσμο.

Η μνήμη προγράμματος περιέχει τον κώδικα του προς εκτέλεση

προγράμματος. Κάθε εντολή περιλαμβάνει τον κώδικα εντολής και μία ή δύο διευθύνσεις προορισμού, για το αποτέλεσμα της εντολής. Το αποτέλεσμα της εκτέλεσης μιας εντολής από κάποια μονάδα επεξεργασίας αποτελείται από τρία πεδία: (α) το πεδίο τιμής, που κρατάει την τιμή του δεδομένου, (β) το πεδίο διεύθυνσης που περιέχει τη διεύθυνση της εντολής, για την οποία προορίζεται το δεδομένο και, (γ) το πεδίο ετικέττας. Το πεδίο ετικέττας προσδιορίζει τρία στοιχεία: (i) τη διεργασία (process), στην οποία ανήκει το παρόν δεδομένο, (ii) την ακμή του γράφου ροής δεδομένων, στην οποία το ίδιο το πεδίο ανήκει και, (iii) τον αριθμό ανακύκλωσης (iteration number), που διαφοροποιεί τα διάφορα δεδομένα, που ανήκουν στην ίδια ακμή του γράφου. Άρα η τιμή του αποτελέσματος οποιασδήποτε εντολής συνοδεύεται πάντα και από άλλες πληροφορίες, οι οποίες είναι απαραίτητες για την ομαλή εξέλιξη του προγράμματος.

Κάθε αποτέλεσμα οδηγείται αρχικά στην ουρά αποτελεσμάτων (FIFO), όπου αναμένει τη σειρά του για να εξετασθεί από τη μνήμη ταιριάσματος. Εάν το αποτέλεσμα αυτό προορίζεται για εντολή που δέχεται ένα μόνο όρισμα, τότε η μνήμη ταιριάσματος παρακάμπτεται και το αποτέλεσμα οδηγείται κατ' ευθείαν στη μνήμη προγράμματος. Εάν όμως το αποτέλεσμα προορίζεται για εντολή που δέχεται δύο ορίσματα, τότε, αρχικά εξετάζεται



ΟΑ: Ουρά Αποτελεσμάτων
ΜΤ: Μνήμη Ταιριάσματος
ΜΠ: Μνήμη Προγράμματος
ΜΕ: Μονάδες Επεξεργασίας

Σχ. 10.21. Δακτύλιος πολλαπλών μηχανών ροής δεδομένων.

το ενδεχόμενο να είναι ήδη αποθηκευμένο στη μνήμη ταιριάσματος το δεύτερο όρισμα της εντολής αυτής. Εάν δεν βρεθεί στη μνήμη ταιριάσματος το δεύτερο όρισμα, τότε το παρόν αποτέλεσμα απλώς καταχωρείται στη μνήμη αυτή. Αντίθετα, εάν βρεθεί στη μνήμη ταιριάσματος το άλλο όρισμα, τότε αυτό αποσύρεται από τη μνήμη και αποστέλλεται, μαζί με το παρόν αποτέλεσμα, στη μνήμη προγράμματος. Σημειώνεται ότι η λειτουργία της σύγκρισης των αποτελεσμάτων με τα περιεχόμενα της μνήμης ταιριάσματος γίνεται συσχετιστικά με τη χρησιμοποίηση της τεχνικής hashing.

Από τη μνήμη εντολών διαβάζονται, για κάθε εντολή, οι διευθύνσεις προορισμού των αποτελεσμάτων της, δημιουργείται ένα κατάλληλο πακέτο πληροφοριών και το τελευταίο οδηγείται σε κάποια μονάδα επεξεργασίας για εκτέλεση. Μετά την εκτέλεση δημιουργείται ένα νέο αποτέλεσμα που ακολουθεί την ίδια πορεία, όπως περιγράφηκε προηγουμένως.

Κάθε μονάδα επεξεργασίας είναι ένας μικροεπεξεργαστής τύπου ψηφιοφέτας (bit-slice) και εκτιμάται ότι απαιτεί, κατά μέσον όρο, χρόνο 3 μsec για την εκτέλεση μιας αριθμητικής εντολής. Έτσι, με την προϋπόθεση ότι οι 20 μονάδες επεξεργασίας κρατιούνται διαρκώς απασχολημένες, εκτιμάται ότι η μηχανή του Manchester, ως σύστημα, μπορεί να εκτελεί 6 εκατομμύρια εντολές το δευτερόλεπτο (6 MIPS).

Μια επέκταση της μηχανής του Manchester επιτρέπει το συνδυασμό πολλών μηχανών σε ένα δακτύλιο, όπως φαίνεται στο Σχ. 10.21. Με τον τρόπο αυτό, μπορούν να επιτευχθούν ταχύτητες πολλαπλάσιες των 6 MIPS της αρχικής μηχανής του Manchester. Ωστόσο, ο κώδικας του προγράμματος πρέπει να «διανεμηθεί» στις διάφορες μνήμες εντολών, με τρόπο ώστε να ισοκατανέμεται το φορτίο των υπολογισμών μεταξύ των επί μέρους κλάδων του δακτυλίου.

10.3.2 Μηχανές Αναγωγής

Οι **μηχανές αναγωγής** (reduction machines) αποτελούν μιαν άλλη κατηγορία μηχανών που αποβλέπουν στην παράλληλη εκτέλεση των προγραμμάτων, χωρίς την υποστήριξη κεντρικού ελέγχου. Αν και η έρευνα στο πεδίο των μηχανών αναγωγής δεν έχει προχωρήσει αρκετά, όπως στις ΜΡΔ, εν τούτοις έχουν παρουσιασθεί αρκετές προτάσεις με πολύ ενδιαφέροντα χαρακτηριστικά.

