Κεφάλαιο 3

Αρχιτεκτονικές Ενσωματωμένων Συστημάτων

Ο σχεδιασμός των ΕΣ μπορεί να επιτευχθεί σε διάφορα επίπεδα, ανάλογα με τις ανάγκες και τις προδιαγραφές που συνοδεύουν το εκάστοτε σύστημα. Από τη χρήση ενός μίνι έτοιμου υπολογιστή από το ράφι, που θα μπει μέσα σε ένα μηχάνημα ΑΤΜ, έως τη σχεδίαση εκ του μηδενός του επεξεργαστή και των περιφερειακών που θα χρησιμοποιηθούν. Ανάμεσα στις δυο αυτές ακραίες περιπτώσεις υπάρχουν πάρα πολλοί συνδυασμοί. Προκειμένου κάποιος μηχανικός να μπορεί να επιλέξει μια καλή υλοποίηση για το πρόβλημά του (ποτέ δεν υπάρχει απόλυτα βέλτιστη υλοποίηση, αφού υπάρχουν πάρα πολλοί συμβιβασμοί-trade-offs), θα πρέπει να γνωρίζει θέματα αρχιτεκτονικής και τεχνολογίες υλοποίησης ενσωματωμένων συστημάτων, καθώς και τις προκλήσεις, προβλήματα και οφέλη που συνοδεύουν κάθε επιλογή. Σε αυτό το κεφάλαιο καλύπτουμε την πρώτη απαίτηση και στο κεφάλαιο που ακολουθεί τη δεύτερη απαίτηση.

3.1 Θεμελιώδη στοιχεία της Αρχιτεκτονικής

Όπως έχει γίνει κατανοητό από την εισαγωγή, ένα ΕΣ βασίζεται στις ίδιες αρχές αρχιτεκτονικής που διέπουν κάθε προγραμματιζόμενο σύστημα. Τα δομικά του τμήματα είναι 3: (i) μνήμη, (ii) επεξεργαστής και (iii) είσοδος/έξοδος. Σε αυτό το κεφάλαιο συνοψίζουμε τις βασικές αρχές αρχιτεκτονικής, που βρίσκουν εφαρμογή στη συγκεκριμένη κατηγορία συστημάτων, περιγράφοντας βασικές γνώσεις για τα 3 δομικά τμήματα. Τα δεδομένα αποθηκεύονται ή ανακαλούνται από το (i), επεξεργάζονται και μετασχηματίζονται από το (ii) και μεταφέρονται μέσω διαύλων και δικτύων διασύνδεσης για το (iii). Σε αυτό το κεφάλαιο αναπτύσσουμε θέματα αρχιτεκτονικής που άπτονται της μνήμης και του επεξεργαστή, και σε επόμενο κεφάλαιο αναλύουμε την είσοδο/έξοδο.

3.2 Η μνήμη σε ένα προγραμματιζόμενο σύστημα

Όλα τα ΕΣ φέρουν μια ποικιλία τεχνολογιών και ιδιοτήτων μνήμης. Ακόμη και το πιο απλό ΕΣ που βασίζεται σε ένα μικροεπεξεργαστή 8 bit, όπως το δημοφιλές ATMEGA328P, έχει διάφορες μνήμες, όπως μνήμη SRAM για τους καταχωρητές, 2KB μνήμη RAM για την εκτέλεση του προγράμματος, μνήμη FLASH για την αποθήκευση του προγράμματος, και μνήμη EEPROM 1KB για την αποθήκευση ρυθμίσεων. Στο κεφάλαιο αυτό περιγράφονται όλες αυτές οι τεχνολογίες (SRAM, EEPROM, FLASH) και πολλές άλλες που συναντάμε στα ΕΣ.

Η μνήμη χρησιμοποιείται για να διατηρεί τα δεδομένα (είτε μόνιμα είτε όσο διατηρείται η τροφοδοσία), και το πρόγραμμα εκτέλεσης του επεξεργαστή. Υπάρχει ποικιλία τεχνολογιών μνήμης, και συχνά χρησιμοποιείται ένα μίγμα μέσα σε ένα ΕΣ. Κάποιο είδος μνήμης θα διατηρήσει το περιεχόμενό της, ενώ δεν τροφοδοτείται καθόλου με ενέργεια (αυτή η κατηγορία μνήμης ονομάζεται non volatile-μη πτητική, ενώ η αντίθετη περίπτωση ονομάζεται volatile-πτητική), όμως η πρόσβαση σε αυτή θα είναι αργή. Άλλες συσκευές μνήμης θα είναι μεγάλης χωρητικότητας, όμως θα απαιτούν στοιχεία κυκλώματος πρόσθετης υποστήριξης για την ανανέωση ή τον έλεγχο και θα είναι πιο αργές στην πρόσβαση. Ακόμα άλλες συσκευές μνήμης θα θυσιάσουν την χωρητικότητα για την ταχύτητα, που παράγουν οι σχετικά μικρές συσκευές, όμως θα είναι σε θέση να αντεπεξέλθουν με το γρηγορότερο επεξεργαστή. Γενικά, υπάρχουν πολλές τεχνολογίες και κυκλώματα μνήμης, γιατί όπως ισχύει σε κάθε σχεδιασμό συστήματος, υπάρχουν πάρα πολλοί βαθμοί ελευθερίας και αναλόγως της βαρύτητας που δίνεται, προκύπτουν διαφορετικά χαρακτηριστικά. Για αυτό και έχει ειπωθεί ότι ενώ οι χρήστες θέλουν την ελευθερία της επιλογής (όπως π.χ. στα καταναλωτικά προϊόντα) οι σχεδιαστές συστημάτων την απεχθάνονται γιατί τους κάνει πολύ δύσκολο το έργο της επιλογής των κατάλληλων υποσυστημάτων.

Μια οποιαδήποτε μνήμη αποτελείται από κελιά, όπου το κάθε κελί αποθηκεύει ένα bit. Το bit αποθηκεύεται ως τάση, όπου η ύπαρξη ή απουσία τάσης καθορίζει αν θα είναι 1 ή 0. Υπάρχουν και κάποιες πιο σύνθετες υλοποιήσεις που χρησιμοποιούν πολλαπλές τάσεις στο ίδιο κελί (ονομάζονται MLC, multi-level cell ή κελί πολλαπλών επιπέδων τάσης [15]) και επιτρέπουν την αύξηση της χωρητικότητας στο ίδιο κελί δηλαδή αύξηση της χωρητικότητας στην ίδια επιφάνεια πυριτίου (συνήθως αποθηκεύονται 2 bit σε ένα κελί του 1 bit), που όμως αν και έχουν περάσει δεκαετίες από τότε που προτάθηκαν βρίσκονται σε ερευνητικό στάδιο, γιατί απαιτούνται πολύ ευαίσθητοι ανιχνευτές τάσης και είναι πολύ επιρρεπείς σε τροποποίηση και αλλοίωση των αποτελεσμάτων, ιδιαίτερα για τις πολύ μικρές τεχνολογίες υλοποίησης (π.χ. πλάτος τρανζίστορ 14nm).

Τα bit ομαδοποιούνται συνήθως σε λέξεις των 8 στοιχείων (1 Byte) ή σε κάποιες περιπτώσεις και σε άλλα μεγέθη όπως των 32 Byte. Έτσι λοιπόν, αν σε μια μνήμη υπάρχουν m λέξεις των n bit η κάθε μια, συνολικά η μνήμη θα φέρει

$m*n$ bit. Αναφερόμαστε σε μια μνήμη ως μνήμη $m \times n$ ('m-από-n'). Κάποιες μνήμες έχουν επιπρόσθετα bit για διάγνωση και επιδιόρθωση σφαλμάτων, όπως οι μνήμες με υποστήριξη ECC (error correcting code, κώδικα για διόρθωση σφαλμάτων). Οι ECC μνήμες χρησιμοποιούνται σε περιβάλλοντα ακραίων καταστάσεων λειτουργίας, όπως στο ΕΣ που θα τοποθετηθεί σε ένα δορυφόρο και θα βομβαρδίζεται συνεχώς από την κοσμική ηλεκτρομαγνητική ακτινοβολία, δημιουργώντας μαλακά σφάλματα (soft errors), δηλαδή προσωρινή αλλοίωση των τιμών των bit [16].

Μια μνήμη αποτελείται από m λέξεις των n bit, οπότε θα πρέπει να μπορεί κάποιος να διευθυνσιοδοτήσει αυτές τις m λέξεις, ώστε να διαβάσει στην έξοδο δεδομένων της μνήμης, τη λέξη. Τα σήματα εισόδου διευθύνσεων $log_2(m)$ είναι απαραίτητα για να προσδιορίσουν μια συγκεκριμένη λέξη. Με άλλα λόγια, εάν μια μνήμη έχει k διευθύνσεις ως είσοδο, αυτή μπορεί να έχει μέχρι 2^k λέξεις. Ο δίαυλος των διευθύνσεων απαιτεί k bit. Αυτά μπορεί να βρίσκονται σε k διαφορετικές γραμμές ή σε λιγότερες με τη χρήση της πολύπλεξης. Η διευθυνσιοδότηση μιας μνήμης επιτρέπει την εγγραφή ή ανάγνωση μιας λέξης. Μερικές μνήμες μπορούν μόνο να προσπελαστούν (όπως η ROM), ενώ άλλες επιτρέπουν την προσπέλαση και την εγγραφή δεδομένων (όπως η RAM). Σε αντίθεση με τη μνήμη μόνο για ανάγνωση, δεν υπάρχει μνήμη μόνο για εγγραφή (δηλαδή, δε μπορεί να γίνει ανάγνωση), και ένα φύλλο δεδομένων (data sheet) για μια υποτιθέμενη τέτοια συσκευή, αποτελεί κλασσικό ανέκδοτο των αρχιτεκτόνων της δεκαετίας του 1970 [17].

Όλες οι μνήμες μπορούν να διευθυνσιοδοτούν λέξεις. Ένα στοιχείο που διαφοροποιεί τις μνήμες, είναι πόσα bit ή Byte έχει κάθε λέξη. Συνήθως κάθε λέξη είναι 8bit, ή 32bit, και αυτό σημαίνει ότι με μια διεύθυνση μνήμης ο επεξεργαστής αποκτά πρόσβαση σε 8 bit ή 32 bit αντίστοιχα. Άρα, σε περίπτωση που χρησιμοποιούνται 8 bit απαιτούνται 4 διαδοχικές διευθύνσεις για τη μεταφορά 32 bit. Υπάρχουν όμως και ειδικοί τρόποι πρόσβασης σε κάποιες μνήμες, που στέλνεται μόνο η αρχική διεύθυνση και το συνολικό μέγεθος των Byte που απαιτούνται για την ανάγνωση (burst transfer, μεταφορά ριπής).

Τα ολοκληρωμένα τσιπ μνήμης έρχονται σε διαφορετικά μεγέθη, ως προς την οργάνωση. Παραδείγματος χάριν, ένα τσιπ DRAM (δυναμική RAM) μπορεί να περιγραφεί ως 4MBx1 (οργανωμένο σε bit), ενώ ένα τσιπ SRAM (στατική RAM) μπορεί να είναι 512KBx8 (οργανωμένο σε λέξεις). Και στις δύο περιπτώσεις, κάθε τσιπ έχει ακριβώς την ίδια ικανότητα αποθήκευσης, αλλά είναι οργανωμένο με διαφορετικούς τρόπους. Στην περίπτωση της DRAM, θα απαιτούνταν οκτώ τσιπ συνδεδεμένα όλα στις ίδιες γραμμές διευθύνσεων, όπου το πρώτο τσιπ θα έδινε το bit ελάχιστης σημαντικότητας (LSB), ενώ το τελευταίο τσιπ θα μετέφερε το μεγαλύτερης σημαντικότητας (MSB). Ασφαλώς, η πρόσβαση θα ήταν παράλληλη σε αυτά τα τσιπ και ταυτόχρονα θα τοποθετούνταν στην έξοδο και τα 8 bit μόλις αποκωδικοποιούνταν η διεύθυνση μνήμης.

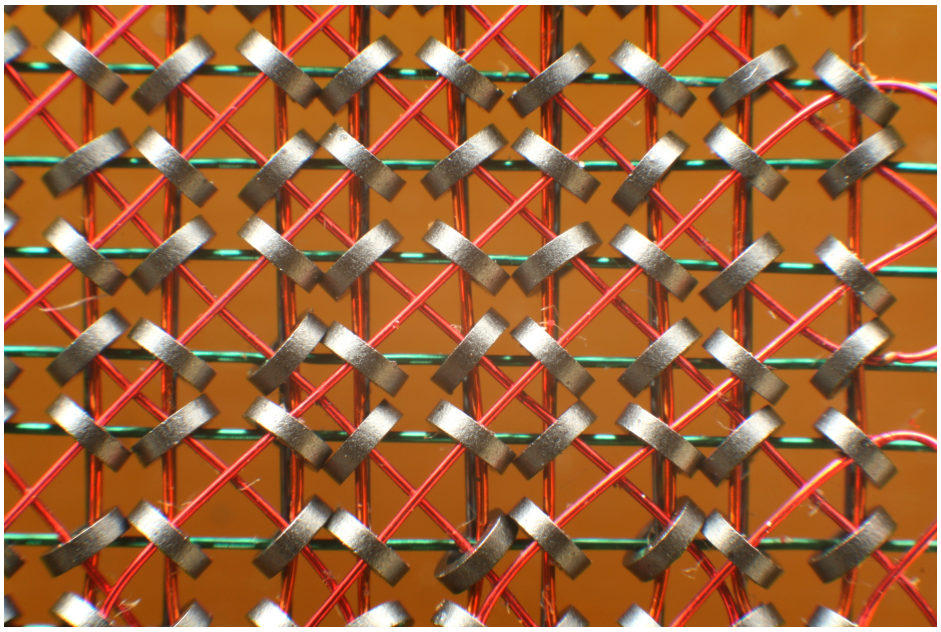
Στην περίπτωση της SRAM θα απαιτούνταν μόνο ένα τσιπ. Εντούτοις, επειδή συνήθως τα τσιπ DRAM οργανώνονται παράλληλα, υποστηρίζουν ταυτόχρονη προσπέλαση και επιτυγχάνονται καλές ταχύτητες πρόσβασης. Το τελικό μέγεθος για τη DRAM είναι (4MBx1)x8 ή 32MB.