Τα προγράμματα μιας μηχανής αναγωγής (ΜΑ) δίνονται στη μορφή γραμμικών εκφράσεων. Οι εκφράσεις αυτές περιγράφουν με συναρτησιακό τρόπο τις λειτουργίες που πρέπει να εκτελεσθούν για να υπολογισθεί το αποτέλεσμα. Βασικό χαρακτηριστικό των ΜΑ είναι ότι υπολογίζουν το αποτέλεσμα ενός προγράμματος εκτελώντας διαδοχικές αναγωγές (reductions) της αρχικής έκφρασης που το περιγράφει. Οι εκφράσεις δίνονται συνήθως σε μορφή πρόθετου συμβολισμού (prefix notation)· έτσι η εντολή:

$$f := (a + b) * (c - 1)$$

μπορεί να γραφεί ως:

$f : (* \ e_1 \ e_2)$
$e_1 : (+ \ \alpha \ b)$
$e_2 : (- \ c \ 1)$

Εάν $\alpha = 2$, $b = 4$ και $c = 3$, τότε η αναγωγή οποιασδήποτε έκφρασης που περιέχει το f θα γίνει σε πέντε φάσεις, ως εξής:

$(\ldots f \ldots)$
φάση 1: $(\ldots (* \ e_1 \ e_2) \ldots)$
φάση 2: $(\ldots (* \ (+ \ \alpha \ b) \ (- \ c \ 1)) \ldots)$
φάση 3: $(\ldots (* \ (+ \ 2 \ 4) \ (- \ 3 \ 1)) \ldots)$
φάση 4: $(\ldots (* \ 6 \ 2) \ldots)$
φάση 5: $(\ldots 12 \ldots)$

Η τεχνική αυτή, με την οποία σε κάθε φάση αντιγράφονται οι ορισμοί των καλούμενων εκφράσεων, ονομάζεται **αναγωγή ακολουθίας** (string reduction). Εναλλακτικά μπορεί να χρησιμοποιηθεί η **αναγωγή γράφου** (graph reduction), όπου σε κάθε φάση της αναγωγής δημιουργούνται ειδικοί δείκτες στους ορισμούς των καλούμενων εκφράσεων. Με τη βοήθεια των δεικτών αυτών γίνεται δυνατός ο υπολογισμός της αρχικής έκφρασης, χωρίς να υπάρχει ανάγκη αντιγραφής των ορισμών των καλούμενων εκφράσεων.

Στη διαδικασία υπολογισμού των εκφράσεων, με τις τεχνικές που περιγράφηκαν προηγουμένως, μία αναγωγή ενεργοποιείται μόνον όταν η προς υπολογισμό έκφραση το απαιτήσει. Συνεπώς, ενώ στις ΜΡΔ η εκτέλεση μιας εντολής ενεργοποιείται από την ύπαρξη των δεδομένων εισόδου της, αντιθέτως στις ΜΑ μία εντολή ενεργοποιείται (ανάγεται) μόνον όταν ζητηθεί το αποτέλεσμά της (δηλ. η έξοδός της). Για το λόγο αυτό οι ΜΑ αναφέρονται συχνά και ως **μηχανές οδηγούμενες από τις απαιτήσεις** (demand-driven machines), ενώ οι ΜΡΔ ως **μηχανές οδηγούμενες από τα δεδομένα** (data-driven machines).

Οι ΜΑ έχουν σχεδιασθεί ειδικά για την παράλληλη εκτέλεση των αναγωγών ενός προγράμματος (ή ακριβέστερα, μιας έκφρασης). Σε όλες σχεδόν τις περιπτώσεις προγραμματίζονται σε κάποια συναρτησιακή γλώσσα, με τη βοήθεια της οποίας μπορούν να περιγραφούν με εύκολο τρόπο αρκετά πολύπλοκες εφαρμογές. Στις συναρτησιακές γλώσσες τα προγράμματα είναι ουσιαστικά συναρτήσεις, που ορίζονται με βάση απλούστερες συναρτήσεις. Στο κατώτερο επίπεδο υπάρχουν οι πρωταρχικές (primitive) συναρτήσεις, που εφαρμόζονται σε τιμές και δίνουν ως αποτέλεσμα άλλες τιμές.

Στη συνέχεια περιγράφεται η αρχιτεκτονική μιας αντιπροσωπευτικής ΜΑ, του Κυτταρικού Υπολογιστή (Cellulal Computer), που εκτελεί προγράμματα της συναρτησιακής γλώσσας FP.

Παράδειγμα: Η Γλώσσα FP και ο Κυτταρικός Υπολογιστής.

Ο κυτταρικός υπολογιστής (cellural computer), που αναπτύχθηκε στο Πανεπιστήμιο της Βόρειας Καρολίνας (ΗΠΑ), είναι μία παράλληλη ΜΑ που εκτελεί απ' ευθείας προγράμματα, γραμμένα στη συναρτησιακή γλώσσα FP (Functional Programming).

Στην FP, η εφαρμογή μιας συνάρτησης f σε ένα όρισμα (αντικείμενο) x συμβολίζεται ως f:χ. Γενικά, η γλώσσα FP αποτελείται από:

1) Ένα σύνολο από **αντικείμενα** (objects), που περιέχει όλα τα πιθανά ορίσματα. Τα αντικείμενα μπορεί να είναι είτε άτομα (π.χ. 3 ή F), είτε συνδυασμός ατόμων (π.χ. $<2, 6>$ ή $<3, 2, <5, 7>>$).
2) Ένα σύνολο από **πρωταρχικές συναρτήσεις** (primitive functions), με τη βοήθεια των οποίων αντιστοιχίζονται αντικείμενα σε αντικείμενα. Η συνάρτηση +, για παράδειγμα, όταν εφαρμοσθεί στο αντικείμενο $<3, 5>$ δίνει ως αποτέλεσμα το αντικείμενο 8, δηλαδή $+:<3, 5>\Rightarrow 8$. Εξάλλου, για τη συνάρτηση tr (transpose) ισχύει:

$$\begin{aligned}\mathrm{tr}:&<<x_1x_2,\ldots,x_n>, <y_1y_2,\ldots, y_n>>\Rightarrow\\ &\Rightarrow<<x_1,y_1>, <x_2,y_2>,\ldots,<x_n,y_n>>.\end{aligned}$$

3) Ένα σύνολο από **συναρτησιακές μορφές** (functional forms), οι οποίες συνδέουν υπάρχουσες συναρτήσεις για τη δημιουργία νέων συναρτήσεων. Παραδείγματα συναρτησιακών μορφών είναι:
   - Η σύνθεση $(f \circ g):x \Rightarrow f:g:x$
   - Η συνθήκη $(p \rightarrow f;g):x \Rightarrow$
   $$\Rightarrow \begin{cases} f:x, & \text{εάν } p:x = \mathrm{True} \\ g:x, & \text{εάν } p:x = \mathrm{False} \end{cases}$$
   - Η πλήρης εφαρμογή $(\alpha\alpha f):<x_1,x_2,\ldots,x_n>\Rightarrow$
   $$\Rightarrow<f:x_1,f:x_2,\ldots,f:x_n>.$$
4) Ένα σύνολο από **συναρτήσεις** (functions), που περιλαμβάνει όλες τις δυνατές συναρτήσεις (πρωταρχικές και οριζόμενες).

Για παράδειγμα, το εσωτερικό γινόμενο του διανύσματος $<1, 2, 3, 4>$ με το διάνυσμα $<11, 21, 31, 41>$ γράφεται και υπολογίζεται ως εξής:

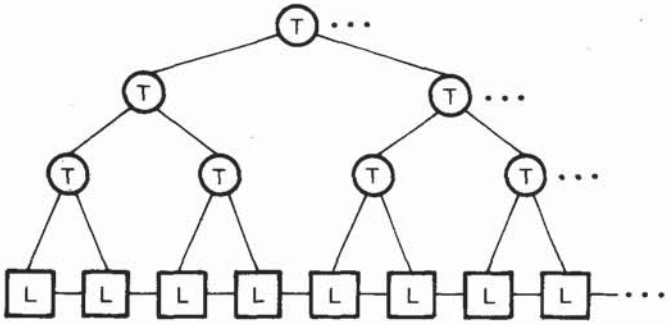
$$\begin{aligned}
P &= +\circ \alpha\alpha * \circ \mathrm{tr}:<<1, 2, 3, 4>, <11, 21, 31, 41>> && (10.1)\\
&= +:\alpha\alpha *:<<1, 11>, <2, 21>, <3, 31>, <4, 41>> && (10.2)\\
&= +:<*:<1, 11>, *:<2, 21>, *:<3, 31>, *:<4, 41>> && (10.3)\\
&= +:<11, 24, 39, 56>= && (10.4)\\
&= 130
\end{aligned}$$

Παρατηρούμε ότι δεν υπάρχουν μεταβλητές· για το λόγο αυτό, οι γλώσσες τέτοιου τύπου αναφέρονται και ως **γλώσσες μηδενικής ανάθεσης** (zero-assignment languages).

Η κυτταρική μηχανή έχει τη μορφή ενός δυαδικού δένδρου (Σχ. 10.22) και χρησιμοποιεί στους κόμβους της δύο ειδών κύτταρα (cells): τα κύτταρα L για τα φύλλα του δένδρου, και τα κύτταρα T για τους εσωτερικούς κόμβους. Τα κύτταρα L χρησιμοποιούνται ως στοιχεία μνήμης, ενώ τα κύτταρα T ως μονάδες επεξεργασίας και μέσα επικοινωνίας. Στα κύτταρα L

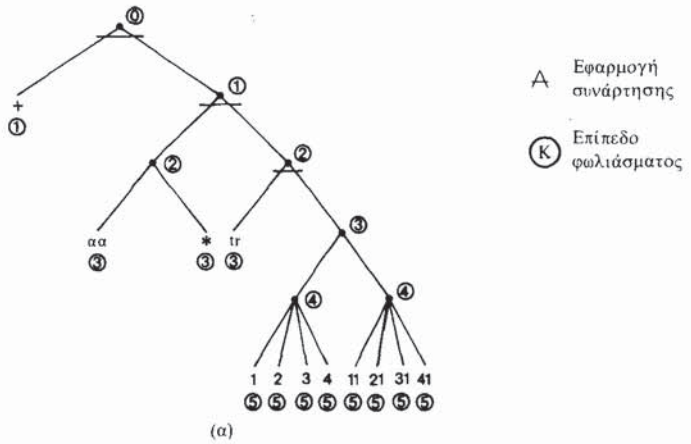
κρατείται η αρχική έκφραση FP σε μια κωδικοποιημένη μορφή. Η έκφραση (10.1) του προηγούμενου παραδείγματος μετατρέπεται στη μορφή:

$$<+, <(\alpha\alpha, *), <\mathrm{tr}, ((1, 2, 3, 4), (11, 21, 31, 41))>>>.$$



Σχ. 10.22. Η αρχιτεκτονική του κυτταρικού υπολογιστή.

Η έκφραση αυτή φαίνεται στο Σχ. 10.23 (α) σε μορφή δένδρου. Κάθε σύμβολο της έκφρασης FP αποθηκεύεται σε ένα κύτταρο L, μαζί με το αντίστοιχο επίπεδο φωλιάσματος (nesting level), όπως φαίνεται στο Σχ. 10.23 (β).



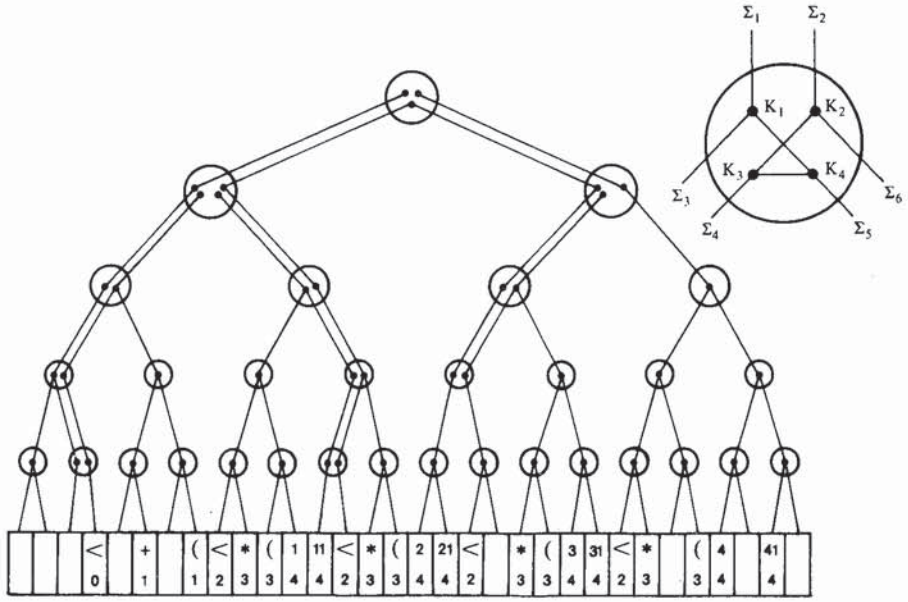
(α)

(β)

Σχ. 10.23. Η δενδρική μορφή μιας έκφρασης FP (α) και η φυσική της αποθήκευση στα κύτταρα L (β).

Η εκτέλεση μιας έκφρασης FP στην κυτταρική μηχανή γίνεται με διαδοχικές αναγωγές της έκφρασης FP, μέχρις ότου προκύψει το τελικό αποτέλεσμα. Σε κάθε αναγωγή, η έκφραση FP είτε αναλύεται σε πιο στοιχειώδεις συναρτήσεις, είτε απλοποιείται με την εκτέλεση εκείνων των συναρτήσεων FP που μπορούν να εκτελεσθούν κατά τη συγκεκριμένη φάση της αναγωγής.

Για την πραγματοποίηση της κάθε φάσης της αναγωγής ακολουθείται αρχικά μια διαδικασία **διαμερισμού** (partitioning) του δυαδικού δένδρου της κυτταρικής μηχανής. Κατά τη διαδικασία αυτή, η μηχανή χωρίζεται (λογικά) σε έναν αριθμό από υπόδενδρα, κάθε ένα από τα οποία αντιστοιχεί σε μια συνάρτηση FP. Στο Σχ. 10.24 φαίνεται η μορφή των υποδένδρων της κυτταρικής μηχανής για την έκφραση (10.3).