3.2.1 Μνήμη τυχαίας προσπέλασης - Random Access Memory, RAM

Το ακρωνύμιο RAM σημαίνει μνήμη τυχαίας πρόσβασης ή προσπέλασης (Random Access Memory). Αυτό όμως είναι ένας ατυχής χαρακτηρισμός, δεδομένου ότι σχεδόν όλη η μνήμη των υπολογιστών μπορεί να θεωρηθεί 'τυχαίας πρόσβασης'. Βέβαια πρόσφατα, έχουν προταθεί στα ΕΣ και μνήμες 'σειριακές' (serial memory), όπως το Hybrid Memory Cube [18] από μια κοινοπραξία μεγάλων εταιριών (Xilinx, Altera, Microsoft, Samsung, ARM κ.α.), με το όρο 'serial' όμως να αφορά τον τρόπο μεταφοράς των δεδομένων προς και από αυτή τη συσκευή και όχι την εσωτερική δομή, όπου πάλι είναι τύπου RAM. Ο όρος RAM δημιουργήθηκε με τους πρώτους υπολογιστές με τρανζίστορ (δεκαετία 1960), αφού ως τότε υπήρχε ένα είδος μνήμης πυρήνα (magnetic core memory), όπως φαίνεται στην Εικόνα 3.1, και η μνήμη ήταν αρκετά αργή στην ανάγνωση και εγγραφή. Επίσης, εκείνη την εποχή, υπήρχαν μαγνητικές ταινίες αποθήκευσης ως μνήμες, που υποστήριζαν μόνο τη σειριακή εγγραφή και ανάγνωση. Η επινόηση της γρήγορης μνήμης RAM, έδωσε μια σημαντική ώθηση στους υπολογιστές.

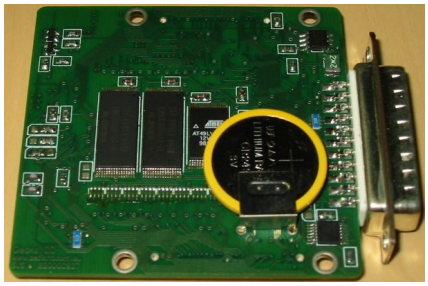
Η RAM είναι η μνήμη λειτουργίας στους ηλεκτρονικούς υπολογιστές. Είναι το μέρος όπου ο επεξεργαστής μπορεί εύκολα να γράψει δεδομένα για προσωρινή αποθήκευση (όσο τροφοδοτείται η μνήμη με ρεύμα). Η RAM είναι γενικά ευμετάβλητη (πτητική), χάνοντας το περιεχόμενό της όταν χάνει το σύστημα ενέργεια (ρεύμα). Οποιεσδήποτε πληροφορίες αποθηκεύονται στη RAM και πρέπει να διατηρηθούν, πρέπει να γραφτούν σε κάποια μορφή μόνιμης αποθήκευσης πριν το σύστημα χάσει την ισχύ του. Προκειμένου να αντιμετωπιστεί το μειονέκτημα της πτητικότητας των δεδομένων, που είναι κρίσιμης σημασίας σε διακομιστές, έχουν κατασκευαστεί μνήμες RAM με εφεδρικό σύστημα τροφοδότησης ενέργειας (Εικόνες 3.2 και 3.3), ώστε αν διακοπεί η τροφοδοσία να συνεχίσουν να τροφοδοτούνται με ρεύμα για να διατηρούν τα δεδομένα.

Οι μνήμες RAM διαιρούνται σε δύο οικογένειες: στατική RAM (επίσης γνωστή ως Static RAM ή SRAM) και δυναμική RAM (επίσης γνωστή ως dynamic RAM ή DRAM). Οι μνήμες SRAM χρησιμοποιούν πύλες λογικής σε συνδεσμολογία μανδαλωτών ή flip-flop, ώστε να διατηρούν κάθε bit των στοιχείων. Αυτή η κατηγορία είναι η γρηγορότερη διαθέσιμη μορφή RAM, απαιτεί ελάχιστα εξωτερικά στοιχεία κυκλώματος υποστήριξης, και έχει σχετικά μικρή κατανάλωση ενέργειας. Στα μειονεκτήματά τους συμπεριλαμβάνονται η μειωμένη



Σχήμα 3.1: Πριν από τη RAM χρησιμοποιούνταν μαγνητικοί δακτύλιοι μνήμης (1955-1975). Εικόνα από wikipedia.org.

χωρητικότητα (αφού συνήθως απαιτούν 6 τρανζίστορ, περίπου 6 φορές περισσότερη επιφάνεια από τις DRAM), το αυξημένο κόστος και η χαμηλότερη πυκνότητα ολοκλήρωσης. Για αυτό, χρησιμοποιούνται μόνο σε ειδικού τύπου κυκλώματα που απαιτούν μεγάλη ταχύτητα πρόσβασης (όπως οι κρυφές μνήμες-cache memorie). Η DRAM αντί για τρανζίστορ χρησιμοποιεί πυκνωτές για την αποθήκευση φορτίων, και αισθητήρια για την ανίχνευση ή όχι του φορτίου. Οι σειρές πυκνωτών θα κρατήσουν τη φόρτιση τους μόνο για μια μικρή χρονική περίοδο προτού η διαρροή εξαλείψει το φορτίο. Επομένως, η DRAM χρειάζεται μια συνεχή αναζωογόνηση-ανανέωση (refresh) όπου διαβάζεται το κελί και ξαναγράφεται, κάθε λίγα χιλιοστά του δευτερολέπτου (για παράδειγμα η μνήμη DRAM τεχνολογίας DDR απαιτεί κάθε κελί να ανανεώνεται ανά 7.8 μs). Αυτή η διαρκής ανάγκη ανανέωσης απαιτεί πρόσθετη υποστήριξη και μπορεί να καθυστερήσει την πρόσβαση σε αυτή, αν τη στιγμή που πρόκειται να γίνει μια ανάγνωση, απαιτείται ανανέωση του κελιού. Οι DRAM είναι συσκευές μνήμης υψηλής-χωρητικότητας διαθέσιμες και έρχονται σε μια ευρεία και διαφορετική ποικιλία υπό κατηγοριών. Λόγω της πολυπλοκότητας που έχουν μπορούν να διασυνδεθούν μόνο με κανονικούς επεξεργαστές, και δεν είναι πρακτική η σύνδεση με μικρούς μικροελεγκτές ΕΣ. Οι περισσότεροι επεξεργαστές με δυνατότητα διευθυνσιοδότησης μεγάλων μεγεθών μνήμης, έχουν εγγενή υποστήριξη για DRAM (ή αν δεν έχουν μπορούν να συνδεθούν σε ένα ενδιάμεσο ελεγκτή



Σχήμα 3.2: Ένα ΕΣ (Gecko100) που φέρει μνήμη RAM με υποστήριξη τροφοδότησης από μπαταρία. Εικόνα από austexsoftware.com.

Σχήμα 3.3: Μνήμη DRAM με ενσωματωμένη μπαταρία.

μνήμης).

Ένα σημαντικό πρόβλημα με τη χρήση της DRAM, προκύπτει από το γεγονός ότι είναι εξωτερική μνήμη (δε βρίσκεται στο ίδιο τσιπ με τον επεξεργαστή) και άρα υπάρχει αυξημένη καθυστέρηση και ενεργειακή ανάγκη για κάθε πρόσβαση. Για να θεραπευτεί κάπως αυτό το μειονέκτημα κάποιοι επεξεργαστές έχουν κρυφές (μη ορατές) μνήμες εντολών και δεδομένων πάνω στο ίδιο πυρίτιο με τον επεξεργαστή και αποθηκεύουν τις πρόσφατες προσβάσεις μνήμης. Αυτές οι μνήμες είναι (συχνά, αλλά όχι πάντα) εσωτερικές στους επεξεργαστές και υλοποιούνται με γρήγορα στοιχεία μνήμης και μεγάλης-ταχύτητας διαύλους δεδομένων. Έτσι λοιπόν δεδομένα που επαναχρησιμοποιούνται μπορούν να αποθηκευτούν προσωρινά σε αυτή τη μνήμη μόλις διαβαστούν και έτσι την επόμενη φορά που θα ζητηθούν να βρίσκονται κοντά στον επεξεργαστή. Οι κρυφές μνήμης είναι τεχνολογίας SRAM.

Οι μνήμες DRAM παρουσιάζουν μια διαρκή εξέλιξη τεχνολογίας από το 1975 που είχαν πρωτοπαρουσιασθεί, την οποία θα συνοψίσουμε σε λίγες γραμμές: Μια αύξηση των επιδόσεων επήλθε όταν η μνήμη συγχρονίστηκε με το ρολόι του διαύλου που συνέδεε τον επεξεργαστή με τη μνήμη. Βέβαια, η μνήμη θα έπρεπε να έχει αυστηρούς χρονισμούς και να υποστηρίζει παραμετροποιήσιμη καθυστέρηση (wait cycles), ώστε η όλη επικοινωνία να είναι συγχρονισμένη. Αυτή η μνήμη ονομάζονταν synchronous DRAM (SDRAM) και είχε ταχύτητες λειτουργίας (Mhz) SDRAM PC-66, SDRAM PC-100, SDRAM PC-133.

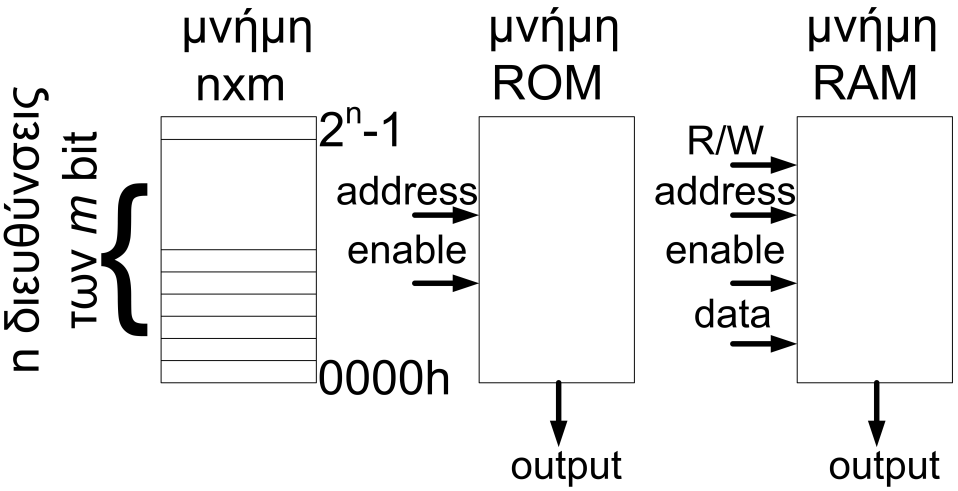
Μια βελτίωση στη SDRAM πραγματοποιήθηκε όταν άλλαξε η επικοινωνία με τη μνήμη, και από εκεί που η επικοινωνία γίνονταν με τις στάθμες της λογικής (1 και 0), στη συνέχεια γίνονταν με τις ανερχόμενες και κατερχόμενες ακμές του ρολογιού. Αυτό επέτρεψε να διπλασιαστεί ο ρυθμός επικοινωνίας με τη μνήμη (double data rate, DDR) για το ίδιο ρολόι του συστήματος. Οι ενδεικτικές ταχύτητες σε Mhz για αυτές τις μνήμες ήταν DDR-200, DDR-233, DDR-333, DDR-400.

Μια ακόμη βελτίωση συντελέστηκε όταν αυξήθηκε η συχνότητα λειτουργίας, μειώθηκε η τάση τροφοδοσίας και διπλασιάστηκε η χωρητικότητα του διαύλου δεδομένων (από 128 bit σε 256 bit). Αυτές οι μνήμες ονομάστηκαν DDR2 SDRAM με συχνότητες λειτουργίας DDR2-400, DDR2-533, DDR2-667, DDR2-800 και DDR2-1066. Η επόμενη γενιά των μνημών (DDR3 SDRAM) διπλασίασε το εύρος του διαύλου σε 512 bit, και την τάση λειτουργίας στα 1.5 Volt και συχνότητα λειτουργίας DDR3-1600. Η επόμενη γενιά που ακόμη δεν έχει καθιερωθεί από την αγορά γιατί είναι στα πρώιμα στάδια (παρουσιάστηκε το 2014), είναι η DDR4, με 1.2 Volt και με συχνότητες λειτουργίας DDR4-2133 και DDR4-3200. Οι μνήμες της τελευταίας γενιάς έχουν τόσο μεγάλες ταχύτητες που μπορούν να τις εκμεταλλευτούν μόνο εφαρμογές που κάνουν εντατική χρήση της μνήμης (εφαρμογές διακομιστών), για αυτό και η αύξηση της απόδοσης σε ένα σύστημα που αναβαθμίζεται από DDR3 σε DDR4 είναι 0 έως 5%, ενώ το κόστος αγοράς της μνήμης είναι 2 με 3 φορές περισσότερο [19]. Επίσης, υπάρχουν και τροποποιήσεις των DDR, όπως οι LPDDR() ($X = 2, 3, 4$) με το LP να σημαίνει low-power (χαμηλή κατανάλωση), δηλαδή μνήμες που λειτουργούν με μειωμένη τάση τροφοδοσίας και χρησιμοποιούνται σε φορητές συσκευές (π.χ. σε κινητά), GDDR(X) ($X = 2, 3, 4, 5$) με το G να σημαίνει graphics (γραφικά) και είναι μνήμες που χρησιμοποιούνται σε κάρτες γραφικών και φέρουν κάποιες εξειδικευμένες αρχιτεκτονικές τροποποιήσεις, Rambus RDRAM και η εξέλιξή της ως XDR SDRAM, οι οποίες είναι κλειστές τεχνολογίες με πατέντες που κατέχει η Rambus και υποστηρίζουν μεγαλύτερες συχνότητες λειτουργίας από τις αντίστοιχες DDR, αλλά με αυξημένο κόστος. Σε ερευνητικό επίπεδο (μετά τη DDR4) υπάρχει η τεχνολογία Υβριδικού Κύβου Μνήμης (Hybrid Memory Cube, HMC) που συναγωνίζεται την τεχνολογία Μνήμη Υψηλού Εύρους ζώνης (High Bandwidth Memory, HBM) ως προς το ποια θα επικρατήσει. Οι τεχνολογίες αυτές χρησιμοποιούν την τρισδιάστατη ολοκλήρωση, δηλαδή την κατακόρυφη τοποθέτηση πολλών τσιπ και την κατάλληλη σύνδεση με οπές (vias), ενώ έχουν μειώσει τις γραμμές μεταφοράς δεδομένων χρησιμοποιώντας υψηλής ταχύτητας σειριακή αποστολή και λήψη δεδομένων από τον επεξεργαστή. Όπως και σε άλλα δομικά στοιχεία, η τεχνολογία δε σταματάει να κάνει άλματα ανάπτυξης.

3.2.2 Μνήμη μόνο για Ανάγνωση –ROM

Η Read Only Memory (ROM), ή μνήμη μόνο για ανάγνωση, είναι μια μνήμη που μπορεί να προσπελαστεί για ανάγνωση, αλλά όχι να αποθηκεύσει δεδομένα κατά την τυπική λειτουργία. Φυσικά, υπάρχει τρόπος να γραφούν μια ή ελάχιστες φορές δεδομένα, αλλά αυτή η διαδικασία καλείται προγραμματισμός και όχι αποθήκευση. Τέτοιος προγραμματισμός γίνεται συνήθως εκτός σύνδεσης (off line), δηλαδή όταν δεν χρησιμοποιείται ενεργά η μνήμη ως μνήμη σε ένα ΕΣ. Μια ROM προγραμματίζεται συνήθως πριν τοποθετηθεί στην κατάλ-

ληλη υποδοχή του ΕΣ. Η Εικόνα 3.4 παρέχει ένα διάγραμμα δομικών στοιχείων μιας ROM.



Σχήμα 3.4: Βασικές μνήμες (a) λέξεις και Bits ανά λέξη, (b) διάγραμμα τμήματος ROM, (c) διάγραμμα τμήματος RAM.

Μια ROM μπορεί να χρησιμοποιηθεί για διάφορους σκοπούς σε ένα ΕΣ. Για παράδειγμα, μπορεί να αποτελεί το πρόγραμμα εκκίνησης (boot loader), δηλαδή το πρώτο πρόγραμμα που θα εκτελέσει ένα ΕΣ μόλις τροφοδοτηθεί με ρεύμα. Το πρόγραμμα αυτό μπορεί στη συνέχεια να μεταφορτώσει το υπόλοιπο πρόγραμμα ή το λειτουργικό σύστημα από κάποιο μέσο μεγαλύτερης χωρητικότητας, όπως μια μνήμη Flash. Για μικρά ΕΣ μια μνήμη ROM μπορεί να φέρει όλο το πρόγραμμα και να μην απαιτείται κάτι παραπάνω. Όμως, όπως θα δούμε παρακάτω, οι μνήμες RAM δε χρησιμοποιούνται πια, αφού έχουν κατασκευαστεί νέες οικογένειες μόνιμης αποθήκευσης με μεγαλύτερη ευελιξία. Μια ROM επίσης, μπορεί να χρησιμοποιηθεί για την υλοποίηση ενός συνδυαστικού κυκλώματος. Μπορεί να υλοποιηθεί οποιαδήποτε συνδυαστική συνάρτηση k μεταβλητών με τη χρησιμοποίηση μιας $2k$ x 1 ROM, και μπορούν να υλοποιηθούν n λειτουργίες των ίδιων μεταβλητών k χρησιμοποιώντας μια $2k \times n$ ROM.

Μια άλλη κοινή χρήση είναι η υλοποίηση ενός συνδυαστικού κυκλώματος. Μπορούμε να υλοποιήσουμε οποιαδήποτε συνδυαστική συνάρτηση. Προγραμματίζουμε απλά τη ROM για να υλοποιήσουμε τον πίνακα αλήθειας για την ή τις συναρτήσεις, που θέλουμε.

Η μνήμη ROM προγραμματίζεται μόνο στα εργοστάσια κατασκευής και δεν είναι δυνατός ο επαναπρογραμματισμός της. Η διαδικασία της φόρτωσης του λογισμικού σε μία ROM είναι γνωστή ως 'κάψιμο (burn) της ROM'. Αυτός ο όρος προέρχεται από το γεγονός ότι η διαδικασία προγραμματισμού εκτελείται περ-

νώντας ένα αρκετά μεγάλο ρεύμα μέσω των κατάλληλων διακοπτών που έχει το εσωτερικό πλέγμα της ROM για 'να τους καταστρέψει' ή να τους 'κάψει' και με αυτόν τον τρόπο δημιουργεί ένα μηδέν σε εκείνη την θέση bit. Μια συσκευή γνωστή ως 'καυστήρας' ή προγραμματιστής ROM μπορεί να επιτελέσει αυτή τη λειτουργία ή αν η ROM είναι ειδικού τύπου να προγραμματιστεί εντός συστήματος (In-System Programing-ISP) ή εντός κυκλώματος (In-Circuit Programming - ICP). Η απλή μνήμη ROM προγραμματίζεται μόνο μια φορά, γιατί κατά τον προγραμματισμό τροποποιείται μόνιμα το κύκλωμα. Αυτή η μνήμη χρησιμοποιείται σε ΕΣ που μόλις φύγουν από το εργοστάσιο δεν πρόκειται να μεταβληθούν (ή να ενημερωθούν).

Μια πιο ευέλικτη κατηγορία ROM, είναι η PROM (programmable ROM), η οποία μπορεί να προγραμματιστεί μια φορά από το χρήστη με ειδικό εξοπλισμό. Αυτές οι μνήμες ταιριάζουν καλύτερα για τη γρήγορη προτυποποίηση και στα ΕΣ χαμηλών απαιτήσεων.

Για να προγραμματιστεί μια συσκευή PROM, ο χρήστης παρέχει ένα αρχείο που καθορίζει το επιθυμητό περιεχόμενο για τη μνήμη. Ένα κομμάτι του εξοπλισμού αποκαλούμενο προγραμματιστής ROM (σημείωση: ο προγραμματιστής είναι εξοπλισμός υλικού, όχι ένα πρόσωπο που γράφει κώδικα λογισμικού) και έπειτα διαμορφώνει κάθε προγραμματίσημη σύνδεση σύμφωνα με το αρχείο. Μόλις προγραμματιστεί η ROM καταστρέφονται με μεγάλο ρεύμα οι κατάλληλες διασυνδέσεις, και οι εναπομείναντες διασυνδέσεις υλοποιούν το κύκλωμα. Οι συνδέσεις που καταστρέφονται δε μπορούν ποτέ να επανεγκαθιδρυθούν. Για αυτόν τον λόγο, η βασική PROM αναφέρεται συχνά ως μιας φοράς-χρονοπρογραμματίσημη συσκευή, ή OTP (One Time Programmable).

3.2.3 Erasable Programmable Memory

Ένας άλλος τύπος του PROM είναι μια διαγράψιμη PROM (erasable PROM, EPROM). Αυτή η συσκευή χρησιμοποιεί τρανζίστορ αιωρούμενης πύλης (floating gate), που κατά τον προγραμματισμό συνδέονται μέσω της έκχυσης ηλεκτρονίων (με υψηλότερη τάση από την κανονική για το τρανζίστορ 12V έως 25V), από το υπόλοιπο κύκλωμα. Με την κατάλληλη εφαρμογή της υψηλής τάσης επανπρογραμματίζεται στα νέα δεδομένα.

Τα OTP ROMs χρησιμοποιούνται όλο και λιγότερο στα ΕΣ, αφού δεν είναι ευέλικτα. Αν απαιτηθεί κάποια αλλαγή θα πρέπει να απομακρυνθεί και να πεταχθεί το παλαιό τσιπ και να προγραμματιστεί ένα νέο. Έτσι, η χρήση των OTP ROMS είναι μια ακριβή επιλογή ανάπτυξης ΕΣ, και προτιμούνται άλλα είδη μνήμης. Η καλύτερη επιλογή για την ανάπτυξη συστημάτων και τη διόρθωση είναι η διαγράψιμη προγραμματιζόμενη μνήμη μόνο για ανάγνωση (EPROM). Μια λάμψη υπεριώδους φωτός μέσω ενός μικρού παραθύρου στην κορυφή του τσιπ μπορεί να σβήσει το EPROM, επιτρέποντας να επαναπρογραμματιστεί και να

επαναχρησιμοποιηθεί. Αυτές οι μνήμες είναι συμβατές ως προς το σήμα και τον προγραμματισμό OTP ROMs. Κατά συνέπεια, ένα EPROM μπορεί να χρησιμοποιηθεί κατά τη διάρκεια της ανάπτυξης, ενώ ένα OTP ROM μπορεί να χρησιμοποιηθεί στην παραγωγή χωρίς καμία αλλαγή στο υπόλοιπο του συστήματος.

Τα EPROM και τα ισοδύναμα συγγενικά OTP τους κυμαίνονται, ως προς την χωρητικότητα, από μερικά KB (υπερβολικά σπάνιο αυτές τις μέρες), ως ένα megabyte ή περισσότερο.

3.2.4 Electrically Erasable Programmable Memory - EEPROM

Η EEPROM είναι ηλεκτρικά διαγράψιμη μνήμη μόνο για ανάγνωση, επίσης γνωστή ως EEPROM (ηλεκτρικά διαγράψιμη και προγραμματίσιμη μνήμη μιας ανάγνωσης). Πολύ σπάνια, καλείται επίσης ηλεκτρικά μεταβλητή μνήμη μιας ανάγνωσης (Electrically alterable read-only memory, EAROM).

Η EEPROM μπορεί να σβηστεί και να επαναπρογραμματιστεί μέσα σε ένα κύκλωμα, σε αντίθεση με τις προηγούμενες μνήμες που απαιτείται η απομάκρυνση από το κύκλωμα και η τοποθέτηση σε ειδική συσκευή. Η χωρητικότητα τους είναι σημαντικά μικρότερη από το τυποποιημένο ROM (μόνο μερικά kilobyte), και έτσι δεν χρησιμοποιούνται για αποθήκευση κώδικα προγραμματισμού. Αντ' αυτού, χρησιμοποιούνται, για να κρατούν τις παραμέτρους των συστημάτων και τις πληροφορίες που διατηρούνται κατά την αποσύνδεση.

Είναι κοινό για πολλούς μικροελεγκτές (όπως ATMEGA 328P), να ενσωματώνουν ένα μικρό τσιπ EEPROM για την αποθήκευση των παραμέτρων του συστήματος. Αυτό είναι ιδιαίτερα χρήσιμο στα ενσωματωμένα συστήματα και μπορεί να χρησιμοποιηθεί για την αποθήκευση των διευθύνσεων δικτύων, τοποθετήσεις διαμόρφωσης, αύξοντες αριθμούς, που συντηρούν τα αρχεία, κτλ. Για παράδειγμα, σε ΕΣ διαδικτύου που έχει σχεδιάσει ο συγγραφέας του βιβλίου, έχει τοποθετήσει κώδικα που ενεργοποιείται και διαβάζει τις εφεδρικές ρυθμίσεις λειτουργίας από την EEPROM σε περίπτωση που το ΕΣ δε μπορεί να συνδεθεί σε κάποιο διακομιστή ή να λάβει τις αυτόματες ρυθμίσεις από DHCP.