Σχ. 10.24. Διαμερισμός δένδρου κυτταρικής μηχανής.

Κάθε κύτταρο Τ της μηχανής διαθέτει τέσσερις προγραμματιζόμενους κόμβους ($K_1 - K_4$, Σχήμα 10.24), με τη βοήθεια των οποίων πραγματοποιείται ο διαμερισμός. Επιπλέον κάθε κύτταρο Τ διαθέτει έξι συνδέσμους επικοινωνίας ($\Sigma_1 - \Sigma_6$), δύο με κάθε ένα από τα γειτονικά του κύτταρα, για τη δημιουργία του διαμερισμένου δένδρου.

Τα τυχόν κενά κύτταρα L που προκύπτουν από προηγούμενες αναγωγές, δεν επηρεάζουν την αναγωγή της αποθηκευμένης έκφρασης FP, επειδή ο διαμερισμός γίνεται με βάση τα περιεχόμενα των μη κενών κυττάρων L.

Μετά το διαμερισμό, το κύτταρο Τ της ρίζας του κάθε υποδένδρου αναγνωρίζει την εκτελέσιμη συνάρτηση που είναι αποθηκευμένη στα

φύλλα του και, στη συνέχεια, λαμβάνει από κάποια κεντρική μνήμη το μικροπρόγραμμα που απαιτείται για την εκτέλεση αυτής της συνάρτησης. Για παράδειγμα, ο πολλαπλασιασμός 2 * 21, για την αναγωγή της έκφρασης του Σχ. 10.24, θα εκτελεσθεί από το κύτταρο Τ, που βρίσκεται στη ρίζα του υποδένδρου και τυχαίνει να είναι η ρίζα του όλου δένδρου. Οι πληροφορίες που είναι απαραίτητες για την πράξη αυτή μεταφέρονται μέσω των κυττάρων του αντίστοιχου υποδένδρου μέχρις ότου φθάσουν στη ρίζα του υποδένδρου. Τα αποτελέσματα της κάθε αναγωγής οδηγούνται στα κύτταρα L δια μέσου των κυττάρων Τ, έτσι ώστε να προετοιμασθεί η μηχανή για την επόμενη αναγωγή. Προφανώς, όλα τα υπόδενδρα που αντιστοιχούν σε εκτελέσιμες συναρτήσεις FP, μπορούν να εκτελούνται ταυτοχρόνως, εξασφαλίζοντας έτσι την αναγωγή των εκφράσεων FP με το μέγιστο δυνατό παραλληλισμό.

Η διαδικασία των διαδοχικών αναγωγών επαναλαμβάνεται έως ότου ολοκληρωθεί η εκτέλεση της αρχικής έκφρασης FP. Μετά από κάθε φάση της αναγωγής η δομή της κυτταρικής μηχανής επαναπροσδιορίζεται, με βάση την προκύπτουσα έκφραση FP.

Η απλότητα των κυττάρων και η τοπικότητα της επικοινωνίας τους καθιστούν τον κυτταρικό υπολογιστή ιδανικό για υλοποίηση με την τεχνολογία VLSI. Ένα άλλο σπουδαίο χαρακτηριστικό αυτής της μηχανής είναι η ασύγχρονη λειτουργία των κυττάρων Τ, γεγονός που επιτρέπει τη δημιουργία κυτταρικών υπολογιστών με πολύ μεγάλο αριθμό κυττάρων και, συνεπώς, με μεγάλο βαθμό παραλληλισμού. Ωστόσο, οι συνεχείς ανταλλαγές δεδομένων μεταξύ των διαφόρων κυττάρων δημιουργούν μια χρονική επιβάρυνση στην εκτέλεση των συναρτήσεων· αυτό είναι δυνατόν να έχει αρνητικές συνέπειες στο συνολικό χρόνο εκτέλεσης των προγραμμάτων (εκφράσεων FP).

10.4 ΠΑΡΑΛΛΗΛΕΣ ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ ΕΙΔΙΚΟΥ ΣΚΟΠΟΥ

Στις προηγούμενες παραγράφους αναφέρθηκε ότι μία από τις κυριότερες κινητήριες δυνάμεις για την ανάπτυξη παράλληλων μηχανών είναι οι νέες δυνατότητες της τεχνολογίας VLSI. Η τάση όμως για συστήματα παράλληλης επεξεργασίας δεν σταμάτησε μόνο στη σχεδίαση προγραμματιζόμενων μηχανών γενικού σκοπού, αλλά επεκτάθηκε και στη σχεδίαση αρχιτεκτονικών ειδικού σκοπού (special purpose architectures), για την εκτέλεση δηλαδή συγκεκριμένων εφαρμογών. Έτσι, παρουσιάσθηκαν μηχανές ειδικά σχεδιασμένες για επεξεργασία σήματος (signal processing machines), για επεξεργασία εικόνας (image processing machines), για βάσεις δεδομένων (database machines) κλπ. Οι μηχανές αυτές είναι συνήθως βοηθητικές (back-end) κάποιου άλλου κεντρικού υπολογιστή (host). Ο κεντρικός υπολογιστής φροντίζει να ενεργοποιεί τις μηχανές ειδικού σκοπού, να τους παρέχει δεδομένα εισόδου και να λαμβάνει τα αποτελέσματα. Η αρχιτεκτονική μιας μηχανής ειδικού σκοπού προσδιορίζεται κατά τρόπο ώστε να

διευκολύνεται στο μέγιστο δυνατό βαθμό η παράλληλη εκτέλεση των προκαθορισμένων εφαρμογών.

Τα τελευταία χρόνια εμφανίσθηκαν πολλές αρχιτεκτονικές που είναι σχεδιασμένες αποκλειστικά για την εκτέλεση ενός αλγορίθμου (όπως π.χ. διακριτός μετασχηματισμός Fourier ή πολλαπλασιασμός δισδιάστατων πινάκων). Τα κυριότερα χαρακτηριστικά αυτών των μηχανών είναι ο υψηλός βαθμός παραλληλισμού, η απλότητα των χρησιμοποιούμενων μονάδων επεξεργασίας, καθώς και η απλότητα του δικτύου επικοινωνίας. Ωστόσο, οι αρχιτεκτονικές αυτές υλοποιούν μόνο συγκεκριμένους αλγορίθμους και συχνά θέτουν όρια στις διαστάσεις των δεδομένων εισόδου. Ένα άλλο σημαντικό μειονέκτημα των συστημάτων αυτών είναι το υψηλό κόστος σχεδίασής τους, επειδή πρέπει, για κάθε αλγόριθμο, να αναζητείται η βέλτιστη αρχιτεκτονική τόσο σε σχέση με το βαθμό παραλληλισμού, όσο και σε σχέση με το φυσικό μέγεθος του αντίστοιχου VLSI κυκλώματος. Για την επίλυση αυτού του προβλήματος γίνονται πολλές προσπάθειες για την όσο το δυνατόν μεγαλύτερη αυτοματοποίηση της διαδικασίας σχεδίασης αρχιτεκτονικών ειδικού σκοπού.