3.2.5 Η μνήμη FLASH

Η FLASH είναι η νεότερη τεχνολογία ROM και είναι αρκετά δημοφιλής. Συνήθως, συνυπάρχει μαζί με τη μνήμη EEPROM ή σε κάποια ΕΣ την έχει αντικαταστήσει τελείως. Η μνήμη FLASH έχει τα πλεονεκτήματα της επαναπρογραμματισημότητας εντός κυκλώματος (όπως η EEPROM) και τη μεγάλη χωρητικότητα όπως η ROM. Τα τσιπ FLASH αναφέρονται μερικές φορές ως 'flash ROMs' ή 'flash RAMs'. Δεδομένου ότι δεν είναι όπως τα τυποποιημένα ROMs ή όπως η τυποποιημένη RAM, προτιμούμε ακριβώς να τα λέμε 'FLASH' για αποφυγή σύγχυσης.

Η FLASH οργανώνεται κανονικά όπως οι τομείς σε ένα μαγνητικό δίσκο, και έχει το πλεονέκτημα ότι οι μεμονωμένοι τομείς μπορούν να διαγραφούν και να ξαναεγγραφούν χωρίς να επηρεάζουν το περιεχόμενο του υπολοίπου της συσκευής. Συγκεκριμένα, πριν μπορέσει να γραφτεί ένας τομέας, πρέπει να διαγραφεί. Μπορεί απλά να ξαναεγγραφτεί, όπως γίνεται και με ένα RAM.

Υπάρχουν αρκετές διαφορετικές τεχνολογίες FLASH, και οι απαιτήσεις διαγραφής και προγραμματισμού των συσκευών FLASH ποικίλλουν από κατασκευαστή σε κατασκευαστή. Οι δυο βασικές τεχνολογίες FLASH είναι η NOR FLASH και η NAND FLASH. Οι μνήμες FLASH είναι ιδιαίτερα δημοφιλείς και χρησιμοποιούνται σχεδόν σε όλα τα ΕΣ για την αποθήκευση δεδομένων και προγράμματος. Το μεγάλο πλεονέκτημα είναι ότι μπορούν να ενημερωθούν από τον κατασκευαστή εν ώρα λειτουργίας, χωρίς να χρειάζεται να μετακινηθεί η συσκευή (αν υπάρχει σύνδεση με το διαδίκτυο ή δυνατότητα σύνδεσης συσκευής USB που φέρει την ενημέρωση). Η NOR FLASH επιτρέπει τυχαία πρόσβαση σε οποιοδήποτε byte αλλά έχει μεγάλους χρόνους εγγραφής και είναι ακριβό κύκλωμα. Επειδή επιτρέπει διευθυνσιοδότηση ανά byte μπορεί να αντικαταστήσει άμεσα τα υπόλοιπα είδη ROM σε οποιοδήποτε κύκλωμα (drop-in replacement), και μπορεί να περιέχει και κώδικα που θα εκτελείται άμεσα από αυτήν (π.χ. ROM). Η NAND FLASH έχει μεγαλύτερη πυκνότητα αποθήκευσης, μικρότερο κόστος, λιγότερα καλώδια και πιο γρήγορους χρόνους εγγραφής, αλλά το μειονέκτημα είναι ότι οι προσβάσεις γίνονται πάντα σε ομάδες από bits (π.χ. 4096). Έτσι δεν υπάρχει άμεση τυχαία προσπέλαση σε οποιοδήποτε σημείο, αφού μεταφέρονται όλα τα bit σε κάθε πρόσβαση. Χρησιμοποιείται μόνο για αποθήκευση και όχι για άμεση εκτέλεση όπως η NOR FLASH. Αν απαιτηθεί να εκτελεστεί ο κώδικας που περιέχει, αντιγράφεται η αντίστοιχη ομάδα στη μνήμη RAM, και εκτελείται από εκεί. Εντούτοις, επειδή είναι αρκετά φθηνή χρησιμοποιείται σε πάρα πολλά ΕΣ. Όλες οι κάρτες SD για παράδειγμα που τοποθετούνται στα κινητά τηλέφωνα ή σε ΕΣ, είναι τεχνολογίας NAND FLASH.

3.2.6 Τρόπος Αποθήκευσης των Byte

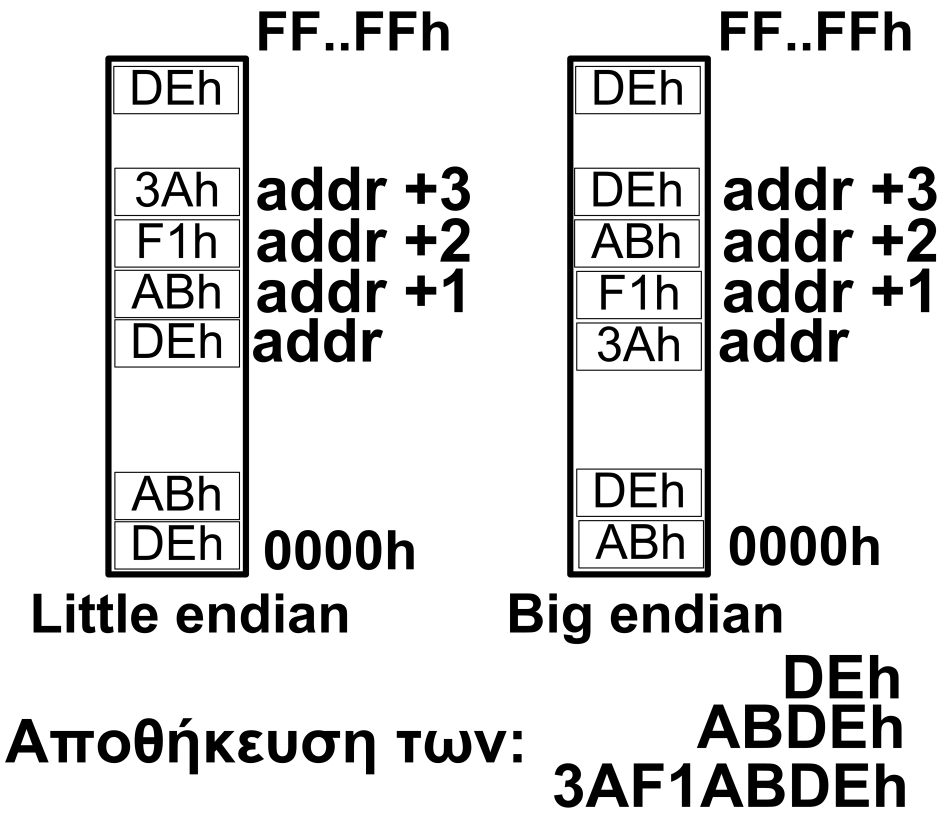
Αναπόσπαστο στοιχείο κάθε υπολογιστή είναι η μνήμη. Στην πληθώρα των περιπτώσεων ένας επεξεργαστής έχει μνήμη εντός (on-chip) και εκτός (off-chip) ολοκληρωμένου κυκλώματος. Στην on-chip μνήμη συγκαταλέγονται οι φανεροί και μη φανεροί καταχωρητές, η κρυφή μνήμη, και η μνήμη FLASH ή EEPROM (αν υπάρχει). Η off-chip μνήμη ομοίως μπορεί να είναι FLASH ή κάποιο είδος RAM. Τις περισσότερες περιπτώσεις η off-chip μνήμη είναι διευθυνσιοδοτούμενη ανά Byte, δηλαδή κάθε διεύθυνση μνήμης αντιστοιχεί σε μια θέση μνήμης χωρητικότητας 8 bit. Σε περίπτωση που ο επεξεργαστής χρησιμοποιεί μεγέθη αποθήκευσης δεδομένων μεγαλύτερα του 1 Byte (multi Byte), υπάρχουν δυο τρόποι αποθήκευσης στην εξωτερική μνήμη· αυτό ονομάζεται endianess (σειρά αποθή-

κευσης των Byte): big endian και little endian. Ο κάθε επεξεργαστής σχεδιάζεται για να υποστηρίζει τον έναν ή τον άλλο τρόπο αποθήκευσης και δεν αλλάζει. Υπάρχουν περιπτώσεις που ο επεξεργαστής έχει δυνατότητα να υποστηρίζει και τους 2 τρόπους αποθήκευσης με χρήση επιπρόσθετων κυκλωμάτων, οπότε σε αυτή την περίπτωση ονομάζεται bi-endian, και μπορεί με την κατάλληλη εντολή κώδικα μηχανής να αποφασίζεται κατά την εκτέλεση ενός προγράμματος ποιον τρόπο θα χρησιμοποιήσει, από τους ανωτέρω δυο. Η endianess αφορά την αρχιτεκτονική του επεξεργαστή και όχι της μνήμης, οπότε όλες οι μνήμες μπορούν να χρησιμοποιηθούν είτε με big είτε με little endian.

Η endianess αφορά μόνο την αποθήκευση μιας λέξης που αποτελείται από πολλά Byte. Ένα μόνο Byte θα αποθηκευτεί με τον ίδιο τρόπο είτε υπάρχει little είτε big endian. Σε περίπτωση όμως που η λέξη προς αποθήκευση αποτελείται από πολλά Byte, τότε η endianess καθορίζει τι θα αποθηκευτεί και που. Ένας big endian επεξεργαστής αποθηκεύει το σημαντικότερο Byte (Most Significant Byte, MSB) στη μικρότερη διεύθυνση και συνεχίζει να αποθηκεύει διαδοχικά τα επόμενα Byte, ως έως το χαμηλότερης σημαντικότητας Byte (Least Significant Byte, LSB), ενώ ένας little endian αρχιτεκτονικής επεξεργαστής αποθηκεύει το LSB στη μικρότερη διεύθυνση μνήμης και συνεχίζει διαδοχικά την αποθήκευση έως το MSB.

Η Εικόνα 3.5 παρουσιάζει ένα παράδειγμα αποθήκευσης μιας λέξης που αποτελείται από 1 Byte (η λέξη είναι DEh), μιας λέξης που αποτελείται από 2 Byte (η λέξη είναι ABDEh), και μιας λέξης που αποτελείται από 4 Byte (η λέξη είναι 3AF1ABDEh). Όταν η λέξη είναι 1 Byte μόνο, τότε θα αποθηκευτεί στη συγκεκριμένη διεύθυνση (π.χ. FFFFh) με τον ίδιο τρόπο και στις δυο endianess. Αν η λέξη είναι ABDEh, τότε το μεγαλύτερης σημαντικότητας Byte είναι το ABh, και το χαμηλότερης σημαντικότητας Byte είναι το DEh. Στη little endian θα αποθηκευτεί στη χαμηλότερη διεύθυνση πρώτα το DEh, και στην επόμενη διεύθυνση το ABh, ενώ στη big endian θα αποθηκευτεί στη χαμηλότερη διεύθυνση πρώτα το ABh και στην επόμενη διεύθυνση το DEh. Ομοίως, αν η λέξη είναι η 3AF1ABDEh, με MSB 3Ah και LSB DEh, και θέλουμε να το αποθηκεύσουμε στη διεύθυνση μνήμης $addr$, τότε στη little endian θα αποθηκευτεί πρώτα το LSB, δηλαδή το DEh, στην επόμενη διεύθυνση $addr + 1$ το επόμενο Byte, δηλαδή το ABh, στην επόμενη διεύθυνση $addr + 2$ το F1h και στην τελευταία addr+3 το 3Ah. Στη big endian η σειρά αποθήκευσης των Byte θα είναι αντίστροφη. Πρώτα το υψηλής σημαντικότητας Byte, δηλαδή το 3Ah, και στη συνέχεια τα υπόλοιπα με φθίνουσα πορεία, δηλαδή F1h, ABh και τέλος το DEh

Με τη little-endian αρχιτεκτονική, το λιγότερο σημαντικό στοιχείο μεταφέρεται πέρα από το λιγότερο σημαντικό μέρος των στοιχείων και αποθηκεύεται στη λιγότερη σημαντική θέση μνήμης. Για έναν προγραμματιστή, είναι ευκολότερο να γίνει κατανοητή η πορεία στοιχείων. Η δυσχέρεια της little endian αρχιτεκτονικής, είναι ότι το στοιχείο εμφανίζεται “προς τα πίσω” στη μνήμη



Σχήμα 3.5: Αρχιτεκτονική Μνήμης Big endian και Little Endian.

του υπολογιστή και έτσι υπάρχει ένα τμήμα θέσεων που πρέπει να διαβαστεί για να καθοριστεί το μέγεθος. Η αποθήκευση της λέξης 3AF1ABDEh στη μνήμη οδηγεί στην αποθήκευση DEABF13Ah στη μνήμη. Σημειώστε ότι ένας επεξεργαστής με little endian αρχιτεκτονική θα διαβάσει αυτό το στοιχείο ορθά, αφού υποστηρίζει αυτόν τον τρόπο αποθήκευσης, ασχέτως του πως αποθηκεύονται τα δεδομένα στη μνήμη. Εναλλακτικά, ένας επεξεργαστής με big endian αρχιτεκτονική που αποθηκεύει την τιμή 3AF1ABDEh στη μνήμη καταλήγει να αποθηκεύσει τελικά την ίδια τιμή 3AF1ABDEh μέσα στο τσιπ μνήμης. Αυτό φαίνεται (για έναν προγραμματιστή) να έχει περισσότερο νόημα. Κανένα σχέδιο δεν έχει περισσότερο πλεονέκτημα από το άλλο, η λειτουργία τους είναι ίδια, είναι δύο διαφορετικοί τρόποι με το ίδιο αποτέλεσμα. Όταν κάνετε υψηλού επιπέδου προγραμματισμό σε ένα σύστημα, το “endian-ness” έχει λίγη διαφορά. Ο μόνος χρόνος που εκτίθεστε πραγματικά σε κάτι τέτοιο, είναι όταν εξετάζετε τα στοιχεία πολλαπλών Byte άμεσα στη μνήμη και καλείστε να τα ερμηνεύσετε. Εντούτοις, όταν αναπτύσσετε και διορθώνετε το υλικό χαμηλού επιπέδου σταθερού λογισμικού, θα τα συναντάτε πολύ συχνά και έτσι μια κατανόηση big endian και

little endian αρχιτεκτονικών στο σχεδιαστή των ενσωματωμένων συστημάτων είναι απαραίτητη.

Η endianess εμφανίζεται οπουδήποτε πρέπει να αποθηκευτεί ή να μεταφερθεί μια λέξη που αποτελείται από πολλαπλά Byte. Εκτός από την αποθήκευση λέξεων στη μνήμη, εμφανίζεται και στις επικοινωνίες δικτύου, δηλαδή κατά την αποστολή και λήψη δεδομένων μέσω δικτυακών πρωτοκόλλων, όπως το TCP/IP. Αυτό οφείλεται στο γεγονός, ότι μπορεί ο αποστολέας και ο παραλήπτης να έχουν διαφορετικά endianess και έτσι να ερμηνεύσουν διαφορετικά τα Byte. Για παράδειγμα, αν ένας υπολογιστής λάβει τη λέξη 3AF1ABDEh και είναι big endian θα ερμηνεύσει τα δεδομένα ως τον 32bit αριθμό 3AF1ABDEh, ενώ αν είναι little endian θα ερμηνεύσει τα δεδομένα ως DEABF13Ah. Για αυτό το λόγο, έχουν δημιουργηθεί ειδικές συναρτήσεις που πρέπει να καλούνται πριν την αποστολή και μετά τη λήψη των δεδομένων, ώστε να μετατρέπουν τα δεδομένα σε σειρά αποθήκευσης δικτύου (network byte order, NBO), να στέλνονται, και στη λήψη να μετατρέπονται από τη NBO στην endianess του υπολογιστή. Ένα ζεύγος τέτοιων συναρτήσεων είναι οι htons()/ntohs() που υπάρχει στο πρότυπο IEEE Std 1003.1-2001 ('POSIX.1'). Το πρόβλημα με το endianess εμφανίζεται και στην αποθήκευση δεδομένων σε αρχεία που μοιράζονται υπολογιστές με διαφορετικό endianess, και η συνηθισμένη λύση είναι η χρήση των συναρτήσεων htons()/ntohs(), ακόμη και αν δεν υπάρχουν δικτυακές επικοινωνίες.

Παραδείγματα επεξεργαστών που υποστηρίζουν big endian είναι οι Sun SPARC, Motorola 68K, η οικογένεια των PowerPC ή παλαιότερα οι IBM S/360, ενώ παραδείγματα επεξεργαστών αρχιτεκτονικής little endian είναι οι επεξεργαστές που υποστηρίζουν την οικογένεια x86, IA32 ή AMD64 εταιριών Intel, AMD και άλλων ή παλαιότερα οι DEC Vax. Η endianess είναι πολύ σημαντική στα ΕΣ, αφού σε αντίθεση με τους επιτραπέζιους υπολογιστές ή διακομιστές που όλα είναι συμβατά με x86 (little endian), υπάρχουν πολλές διαφορετικές αρχιτεκτονικές ή σχεδιασμοί νέων αρχιτεκτονικών και έτσι απαιτείται αυτή η γνώση σε ένα μηχανικό που ασχολείται με αυτά. Ως προς τα οφέλη ή προτερήματα της endianess, ισχύει ότι: Η little endian μπορεί να διαβάσει την ίδια λέξη μερικές φορές σε διάφορα bit width, π.χ. το 4A 00 00 00 είτε διαβαστεί ως 8bit (4A), είτε ως 16 bit (004A) είναι το ίδιο (χρησιμοποιείται στους compilers). Από την άλλη, η big endian βοηθάει στο να βρεθεί πόσος μεγάλο είναι ο αριθμός διαβάζοντας μόνο την πρώτη θέση μνήμης (που φέρει το MSB). Τέλος η little-endian απλοποιεί το hardware σε πράξεις πολλαπλών Byte (αφού απαιτείται μια μόνο εντολή αύξησης inc για να διαβαστεί το επόμενο Byte).

Η ορολογία Endianess, bin-endian ή little-endian προέρχεται από το λογοτεχνικό βιβλίο 'Τα ταξίδια του Γκιούλιβερ'. Μέσα στο βιβλίο περιγράφεται η διαμάχη δύο μυθικών λαών για το πως μπορεί κάποιος να σπάσει ένα βρασμένο αυγό, είτε κτυπώντας το στην μυτερή άκρη, είτε στην μέση (δηλαδή μεγάλη/big ή μικρή/little άκρη/endianess του αυγού). Στην διαμάχη αυτή (μέσα στο λογο-

τεχνικό έργο), όσοι υποστήριζαν την μία ή την άλλη τεχνική χαρακτηρίστηκαν ως 'big-endians' ή 'little-endians' και από εκεί η ορολογία υιοθετήθηκε στην πληροφορική.

3.2.7 Ευθυγράμμιση Δεδομένων

Ένα άλλο στοιχείο που χαρακτηρίζει τους επεξεργαστές είναι η πιθανή απαίτηση των ευθυγραμμισμένων προσβάσεων (αναγνώσεων και εγγραφών) στη μνήμη. Αυτό το χαρακτηριστικό είναι παρόμοιο προς τον τρόπο αποθήκευσης των Byte (Ενότητα 3.2.6) ως προς το ότι εμφανίζεται σε λέξεις που αποτελούνται από πολλαπλά Byte. Αν η πρόσβαση αφορά μόνο 1 Byte τότε δεν υπάρχει θέμα ευθυγράμμισης. Η επίτευξη ευθυγραμμισμένων προσβάσεων μερικές φορές επιβάλλεται από τον ίδιο τον επεξεργαστή, και αν ανιχνευθεί μη ευθυγραμμισμένη πρόσβαση δημιουργείται μια προβληματική κατάσταση εξαίρεσης. Γεγονός όμως είναι, ότι οι μη ευθυγραμμισμένες προσβάσεις προκαλούν σημαντική επιβάρυνση στο χρόνο πρόσβασης (διπλασιάζεται) και για αυτό όλοι οι σύγχρονοι συμβολομεταφραστές αναδιατάσσουν τον κώδικα ή προσθέτουν στη μνήμη Byte συμπλήρωσης (memory padding) προκειμένου να ευθυγραμμιστούν οι προσβάσεις, όπως θα περιγραφεί στη συνέχεια.

Το πρόβλημα με την ευθυγράμμιση των προσβάσεων μνήμης δημιουργείται από το γεγονός ότι ο κάθε επεξεργαστής κατασκευάζεται με ένα εγγενές μήκος λέξης (native bit width, NBW) που διαχειρίζεται και μεταφέρει στις εσωτερικές και εξωτερικές μονάδες. Αυτό το μήκος λέξης χαρακτηρίζει τον επεξεργαστή: Ως σήμερα μπορεί κάποιος να χρησιμοποιήσει στα ΕΣ επεξεργαστές με 8, 16, 32 ή 64 bit. Το μήκος αυτό δηλώνει και την χωρητικότητα των εσωτερικών και εξωτερικών διαύλων του επεξεργαστή. Όταν μια μεταφορά ταυτίζεται με το NBW τότε έχει τη μέγιστη ταχύτητα εξυπηρέτησης (το μικρότερο χρόνο πρόσβασης). Για παράδειγμα, μια μεταφορά 16 bit σε έναν επεξεργαστή 16 bit είναι προτιμότερη από μια μεταφορά 8 ή 32 bit σε έναν επεξεργαστή 16 bit. Το πρόβλημα γίνεται ακόμη πιο περίπλοκο όταν υπάρχει κρυφή μνήμη, αλλά προς το παρόν θα υποθέσουμε ότι δεν υπάρχει κρυφή μνήμη.

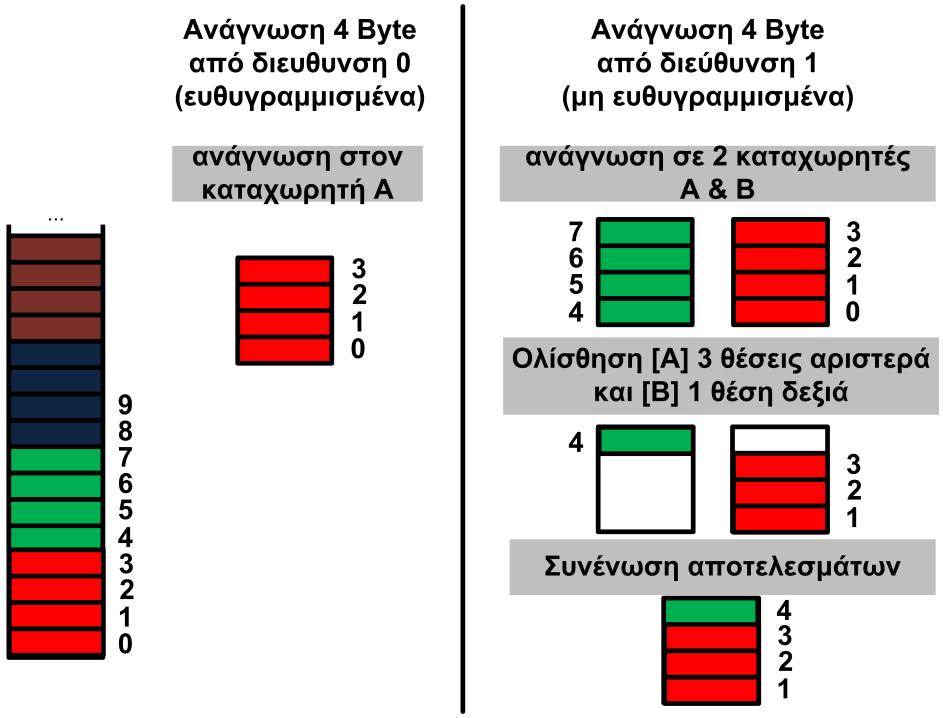
Η βέλτιστη ταχύτητα πρόσβασης στη μνήμη επιτυγχάνεται όταν οι επεξεργαστές διαβάζουν ή γράφουν με μήκος λέξης όσο το εγγενές μήκος λέξης του επεξεργαστή. Οι επεξεργαστές ομαδοποιούν τις διευθύνσεις μνήμης σε ομάδες μεγέθους τόσων Byte, όσα τα Byte του NBW. Έτσι, ένας επεξεργαστής 32 bit ομαδοποιεί τις διευθύνσεις μνήμης ανά 4 Byte· μια ομάδα φέρει τις διευθύνσεις 0h, 1h, 2h, 3h, ενώ μια άλλη τις 4h, 5h, 6h, 7h. Η ομαδοποίηση γίνεται αγνοώντας τα bit ελάχιστης σημαντικότητας που διευθυνσιοδοτούν εσωτερικά της ομάδας τα Byte. Για την περίπτωση των 32 bit, 2 bit διευθυνσιοδοτούν τα 4 Byte της ομάδας, οπότε αγνοούνται τα 2 bit ελάχιστης σημαντικότητας της διεύθυνσης μνήμης. Για 64 bit, 4 bit διευ-

θυνσιοδοτούν τα 8 Byte της ομάδας και έτσι αγνοούνται τα 8 bit ελάχιστης σημαντικότητας της διεύθυνσης μνήμης. Για παράδειγμα, η διεύθυνση ABF1h έχει αναπαράσταση στο δυαδικό σύστημα 1010101111110001 και η ομάδα σε ένα 32 bit σύστημα είναι η 10101011111100??, δηλαδή από 1010101111110000 (ABF0h) έως 1010101111110011 (ABF3h), ενώ σε ένα 64 bit η ομάδα είναι η 1010101111111????, δηλαδή από 1010101111110000 (ABF0h) έως 1010101111111111 (ABFFh). Η μικρότερη διεύθυνση ονομάζεται βάση της ομάδας, και η μεγαλύτερη κορυφή. Μερικοί σχεδιαστές εκμεταλλεύονται την ομαδοποίηση με το να απαλείψουν τα χαμηλότερα bit από τον εξωτερικό δίαυλο διευθύνσεων (καταλήγοντας σε λιγότερα καλώδια και χαμηλότερη κατανάλωση ενέργειας), αφού θα είναι πάντα 00 (η βάση της ομάδας). Έτσι σε ένα 32bit σύστημα, αντί για 32bit χρησιμοποιούν 30bit ή μπορούν να χρησιμοποιηθούν τα ίδια bit για να διευθυνσιοδοτήσουν 4 φορές περισσότερη μνήμη (π.χ. για 32bit να διευθυνσιοδοτήσουν 34 bit).