Η πιο χαρακτηριστική κατηγορία ειδικευμένων αρχιτεκτονικών είναι τα **συστολικά συστήματα** (systolic systems). Ένα συστολικό σύστημα αποτελείται από ένα σύνολο διασυνδεδεμένων κυττάρων (μονάδων επεξεργασίας), κάθε ένα από τα οποία έχει την ικανότητα να εκτελεί κάποιες πολύ απλές λειτουργίες. Αναλόγως με τη μορφή του δικτύου επικοινωνίας μεταξύ των κυττάρων, διακρίνουμε τις **συστολικές μήτρες** (systolic arrays) και τα **συστολικά δένδρα** (systolic trees).

Οι λειτουργίες E/E στα συστολικά συστήματα γίνονται μόνο μέσω των κυττάρων που βρίσκονται στις πλευρές του συστήματος. Η μεταφορά πληροφοριών μεταξύ των κυττάρων γίνεται με τη βοήθεια των γραμμών του δικτύου επικοινωνίας. Η σύγχρονη λειτουργία των κυττάρων του συστήματος εξασφαλίζει ότι τα κατάλληλα δεδομένα θα βρίσκονται στην κατάλληλη θέση (κύτταρο) την κατάλληλη χρονική στιγμή. Εξάλλου, η ταυτόχρονη λειτουργία όλων των κυττάρων του συστήματος οδηγεί σε μεγάλους βαθμούς παραλληλισμού και συνεπώς σε μεγάλες ταχύτητες εκτέλεσης των αλγορίθμων. Στη συνέχεια περιγράφεται ένα συστολικό σύστημα για τον υπολογισμό της συνέλιξης (convolution).

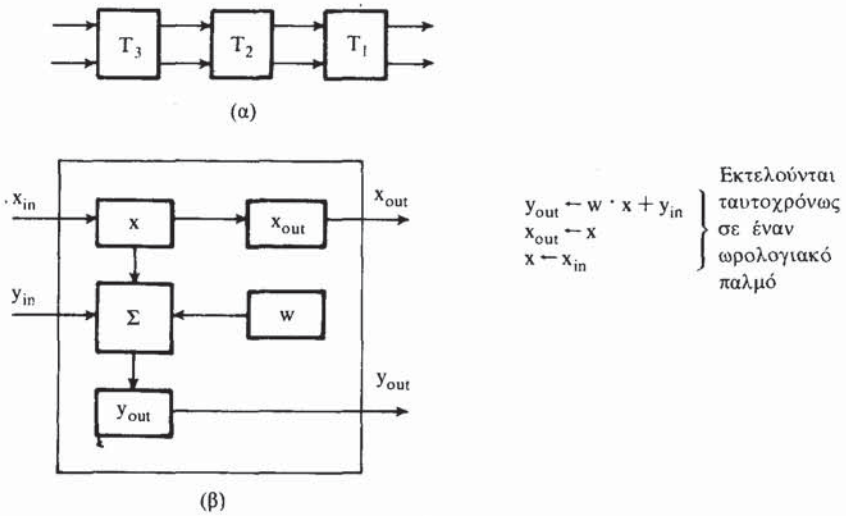
Η συνέλιξη ενός διανύσματος $X=\{x_1,x_2,...\}$ ορίζεται ως το διάνυσμα $Y=\{y_1,y_2,...\{$ έτσι ώστε:

$$y_i = \sum_{j=1}^{k} w_j \cdot x_{i+j-1} \qquad (10.5)$$

Όπου $W=\{w_1, w_2,...,w_\kappa\}$ οι συντελεστές της συνέλιξης.

Έστω ότι χρησιμοποιούνται τρεις συντελεστές συνέλιξης w_1, w_2 και w_3. Η συστολική αρχιτεκτονική για την περίπτωση αυτή μπορεί να έχει διάφορες μορφές. Στο Σχ. 10.25 (α) φαίνεται μία διάταξη συστολικής μήτρας, όπως προτάθηκε από τον Kung. Στη διάταξη αυτή χρησιμοποιούνται τρία κύτταρα· T_1, T_2 και T_3, η δομή των οποίων φαίνεται στο Σχ.

10.25 (β). Στους καταχωρητές w των κυττάρων, προαποθηκεύονται οι τρεις συντελεστές της συσχέτισης, όπως φαίνεται στο Σχ. 10.26 (α). Κάθε τιμή του x_{in} παραμένει στο εσωτερικό του κυττάρου (στον καταχωρητή x) για έναν κύκλο λειτουργίας (ωρολογιακό παλμό).