Μια πρόσβαση πολλαπλών Byte στη μνήμη θεωρείται ευθυγραμμισμένη, όταν γίνεται στη βάση της ομάδας και όχι σε κάποια ενδιάμεση διεύθυνση της ομάδας. Στο προηγούμενο παράδειγμα, μια πρόσβαση για ανάγνωση 4 Byte σε έναν επεξεργαστή 32 bit στη διεύθυνση ABF0h (που είναι η βάση της ομάδας των Byte ABF0h - ABF3h) θεωρείται ευθυγραμμισμένη, γιατί με την ίδια μοναδική πρόσβαση θα μεταφερθούν σε μια συναλλαγή και τα 4 Byte ABF0h - ABF3h. Αν όμως η πρόσβαση γίνονταν σε μια ενδιάμεση διεύθυνση της ομάδας, όπως πρόσβαση για ανάγνωση 4 Byte στη διεύθυνση ABF1h, τότε θα ήταν μη ευθυγραμμισμένη, επειδή κάποια Byte ανήκουν σε μια ομάδα και κάποια σε άλλη, δηλαδή τα Byte ABF1h, ABF2h, ABF3h ανήκουν σε μια ομάδα και θα απαιτηθεί μια συναλλαγή, ενώ το 4ο Byte (ABF4h) ανήκει στην επόμενη ομάδα και απαιτεί μια επιπρόσθετη συναλλαγή, δηλαδή έχει διπλασιαστεί ο χρόνος πρόσβασης. Τα αποτελέσματα αυτά έχουν αποδειχθεί και παρουσιαστεί στην επιστημονική κοινότητα και αποτελούν βασική γνώση για τους καλούς προγραμματιστές [20].

Προκειμένου κάποιος επεξεργαστής να υποστηρίξει μη ευθυγραμμισμένες προσβάσεις στη μνήμη θα πρέπει να έχει ειδικά κυκλώματα για να χειρίζονται αυτές τις περιπτώσεις. Συγκεκριμένα, θα πρέπει να έχει 2 προσωρινούς καταχωρητές (μη φανεροί στο χρήστη), που θα υποστηρίζουν παραμετροποιήσιμη ολίσθηση έως 3 θέσεις (για 32 bit), και ένα κύκλωμα συνένωσης των τιμών 2 καταχωρητών στο τελικό. Η Εικόνα 3.6 παρουσιάζει ένα τέτοιο παράδειγμα, όπου φαίνεται ότι η ευθυγραμμισμένη πρόσβαση γίνεται άμεσα και αποθηκεύεται η τιμή στον καταχωρητή, ενώ η μη ευθυγραμμισμένη υπόκειται σε κάποια επεξεργασία προκειμένου να ολοκληρωθεί. Η απαίτηση της επιπρόσθετης επεξεργασίας μαζί με το μειονέκτημα του διπλασιασμού του χρόνου πρόσβασης στη μνήμη, εκτός ότι καθυστερεί τον επεξεργαστή, απαιτεί και την αφοσίωση ενός αριθμού τρανζίστορ στην υλοποίηση των ειδικών κυκλωμάτων. Για αυτό

το λόγο, αρκετοί επεξεργαστές δεν υποστηρίζουν μη ευθυγραμμισμένες προσβάσεις, αφού οι αρχιτέκτονες αυτών έχουν αποφασίσει να αφιερώσουν αλλού αυτόν τον αριθμό των τρανζίστορ. Παραδείγματα επεξεργαστών που δεν υποστηρίζουν ευθυγραμμισμένες προσβάσεις είναι ο Motorola 68000 (ενώ το επόμενο μοντέλο 68020 είχε τα ειδικά αυτά κυκλώματα), ο MIPS, και οι περισσότεροι επεξεργαστές ARM και RISC. Από την άλλη μεριά βρίσκονται επεξεργαστές που υποστηρίζουν μη ευθυγραμμισμένες προσβάσεις, όπως ο PowerPC και οι επεξεργαστές που είναι συμβατοί με 8086, IA-32 και AMD64/x86-64.



Σχήμα 3.6: Η υποστήριξη για μη ευθυγραμμισμένες προσβάσεις στη μνήμη απαιτεί επιπρόσθετα κυκλώματα ολίσθησης και συνένωσης.

Εκτός από τη μείωση της ταχύτητας και την απαίτηση για επιπρόσθετο υλικό, η απαίτηση για υποστήριξη μη ευθυγραμμισμένων προσβάσεων δημιουργεί πρόβλημα στις ατομικές λειτουργίες συγχρονισμού. Όλοι οι σύγχρονοι επεξεργαστές που προορίζονται για πολύ-διαδιεργασιακά λειτουργικά συστήματα έχουν δομές συγχρονισμού ανάμεσα σε νήματα και διεργασίες. Αυτές οι δομές (όπως π.χ. οι σημαφόρες στο FreeBSD), απαιτούν ατομικές λειτουργίες ανάγνωσης ή εγγραφής στη μνήμη, δηλαδή λειτουργίες που θα εκτελεστούν ενιαία και αδιάσπαστα σε όσο το δυνατόν ελάχιστους κύκλους. Σε περίπτωση μη ευθυγραμμισμένης πρόσβασης, θα απαιτηθούν πολλαπλοί κύκλοι για την ολοκλήρωση και έτσι ενδέχεται η λειτουργία της ενημέρωσης της κοινής μεταβλητής

συγχρονισμού να διακοπεί για να εκτελεστεί κάτι άλλο, δημιουργώντας πρόβλημα συνέπειας. Για αυτό το λόγο, υπάρχει η απαίτηση στα συστήματα που υποστηρίζουν μη ευθυγραμμισμένες προσβάσεις, τουλάχιστον οι δομές συγχρονισμού να είναι ευθυγραμμισμένες.

Ακόμη και να υποστηρίζει η αρχιτεκτονική μη ευθυγραμμισμένες προσβάσεις, θα πρέπει να αποφεύγονται. Προς αυτή την κατεύθυνση κινούνται τα τελευταία χρόνια όλοι οι συμβολομεταφραστές και μόλις ανιχνεύσουν την ύπαρξη μη ευθυγράμμισης, προσθέτουν έξτρα Byte(s), ώστε να ευθυγραμμιστούν τα δεδομένα. Για παράδειγμα, αν ο χρήστης δημιουργήσει σε ένα 32 bit σύστημα την δομή που φαίνεται στον κώδικα 3.7, τότε είναι εμφανές ότι η μεταβλητή long4Byte ξεκινάει από τη διεύθυνση 1 του struct (η μεταβλητή charA1Byte ξεκινάει από τη θέση 0 του struct) και άρα κάθε φορά που γίνεται πρόσβαση στην long4Byte θα υπάρχει πρόβλημα. Για την αντιμετώπιση αυτού του θέματος, όλοι οι συμβολομεταφραστές κάνουν τη βελτιστοποίηση padding, δηλαδή την τοποθέτηση έξτρα Byte για ευθυγράμμιση. Κάποιοι compilers επίσης, μπορούν να αναδιατάσσουν τις δομές, ώστε να μειωθούν ή να εξαλειφθούν τα επιπρόσθετα Byte· π.χ. στο προηγούμενο παράδειγμα, αν η δομή long4Byte μεταφέρονταν στην αρχή του struct, τότε δε θα υπήρχε κανένα πρόβλημα μη ευθυγράμμισης (τα Byte δε χρειάζονται ευθυγράμμιση). Αναλόγως του compiler βέβαια, ίσως θα τοποθετούνταν 2 dummy Byte στο τέλος, ώστε η επόμενη δομή να είναι ευθυγραμμισμένη.

Σχήμα 3.7: Ο συμβολομεταφραστής προσθέτει dummy Byte σε μια δομή της C (A), ώστε να είναι ευθυγραμμισμένες (B)

```
1  typedef struct {
2      char     charA1Byte;
3      long     long4Byte;
4      char     charB1Byte;
5  }   Struct;
```

Σχήμα 3.8: Κώδικας A

```
1  typedef struct {
2      char     charA1Byte;
3     char   dummy0[2];
4      long     long4Byte;
5      char     charB1Byte;
6  }   Struct;
```

Σχήμα 3.9: Κώδικας B

Εκτός από την παραπάνω ευθυγράμμιση, υπάρχει η έννοια της ευθυγράμμισης και σε μικρότερα μήκη λέξεων, όταν η διεύθυνση που απαιτείται η πρόσβαση είναι η βάση μιας υποομάδας. Με παρόμοιο σκεπτικό με την ομάδα, η

υποομάδα δημιουργείται με τη διατήρηση των μεγαλύτερης σημαντικότητας ψηφίων σε μια σταθερή τιμή και την τροποποίηση μόνο των bit της χαμηλότερης σημαντικότητας. Π.χ. κάποιες ομάδες των 2 byte είναι οι διευθύνσεις (0,1), (2,3), (4,5) κ.ο.κ. αφού σε όλα διατηρούνται τα $n-1$ υψηλότερα bit της $n-bit$ λέξης σταθερά, και εναλλάσσεται μόνο το 1 bit της χαμηλότερης σημαντικότητας στις τιμές 0 και 1. Έτσι, μια πρόσβαση λέξης 16 bit στις διευθύνσεις 0 ή 1 ή 3 είναι ευθυγραμμισμένη, αφού και τα 2 Byte ανήκουν στην ίδια ομάδα, ενώ μια πρόσβαση 16 bit στις διευθύνσεις 1 ή 3 ή 5 είναι μη ευθυγραμμισμένες. Η ευθυγράμμιση προσβάσεων εμφανίζεται και στα λειτουργικά συστήματα (ΛΣ), και ιδιαίτερα στο σύστημα διαχείρισης της μνήμης κατά την αντιστοίχηση των σελίδων μνήμης σε ιδεατές ή φυσικές σελίδες.

Τέλος, σε περίπτωση που χρησιμοποιεί ο επεξεργαστής κρυφή μνήμη, τότε η ευθυγράμμιση γίνεται ως προς το μέγεθος της γραμμής της κρυφής μνήμης που συνδέεται στον επεξεργαστή (δηλαδή την κρυφή μνήμη επιπέδου 1). Για παράδειγμα, αν η κρυφή μνήμη έχει 16 Byte ανά γραμμή, τότε η ευθυγράμμιση γίνεται ανά 16 Byte και ασφαλώς ως προς τις υποομάδες που δημιουργούνται. Έτσι, μια πρόσβαση 8 Byte στις θέσεις 0 ή 8 είναι ευθυγραμμισμένη (σε όλες τις άλλες δεν είναι), μια πρόσβαση 4 Byte είναι ευθυγραμμισμένη στις θέσεις 0 ή 4 ή 8 ή 12 (σε όλες τις άλλες δεν είναι), κ.ο.κ.

3.2.8 Κρυφή Μνήμη

Κάθε πρόσβαση στην εξωτερική μνήμη επιφέρει μια μεγαλύτερη ενεργειακή και χρονική επιβάρυνση, παρά αν θα ήταν πρόσβαση σε μνήμη εντός ολοκληρωμένου κυκλώματος. Επίσης, έχει παρατηρηθεί ότι τα περισσότερα προγράμματα παρουσιάζουν μια χρονική και χωρική τοπικότητα, που μπορεί να την εκμεταλλευτεί κάποιος προς όφελος του. Η χρονική τοπικότητα στις προσβάσεις σημαίνει ότι αν ο επεξεργαστής έχει προσπελάσει μια διεύθυνση μνήμης, τότε σε σύντομο χρονικό διάστημα υπάρχει μεγάλη πιθανότητα να απαιτηθεί πάλι η προσπέλαση της ίδιας διεύθυνσης μνήμης. Η χωρική τοπικότητα στις προσβάσεις σημαίνει ότι αν ο επεξεργαστής έχει προσπελάσει μια διεύθυνση μνήμης, τότε σε σύντομο χρονικό διάστημα υπάρχει μεγάλη πιθανότητα να απαιτηθεί η προσπέλαση των γειτονικών διευθύνσεων της μνήμης. Αν λοιπόν αντιγραφούν τα δεδομένα της εξωτερικής μνήμης (και οι γειτονικές διευθύνσεις) κατά την πρόσβαση σε κάποια εσωτερική μνήμη, τότε θα επιτευχθεί μεγάλο ενεργειακό και χρονικό κέρδος. Αν μάλιστα η μνήμη αυτή λειτουργεί κρυφά (χωρίς να φαίνεται ή να παρεμβαίνει στο πρόγραμμα και χωρίς να απαιτείται ειδική λειτουργία για να χρησιμοποιηθεί), τότε τα υπάρχοντα προγράμματα θα επωφεληθούν άμεσα. Έτσι λοιπόν, δημιουργήθηκε μια τέτοια μνήμη και ονομάζεται ‘κρυφή μνήμη’ (cache memory). Οι σημερινοί επεξεργαστές δεν έχουν μόνο μια τέτοια μνήμη, αλλά μια ιεραρχία από τέτοιες μνήμες, για λόγους που θα εξηγη-

θούν στη συνέχεια.

Η κρυφή μνήμη λειτουργεί ως εξής: Όταν θέλουμε ο επεξεργαστής να προσπελάσει (να διαβάσει ή εγγράψει) μια διεύθυνση κύριας μνήμης, πρώτα ελέγχουμε για την ύπαρξη αντιγράφου των δεδομένων της τοποθεσίας αυτής στην κρυφή μνήμη. Αν έχει γίνει αντιγραφή στην κρυφή μνήμη, τότε υπάρχει επιτυχία κρυφής μνήμης (cache hit) και μπορεί να ολοκληρωθεί άμεσα η πρόσβαση. Αν όμως δεν έχει γίνει αντιγραφή, τότε υπάρχει αστοχία κρυφής μνήμης (cache miss) και πρέπει πρώτα να διαβάσουμε τη διεύθυνση (και ίσως και κάποιες γειτονικές διευθύνσεις) και να μεταφέρουμε τα δεδομένα μέσα στην κρυφή μνήμη. Μόλις ολοκληρωθεί η περιγραφή, ο επεξεργαστής μπορεί να διαβάσει τα δεδομένα από την κρυφή μνήμη. Σημειώστε ότι αναφερόμαστε σε προσπελάσεις ανάγνωσης και όχι εγγραφής (δεν υπάρχει κέρδος ή νόημα να επαναχρησιμοποιούνται δεδομένα που γράφονται συνεχώς).

Η περιγραφή αυτή της λειτουργίας της κρυφής μνήμης οδηγεί σε διάφορες επιλογές σχεδιασμού κρυφής μνήμης: χαρτογράφηση κρυφής μνήμης, πολιτική αντικατάστασης της κρυφής μνήμης και τεχνικές εγγραφής στην κρυφή μνήμη. Αυτές οι επιλογές σχεδιασμού μπορεί να έχουν πολλαπλές επιδράσεις στο κόστος του συστήματος, την απόδοσή του καθώς και στην κατανάλωση ενέργειας και για αυτό θα έπρεπε να αξιολογηθούν προσεκτικά για μια δεδομένη εφαρμογή σε ένα ΕΣ.

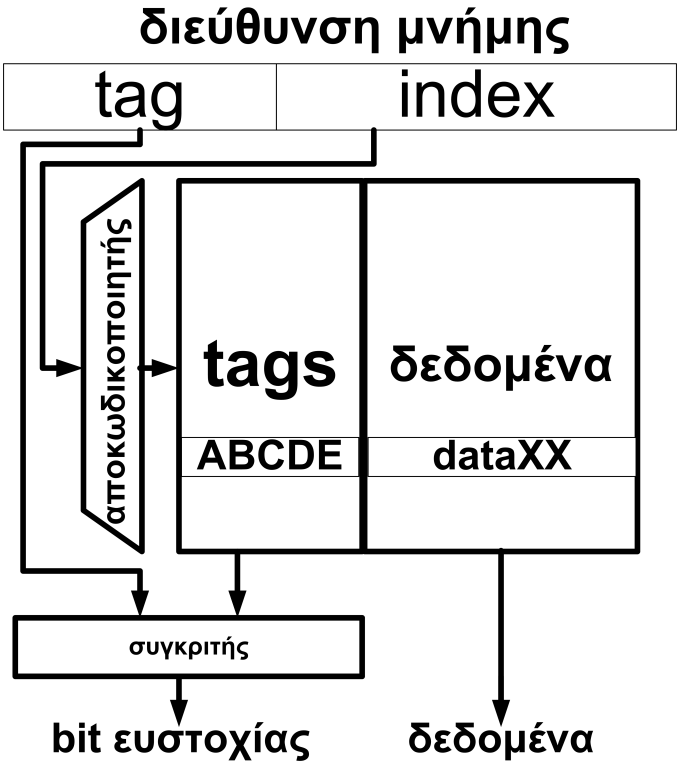
Τεχνικές χαρτογράφησης κρυφής μνήμης

Η χαρτογράφηση κρυφής μνήμης είναι μια μέθοδος για την εκχώρηση των διευθύνσεων της κυρίας μνήμης στον πολύ μικρότερο αριθμό διαθέσιμων διευθύνσεων της κρυφής μνήμης, και για να προσδιοριστεί αν το περιεχόμενο της συγκεκριμένης διεύθυνσης κύριας μνήμης είναι στην κρυφή μνήμη. Μια κρυφή μνήμη μπορεί να έχει π.χ. 128 διευθύνσεις οι οποίες θα πρέπει να αντιστοιχηθούν σε χιλιάδες ή εκατομμύρια διευθύνσεις της εξωτερικής μνήμης RAM. Η χαρτογράφηση της κρυφής μνήμης μπορεί να επιτευχθεί με τη χρήση μιας από τις ακόλουθες τρεις βασικές τεχνικές:

1. Άμεση χαρτογράφηση ή απευθείας απεικόνιση (direct mapped): Σε αυτή τη τεχνική, η διεύθυνση της κύριας μνήμης διαιρείται σε δύο περιοχές, τα bit ευρετηρίου (index) και τις ετικέτας (tag). Είναι η απλούστερη οργάνωση και για αυτό έχει και τη χαμηλότερη κατανάλωση ενέργειας λειτουργίας (αλλά και τη μικρότερη απόδοση). Το ευρετήριο αντιπροσωπεύει την διεύθυνση της κρυφής μνήμης και επομένως ο αριθμός των bits αυτού καθορίζεται από το μέγεθος της κρυφής μνήμης, με τη σχέση: μέγεθος ευρετηρίου = log_2(μέγεθος κρυφής μνήμης). Για παράδειγμα, αν το μέγεθος της κρυφής μνήμης είναι 128 (από 0 έως 127) γραμμές, τότε ισχύει

$2^7 = 128$ και άρα απαιτούνται 7 bit. Τα υπόλοιπα bit της διεύθυνσης μνήμης αποτελούν την ετικέτα, αφού όπως γίνεται κατανοητό πολλές διαφορετικές διευθύνσεις κύριας μνήμης θα χαρτογραφηθούν στην ίδια γραμμή της κρυφής μνήμης. Έτσι, όταν αποθηκεύουμε το περιεχόμενο μιας διεύθυνσης κύριας μνήμης στην κρυφή μνήμη αποθηκεύουμε επίσης και την αντίστοιχη ετικέτα. Για να καθορίσουμε αν μια επιθυμητή διεύθυνση κύριας μνήμης είναι στην κρυφή μνήμη, πηγαίνουμε στη διεύθυνση της κρυφής μνήμης, που υποδεικνύεται από τον δείκτη, και έπειτα συγκρίνουμε την ετικέτα αυτή με την επιθυμητή ετικέτα.

Η κρυφή μνήμη απευθείας απεικόνισης για μέγεθος γραμμής 1 byte, φαίνεται στην Εικόνα 3.10. Συγκεκριμένα, φαίνεται ο διαχωρισμός της διεύθυνσης σε tag και index. Τα bit του index καθορίζουν τη γραμμή της κρυφής μνήμης που θα συγκριθεί το tag. Αν σε αυτή τη γραμμή το tag είναι το ίδιο με αυτό που προκύπτει από την αρχική διεύθυνση, τότε η κρυφή μνήμη φέρει έγκυρα δεδομένα για την τρέχουσα πρόσβαση και τα προωθεί στον επεξεργαστή.

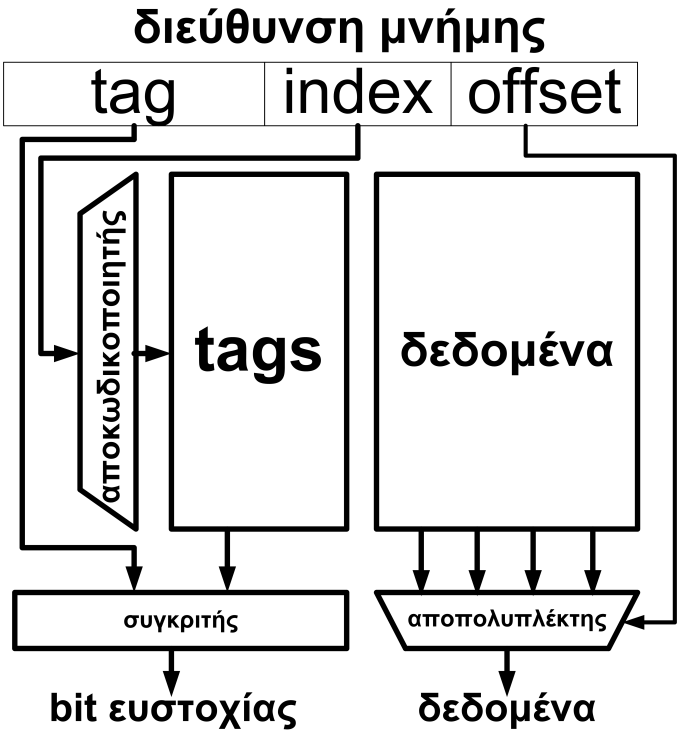


Σχήμα 3.10: Οργάνωση της κρυφής μνήμης άμεσης απεικόνισης όπου κάθε γραμμή κρυφής μνήμης φέρει 1 Byte.

Ένα παράδειγμα χρήσης είναι το παρακάτω. Έστω ότι η κρυφή μνήμη έχει 64 Byte και γραμμές του ενός Byte. Αυτό σημαίνει ότι συνολικά υπάρχουν 64 γραμμές κρυφής μνήμης (από 0 έως 63), και άρα απαιτούνται 6 bit ($2^6 = 64$). Αν τα bit της αρχικής διεύθυνσης ήταν 12 (που σημαίνει ότι το σύστημα διευθυνσιοδοτούσε συνολική εξωτερική μνήμη $2^{12} = 4096$ Bytes, τότε το tag θα είχε 4 bit (τα πιο σημαντικά bit) και το index 6 bit (τα χαμηλότερης σημαντικότητας bit). Άρα μια διεύθυνση 10 $4Fh$ που έχει δυαδική αναπαράσταση $010010101111b$ θα χωρίζονταν στα tag bits $010010b$ και στα index bits $101111b$. Η κρυφή μνήμη θα εξέταζε τη γραμμή $101111b$, δηλαδή τη γραμμή 47 και αν εκεί έβρισκε το tag $101111b$ τότε θα υπήρχε ευστοχία και θα προωθούνταν τα δεδομένα άμεσα (μέσα σε ένα κύκλο ρολογιού) στον επεξεργαστή, διαφορετικά αν το tag ήταν διαφορετικό, τότε θα υπήρχε αστοχία.

Η παραπάνω δομή εκμεταλλεύεται τη χρονική τοπικότητα. Όμως, προκειμένου να υπάρξει καλύτερη εκμετάλλευση και της χωρικής τοπικότητας, μια κρυφή μνήμη δε φέρει μόνο 1 byte σε μια γραμμή, αλλά πολλαπλά Byte (από 2 έως 16). Αυτά τα byte έχουν το ίδιο tag και έτσι μεταφέρονται ομαδικά από τη μνήμη, κάτι που είναι εύκολο και γρήγορο, αφού όλες οι σύγχρονες μνήμες υποστηρίζουν γρήγορες μεταφορές ριπής (burst transfers). Όταν η κρυφή μνήμη έχει πολλαπλά Byte σε μια γραμμή, θα πρέπει με κάποιον τρόπο να προσδιορίζεται το Byte που απαιτεί ο επεξεργαστης. Αυτό επιτυγχάνεται με τη μετατόπιση (offset), που καταλαμβάνει τα bit χαμηλότερης σημαντικότητας. Αναλόγως πόσα Byte φέρει η κρυφή γραμμή, τόσα επιλέγονται τα bit που μπορούν να διευθυνσιοδοτήσουν μοναδικά τα Byte. Για παράδειγμα, αν η κάθε γραμμή φέρει 4 Byte, τότε απαιτούνται 2 bit, αφού $2^2 = 4$, ενώ αν φέρει 16 απαιτούνται 4 bit ($2^4 = 16$). Η Εικόνα 3.11 δείχνει την εσωτερική δομή μιας κρυφής μνήμης άμεσης απεικόνισης με πολλαπλά Byte ανά γραμμή. Στο παράδειγμα της προηγούμενης παραγράφου, αν η κρυφή μνήμη είχε 2 Byte ανά γραμμή, τότε θα είχε συνολικά $\frac{64}{2}$ γραμμές των 2 Byte, δηλαδή 32 γραμμές, και άρα θα απαιτούνταν 5 bit ($2^5 = 32$) στο index για τη διευθυνσιοδότηση, 1 bit στο offset, και τα υπόλοιπα στο tag, δηλαδή 4 bit (αν το μήκος διεύθυνσης της μνήμης ήταν 12bit). Σε αυτή την περίπτωση, μια διεύθυνση $4Fh$ που έχει δυαδική αναπαράσταση $010010101111b$ θα χωρίζονταν στα tag bits $010010b$, στα index bits $10111b$ και στο offset bit 1b (που είναι το LSB). Η κρυφή μνήμη θα εξέταζε τη γραμμή $10111b$, δηλαδή τη γραμμή 23 και αν εκεί έβρισκε το tag $101111b$, θα υπήρχε ευστοχία και με τον αποπλέκτη θα επιλέγονταν το Byte στη 2η θέση της γραμμής με index bit 1. Αυτό το Byte θα προωθούνταν στον επεξεργαστή. Αν το tag δεν ταίριαζε, τότε θα έπρεπε να μεταφέρει από την εξωτερική μνήμη, όλα τα Byte της γραμμής

(σε αυτό το παράδειγμα 2 Byte), να ενημερώσει κατάλληλα το tag και να τα προωθήσει στον επεξεργαστή.