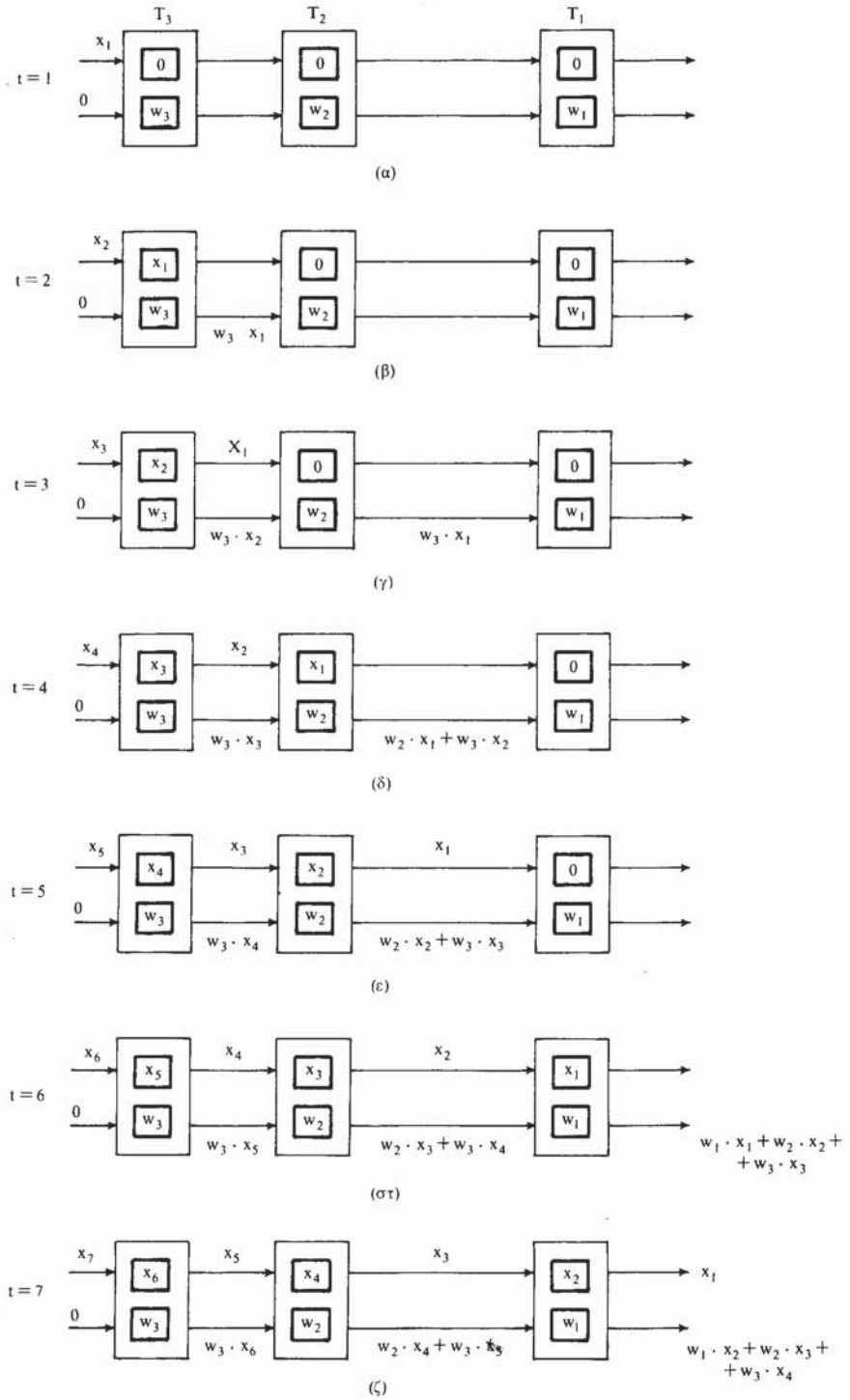


Σχ. 10.25. Μια συστολική μήτρα (α) και η δομή ενός κυττάρου (β).

Η ακολουθία $x_1, x_2, \ldots$ αποστέλλεται σειριακά στη γραμμή x_{in} του κυττάρου T_3. Σε κάθε κύκλο λειτουργίας στέλνεται μία μόνο τιμή του x. Εξάλλου, στην είσοδο y_{in} του κυττάρου T_3 συνδέεται μόνιμα το μηδέν. Θα αποδείξουμε στη συνέχεια ότι, με βάση αυτές τις προδιαγραφές, το σύστημα μπορεί να παράξει το διάνυσμα $Y = \{y_1, y_2, \ldots\}$ από τη γραμμή y_{out} του κυττάρου T_1.

Στο Σχ. 10.26 (α) φαίνεται η αρχική κατάσταση του συστήματος, μόλις φθάνει η τιμή x_1. Στο εσωτερικό του κάθε κυττάρου παριστάνονται τα περιεχόμενα του καταχωρητή x και του καταχωρητή w, ενώ στις γραμμές εξόδου παριστάνονται τα περιεχόμενα των καταχωρητών x_{out} και y_{out}.

Στο Σχ. 10.26 (β) φαίνεται η κατάσταση του συστήματος αμέσως μετά την ολοκλήρωση του πρώτου κύκλου λειτουργίας και λίγο πριν την έναρξη του δεύτερου κύκλου. Τα υπόλοιπα στιγμιότυπα του Σχ. 10.26 παρουσιάζουν την κατάσταση του συστήματος για τους τέσσερις επόμενους κύκλους λειτουργίας. Είναι εύκολο να διαπιστωθεί ότι κατά τον έκτο κύκλο λειτουργίας (Σχ. 10.26 (στ)) η έξοδος y_{out} του κυττάρου T_1 παράγει το στοιχείο y_1 του ζητούμενου διανύσματος συνέλιξης Y. Στους επόμενους κύκλους λειτουργίας της αρχιτεκτονικής, η έξοδος y_{out} του κυττάρου T_1 θα παράγει διαδοχικά και τα υπόλοιπα στοιχεία του διανύσματος της συνέλιξης. Συνεπώς, με την αρχιτεκτονική του Σχ. 10.25 μπορούμε να υπολογίζουμε ένα συντελεστή y ανά έναν κύκλο λειτουργίας. Η υψηλή αυτή ταχύτητα εκτέλεσης οφείλεται στο γεγονός ότι οι υπολογισμοί γίνονται ταυτοχρόνως από όλες τις μονάδες επεξεργασίας (κύτταρα).



Σχ. 10.26. Στιγμιότυπα της κατάστασης του συστήματος για τους επτά πρώτους.

ΑΣΚΗΣΕΙΣ

10.1. *Να εξηγηθεί γιατί η αύξηση της ταχύτητας των μονάδων επεξεργασίας σε έναν υπολογιστή, πρέπει να συμβαδίζει με αύξηση της ταχύτητας προσπέλασης στη μνήμη.*

10.2. *Μία μηχανή μητρώου πρέπει να εκτελεί παράλληλα την εξομάλυνση μιας εικόνας IM, που αποτελείται από $M \times N$ στοιχεία (pixels). Κατά την εξομάλυνση, η τιμή (φωτεινότητα) του κάθε στοιχείου IM[i, j] τροποποιείται και γίνεται ίση με τη μέση φωτεινότητα της περιοχής, που αποτελείται από το στοιχείο (i, j) και από τα 8 γειτονικά του στοιχεία.*

Να δοθεί μία διάταξη (τοπολογία) των μονάδων επεξεργασίας και να περιγραφεί η ακολουθία των εντολών που πρέπει να δημιουργηθεί από τον ελεγκτή της μηχανής μητρώου.

10.3. *Να εξηγηθεί γιατί η ταχύτητα μιας μηχανής αγωγού προσδιορίζεται από την καθυστέρηση που προκαλείται στη βραδύτερη βαθμίδα της.*

10.4. *Μία αιτία για καθυστέρηση στους αγωγούς εντολών είναι οι εντολές άλματος υπό συνθήκη, επειδή η διεύθυνση της επόμενης εντολής προσδιορίζεται μετά την εκτέλεση της παραπάνω εντολής. Μια λύση στο πρόβλημα αυτό θα μπορούσε να είναι η χρήση διπλού αγωγού, κάθε κλάδος του οποίου μπορεί να επεξεργάζεται μία διαφορετική ακολουθία εντολών.*

(α) Να εξηγηθεί ο μηχανισμός εκτέλεσης των εντολών άλματος υπό συνθήκη στην περίπτωση του διπλού αγωγού.

(β) Σε τι πλεονεκτεί και σε τι υστερεί ο διπλός αγωγός σε σχέση με τον απλό αγωγό εντολών;

10.5. *Να δοθεί μία διάταξη αριθμητικού αγωγού για τον παράλληλο υπολογισμό της εντολής*

```
FOR   I := 1 TO N DO
      Z[I]:= A[I] - 2 * B[I] + C[I] - 3;
```

χρησιμοποιώντας καταχωρητές, αθροιστές και κυκλώματα συμπληρώματος του 2. Επίσης, να δοθεί πίνακας με τα διαδοχικά περιεχόμενα των καταχωρητών για $1 \leqslant I \leqslant 5$.

10.6. *Να δοθεί η ακολουθία των εντολών του Cray-1 για την υλοποίηση της εντολής*

```
FOR   I := 1 TO N
      Z[I]:= R * A[I] + W * B[I] + X;
```

όταν $N \leqslant 64$. Επίσης, να βρεθούν οι εντολές που μπορούν να εκτελεσθούν παράλληλα, καθώς και οι εντολές που μπορούν να "αλυσιδοποιηθούν".

10.7. *Ποιά τα συγκριτικά πλεονεκτήματα (και μειονεκτήματα) των χαλαρά συνδεδεμένων συστημάτων πολυεπεξεργασίας σε σχέση με τα στενά συνδεδεμένα συστήματα πολυεπεξεργασίας.*

10.8. *Να σχεδιασθεί η μονάδα ελέγχου μιας μνήμης δύο εισόδων, εάν θεωρηθεί ότι κάθε είσοδος χρησιμοποιείται από ανεξάρτητη μονάδα επεξεργασίας (ΜΕ). Η μονάδα ελέγχου της μνήμης εξυπηρετεί τις αιτήσεις των ΜΕ με μία προκαθορισμένη προτεραιότητα. Η επικοινωνία της ΜΕ με τη μνήμη είναι ασύγχρονη. Έτσι, η ολοκλήρωση ενός κύκλου ανάγνωσης (ή εγγραφής) δηλώνεται από ένα σήμα, το οποίο πρέπει να παράγεται από τη μονάδα ελέγχου της μνήμης.*

10.9. *Να σχεδιασθεί ένα σταυρωτό δίκτυο διασύνδεσης, διαστάσεων 2 × 2, που να επιτρέπει τη μεταγωγή δύο μονάδων μνήμης και δύο μονάδων επεξεργασίας. Η κατάσταση των κόμβων του δικτύου διασύνδεσης ορίζεται από τα περιεχόμενα ενός καταχωρητή κατάστασης (CSS).*

10.10. *Έστω ότι στο παρακάτω πρόγραμμα οι είσοδοι είναι οι μεταβλητές a, b, c, ενώ η έξοδος (αποτέλεσμα) είναι η μεταβλητή z.*

$x := a * b + c;$
$y := a / b;$
$z := (x + y) * (x - y);$

(α) Να δοθεί ο γράφος ροής δεδομένων για τον υπολογισμό του z.
(β) Να δοθεί η μορφή των πλαισίων για τον παραπάνω γράφο.
(γ) Να περιγραφή βήμα-προς-βήμα, η διαδικασία εκτέλεσης του παραπάνω γράφου.

10.11. *Να δοθεί ο γράφος ροής δεδομένων για τον υπολογισμό των W, Z που περιγράφονται ως συναρτήσεις των K και L, σύμφωνα με το εξής πρόγραμμα:*

```
W := 1
Z := 0;
I := K;
Repeat
    W := W * I;
    Z := Z + I;
    I := I + 1;
Until I > L
```

10.12. *Να εξηγηθεί γιατί ο μηχανές που οδηγούνται από τις απαιτήσεις (demand-driven machines) μπορούν να αποφύγουν την εκτέλεση ορισμένων εντολών ενός προγράμματος, σε αντίθεση με ό,τι συμβαίνει στις μηχανές που οδηγούνται από τα δεδομένα (data driven machines). Να δοθεί ένα απλό παράδειγμα όπου να φαίνεται η παραπάνω διαφορά.*

10.13. *Να περιγραφεί η εκτέλεση της αναγωγής της έκφρασης FP (10.3) στην κυτταρική μηχανή, εάν αυτή διέθετε μόνο 15 κύτταρα L; Ποιά η χρονική επιβάρυνση εξαιτίας του περιορισμένου αριθμού κυττάρων L;*

10.14. *Να δοθεί μία συστολική μήτρα για τον υπολογισμό του γινομένου C δύο μητρώων A και B διαστάσεων 2 × N και N × 2, αντιστοίχως. (Υπόδειξη: Να χρησιμοποιηθεί για κάθε στοιχείο του μητρώου C ένα ανεξάρτητο κύτταρο).*

ΒΙΒΛΙΟΓΡΑΦΙΑ - ΑΝΑΦΟΡΕΣ

1. **Ackerman, W.B, Dennis, J.B.,** "VAL - A Value - oriented Algorithmic Language", TR-218, MIT, Lab. for Computer Science, June 1979.
2. **Alexandridis, N.,** Microprocessor System Design Concepts, Computer Science Press, 1984.
3. **Αλεξανδρίδης, Ν.Α.,** Ηλεκτρονικοί Υπολογιστές, Αθήνα 1981.
4. **Agrawala, A.K., Rauscher,** Foundations of Microprogramming: Architecture, Software and Applications, Academic Press, 1976.
5. **Arnold, J.S. et al,** "Design of Tightly - Coupled Multiprocessing Programming", IBM System Journal, 1, 1974.
6. **Arvind et. al.,** "A Dataflow Architecture with Tagged Tokens", Technical Memo 174, Lab. for Comp. Science, MIT, Sept. 1980.
7. **Backus, J.L.,** "Can Programming Be Liberated from the Von Neumann Style? A Functional Style and its Algebra of Programs", Communications of the ACM, 21, 8, Aug. 1978, pp. 613-641.
8. **Baer, J.L.,** Computer Systems Architecture, Comp. Science Press, 1980.
9. **Bouknight, W.J., et al,** "The ILLIAC IV System" Proceedings of the IEEE, 60, 4, Apr. 1972, pp. 369-388.
10. **Chen, S.C.,** "Large-Scale and High-Speed Multiprocessor System for Scientific Applications: Cray X-MP Series", Proc. NATO Advanced Research Workshop on High-Speed Computing.
11. **Comte, D., et al,** "LAU Multiprocessor: Microfunctional Description and Technologi cal Choices" Proc. European Conf. on Parallel and Distr. Processing, Feb. 1979, pp. 8-15.
12. **Denning, P.J.,** "Virtual Memory", ACM Computer Surveys, 2, 3, Sept. 1970, pp. 153-189.
13. **Dennis, J.B.,** "Dataflow Supercomputers", IEEE Computer, Nov. 1980, pp. 48-56.
14. **El-Ayat, K.A.,** "The Intel 8089: An Integrated I/O Processor", IEEE Computer, 12, 6, June 1979, pp. 67-78.
15. **Flynn, M.J.,** "Some Computer Organizations and Their Effectiveness", IEEE Trans. on Computers, C-21, 9, Sept. 1972, pp. 948-960.

16. **Fuller, S.H., Harbison, S.P.,** "The C.mmp Multiprocessor", Techn. Report, Carnegie-Mellon Univ., Comp. Science Dpt., 1978.

17. **Goodyear Aerospace Co.,** "Massively Parallel Processor (MPP)" TR-GER-16684, July 1979.

18. **Gorsline, G.W.,** Computer Organization, Prentice-Hall, 1986.

19. **Grinbert, J., et al,** "A cellular VLSI Architecture", IEEE Computer, Jan. 1984.

20. **Hayes, J.P.,** Computer Architecture and Organization, McGraw-Hill, 1978.

21. **Hwang, K.** (Editor), Tutorial on Supercomputer Design and Apllications, IEEE Comp. Society Press, Aug. 1984.

22. **Hwang, K., Briggs, F.A.,** "Computer Architecture and Parallel Processing", McGraw-Hill, 1984.

23. **IBM Corp.,** "Special Issue on IBM 3081", IBM Journal on Research and Development, 26, 1, Jan. 1982, pp. 2-29.

24. **Jones, A.K., Gehringer, E.F.** (Editors), "Cm* Multiprocessor Project: A Research Review", TR-CMU-CS-80-131, Carnegie-Mellon University, July 1980.

25. **Κάβουρας, Ι.Κ.,** Συστήματα Υπολογιστών, Τόμος Ι, Αθήνα 1986.

26. **Kartashev, S.I., Kartashev, S.P.,** "Problems in Designing Supercomputers with Dynamic Architectures", IEEE Trans on Comp., Dec. 1980, pp. 1114-1132.

27. **Koge, P.M.,** "The Architecture of Pipelined Computers", McGraw-Hill, 1981.

28. **Kuck, D.J.,** The Structure of Computers and Computations, Wiley, 1978.

29. **Kuhn, R.H., Padna, D.A.** (Editors) Tutorial on Parallel Processing, IEEE Computer Society Press, 1981.

30. **Kung, H.T.,** "Why Systolic Architectures", IEEE Computer, Jan. 1982, pp. 37-46.

31. **Lawrence Livermore Laboratory,** "The S-1 Project: Annual Reports on Architecture, Hardware and Software, UCID-18619, 1979.

32. **Lesea, A., Zaks, R.,** Microprocessor Interfacing Techniques, Sybex, 1978.

33. **Mago, G.A.,** "A Network of Microprocessors to Execute Reduction Languages", Int. Journal of Computer and Information Sciences, Vol. 8, No 5 and No 6, 1979.

34. **Mano, M.M.**, Computer System Architecture, Prentice-Hall, 1982.

35. **Mano, M.M.**, Digital Design, Prentice-Hall, 1984.

36. **Mc Namara, J.**, Technical Aspects of Data Communication, Digital Press, 1982.

37. **Mead, C., Conway, L.**, "Introduction to VLSI Systems", Addison-Wesley, 1980.

38. **Michael Andrews**, Computer Organization, Computer Science Press, 1987.

39. **Milutinovic, V., Fura, D.**, "An Introduction to GaAs Microprocessor Architecture for VLSI", IEEE Computer, March 1986, pp. 30-42.

40. **Moto-Oka, T., Fuchi, K.**, "The Architectures in the Fifth Generation Computers" Proc. 1983 IFIP Congress, 1983, pp. 589-602.

41. **Παπακωνσταντίνου, Γ., Μπιλάλης, Ν., Τσανάκας, Π.**, Λειτουργικά Συστήματα, Αθήνα 1986.

42. **Παπακωνσταντίνου, Γ., Φραγκάκης, Γ.**, Ψηφιακοί Υπολογιστές, Αθήνα 1979.

43. **Patterson, D.**, "Reduced Instruction Set Computers", Communications of the ACM, 2, 1, 1985, pp. 8-21.

44. **Perrott, R.H., Zarea-Aliabadi**, "Supercomputer Languages" ACM Computing Surveys, 18, 1, March 1986, pp. 5-23.

45. **Ramamoorthy, C.V., Li, H.F.**, "Pipeline Architecture", ACM Computing Surveys, 9, 1, March 1977, pp. 61-102.

46. **Russell, R.M.**, "The Cray-1 Computer System", Comm. of the ACM, Jan. 1978, pp. 63-72.

47. **Satyanarayanan, M.**, Multiprocessors: A Comparative Study, Prentice-Hall, 1980.

48. **Snini, V.P.**, "An Architectural Comparison of Dataflow Systems", IEEE Computer, March 1986, pp. 68-87.

49. **Stone, H.S.**, Introduction to Computer Architecture, Science Research Associates, 1975.

50. **Tannenbaum, A.S.**, Structured Computer Organization, Prentice-Hall, 1976.

51. **Tiberghien, J.** (Editor), New Computer Architectures, Academic Press, 1984.

52. **Treleaven, P.C., et al**, "Data-Driven and Demand-Driven Computer Architecture" ACM Computing Surveys, March 1982, pp. 93-144.

53. **Treleaven, P.C., et al,** "Combining Data Flow and Control Flow

54. **Φραγκάκης, Γ.,** Λογικά Κυκλώματα, Αθήναι 1975.

55. **Ullman, J.D.,** "Computational Aspects of VLSI, Comp. Science Press, 1984.

56. **Weitzman, Cay.,** Distributed Micro/Mini Computer Systems, Prentice-Hall, 1980.

57. **Wilkes, M.V.,** "The Best Way to Design an Automatic Calculating Machine", Rept. of the Manchester Univ. Comp. In. Conf., Manchester Univ., Electr. Eng. Dep. 1951, pp. 16-18.

58. **Wulf, W.A., et al,** HYDRA/C.mmp: An Experimental Computer System, McGraw-Hill, 1981.