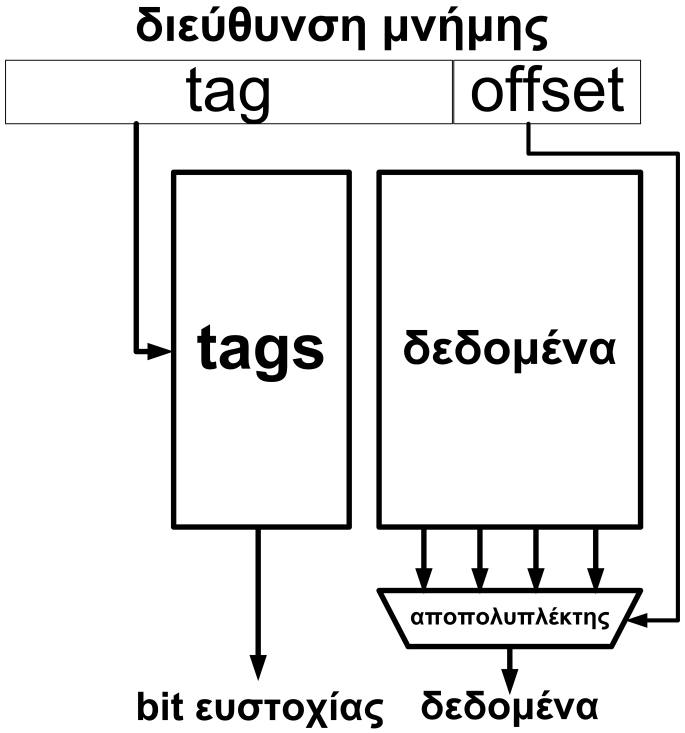
Σχήμα 3.11: Οργάνωση της κρυφής μνήμης άμεσης απεικόνισης με πολλαπλά Byte ανά γραμμή.

Όπως φαίνεται, το πρόβλημα είναι ότι πολλαπλές διευθύνσεις της εξωτερικής μνήμης συναγωνίζονται για την ίδια γραμμή της κρυφής μνήμης, αφού η αντιστοίχηση είναι 1-προς-1. Είναι πιο απλή και γρήγορη (όταν υπάρχει ευστοχία) από τις υπόλοιπες αρχιτεκτονικές κρυφής μνήμης.

Μία τυπική κρυφή μνήμη άμεσης απεικόνισης μπορεί να αποθηκεύει 8 kilobytes δεδομένων σε γραμμές των 16 Byte. Συνεπώς, θα μπορούσαν να υπάρχουν 512 γραμμές. Μία διεύθυνση των 32 bits θα μπορούσε να έχει 4 bit για το offset και 9 bit για το index (επιλογή γραμμής), αφήνοντας τα υπόλοιπα 19 bit για το tag, το οποίο απαιτεί μόλις λίγο πιο πάνω από ένα kilobyte για την αποθήκευση (19 bit $*2^9$ γραμμές $= 9727$ bit ή 1215 Byte). Παρατηρούμε λοιπόν ότι όσο μεγαλύτερο είναι το tag, τόσο αυξάνονται και οι αποθηκευτικές ανάγκες της κρυφής μνήμης.

2. Χαρτογράφηση (ή απεικόνιση) πλήρους συσχέτισης (fully associative): Σε αυτή τη τεχνική, κάθε διεύθυνση κρυφής μνήμης περιέχει όχι μόνο το πε-

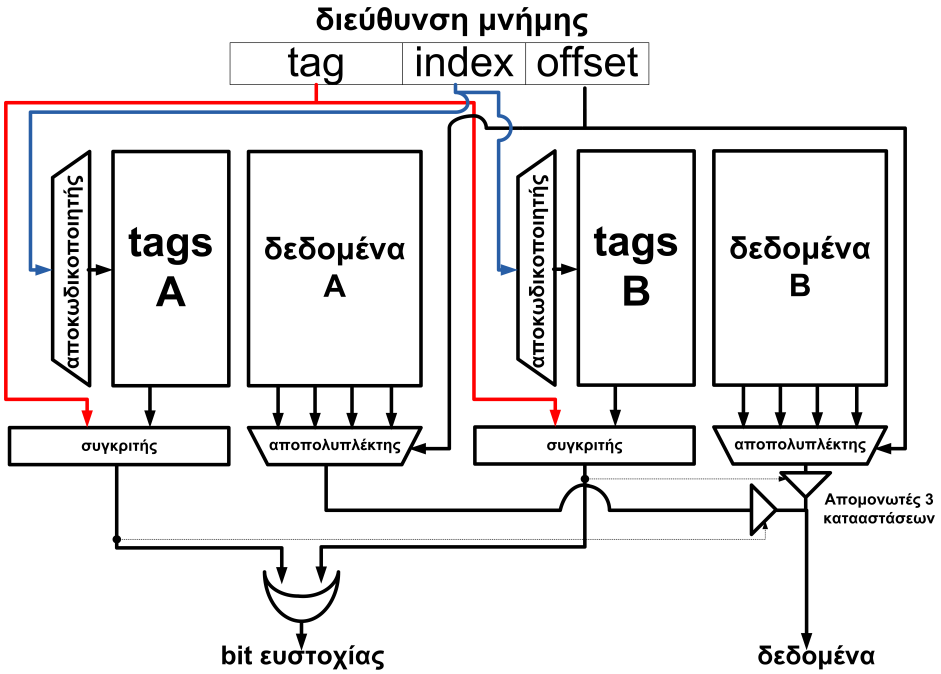
ριεχόμενο της διεύθυνση κύριας μνήμης αλλά και ολόκληρη τη διεύθυνση αυτής. Για να προσδιορίσουμε αν μια επιθυμητή διεύθυνση κύριας μνήμης είναι στην κρυφή μνήμη, συγκρίνουμε ταυτόχρονα (συσχετιζόμενα) όλες τις αποθηκευμένες στην κρυφή μνήμη διευθύνσεις με την επιθυμητή διεύθυνση. Σε αυτήν την περίπτωση οποιαδήποτε διεύθυνση της κρυφής μνήμης μπορεί να τοποθετηθεί σε οποιαδήποτε γραμμή της κρυφής μνήμης. Έχει πάρα πολύ αυξημένες ανάγκες σε υλικό (απαιτούνται πάρα πολλοί συγκριτές), μεγάλη κατανάλωση ενέργειας (λόγω των πολλών συγκρίσεων), αλλά πολύ καλό ποσοστό ευστοχίας. Εντούτοις, λόγω του μεγάλου κόστους δε χρησιμοποιείται. Η Εικόνα 3.12 παρουσιάζει την οργάνωση μιας τέτοιας κρυφής μνήμης.



Σχήμα 3.12: Οργάνωση της κρυφής μνήμης πλήρους συσχέτισης.

3. Χαρτογράφηση (ή απεικόνιση) με συσχέτιση σε ομάδες (set associative): Η τεχνική αυτή αποτελεί έναν συμβιβασμό μεταξύ των δυο παραπάνω τεχνικών. Όπως στην άμεση χαρτογράφηση, έτσι και εδώ, το index χαρτογραφεί κάθε διεύθυνση κύριας μνήμης σε μια διεύθυνση κρυφής μνήμης, αλλά τώρα κάθε διεύθυνση κρυφής μνήμης περιλαμβάνει το περιεχόμενο και τις ετικέτες δύο ή περισσότερων τοποθεσιών μνήμης, που ονομάζονται ομάδες (sets). Για να προσδιορίσουμε αν μια επιθυμητή διεύθυνση

κύριας μνήμης είναι στην κρυφή μνήμη, πηγαίνουμε στη διεύθυνση της κρυφής μνήμης που υποδεικνύεται από το index, και έπειτα ταυτόχρονα (συσχετιζόμενα) συγκρίνουμε όλες τις ετικέτες στην τοποθεσία με την επιθυμητή ετικέτα. Η κρυφή μνήμη με ένα σύνολο N ονομάζεται N-δρόμων ομαδικά συσχετιζόμενη κρυφή μνήμη. Τυπικές δομές κρυφής μνήμης είναι 2-δρόμων, 4-δρόμων και 8-δρόμων. Το Σχήμα 3.13 παρουσιάζει μια δομή κρυφής μνήμης δυο δρόμων. Όπως και στην κρυφή μνήμη απευθείας απεικόνισης, η διεύθυνση της εξωτερικής μνήμης χωρίζεται σε tag|index|offset. Το index σε αυτή την περίπτωση, δείχνει τη γραμμή και στα 2 σετ. Μόλις βρεθεί η γραμμή συγκρίνεται το αρχικό tag ταυτόχρονα και με τα tag από τα 2 σύνολα. Αν βρεθεί σε ένα από αυτά τότε προωθούνται τα δεδομένα προς τον επεξεργαστή από το αντίστοιχο σύνολο. Αν δε βρεθεί σε κανένα, τότε η κρυφή μνήμη θα μεταφέρει από την εξωτερική μνήμη τα δεδομένα και θα τα αποθηκεύσει σε ένα από τα δυο σύνολα. Η επιλογή θα γίνει με κάποιον αλγόριθμο, όπως θα αναφερθεί στη συνέχεια.



Σχήμα 3.13: Οργάνωση της κρυφής μνήμης συσχέτισης κατά ομάδων δυο δρόμων.

Αυξάνοντας το επίπεδο της πολυπλοκότητας, η συσχέτιση κατά ομάδες στοχεύει να μειωθούν τα προβλήματα αντιστοίχησης που υπάρχουν στην απευθείας απεικόνισης μνήμη, δυνατότητα σε ένα συγκεκριμένο αντικείμενο της μνήμης να αποθηκευτεί σε περισσότερες από μια τοποθεσίες

της κρυφής μνήμης. Μία διεύθυνση εξωτερικής μνήμης που παρουσιάζεται στην κρυφή μνήμη μπορεί να έχει δεδομένα σε οποιοδήποτε σύνολο. Σε περίπτωση που υπήρχε διαμάχη αντιστοίχησης (όπως συνέβαινε στην κρυφή μνήμη απευθείας απεικόνισης) από διευθύνσεις που απεικονίζονταν στην ίδια γραμμή, τώρα αυτή η διαμάχη εξαλείφεται αφού στην ίδια γραμμή υπάρχουν πολλαπλές επιλογές (σε άλλα σετ). Σε σύγκριση με το προηγούμενο παράδειγμα, μια διεύθυνση των 32 bit θα μπορούσε να έχει 4 bit για το offset και 8 bit για το index (όχι 9 όπως στην απευθείας απεικόνιση), επειδή το ίδιο μέγεθος μνήμης διαιρείται σε 2 σετ, αφήνοντας 20 bit για το tag. Τα tag bit απαιτούν (20 bit $*2^8$ γραμμές $= 5120$ bit για κάθε set, και άρα συνολικά 10240 bit ή 1280 Byte, δηλαδή λίγο πάνω από ένα KB (η ίδια μεγέθους μνήμη απευθείας απεικόνισης απαιτούσε 1215 Byte). Παρατηρούμε λοιπόν ότι όσο αυξάνεται ο βαθμός συσχέτισης, αυξάνονται και οι αποθηκευτικές ανάγκες της κρυφής μνήμης. Στην ακραία περίπτωση της πλήρης συσχέτισης, το μέγεθος για τα tags είναι 28bit $*2^9$ γραμμές $= 14336$ ή 1792 Byte. Οι κρυφές μνήμες μερικής συσχέτισης, έχουν καλά ποσοστά ευστοχίας και για αυτό χρησιμοποιούνται τόσο σε ΕΣ όσο και στα υπόλοιπα υπολογιστικά συστήματα.

Ως προς το χρόνο, όσο πιο μεγάλη είναι η συσχέτιση, τόσο μεγαλύτερη καθυστέρηση έχει. Η αύξηση στο χρόνο γίνεται εξαιτίας της ανάγκης να πολλαπλασιαστούν τα δεδομένα από τα δύο σετ. Όταν ένα νέο δεδομένο τοποθετείται στην κρυφή μνήμη, πρέπει να ληφθεί μια απόφαση, σε ποιο από τα δύο σετ θα τοποθετηθεί. Η πολιτική αντικατάστασης θα περιγραφεί στην επόμενη ενότητα.

Η κρυφή μνήμη κρατάει μια εγγραφή για να ξέρει ποιο ζευγάρι ποιας τοποθεσίας είχε τοποθετηθεί τελευταία ώστε να τοποθετήσει τα νέα δεδομένα σε μια άλλη τοποθεσία. Η συσχέτιση κατά ομάδες απαιτεί τουλάχιστον δύο επιπλέον δρόμους για κάθε βαθμό συσχέτισης, αλλά πρακτικά τα πλεονεκτήματα της συσχέτισης μετά από 4 δρόμους είναι λιγότερα και δεν δίνουν βάση για περαιτέρω αύξηση της συσχέτισης.

Στην άλλη ειδική συσχέτιση, είναι πιθανόν να σχεδιαστεί μια κρυφή μνήμη συσχέτισης στην τεχνολογία VLSI. Θα ήταν βέβαια προτιμότερο να διαιρεθεί η κρυφή μνήμη άμεσης απεικόνισης σε ακόμη μικρότερα συστατικά μέρη. Τα αποθηκευμένα δεδομένα σχεδιάζονται διαφορετικά χρησιμοποιώντας την κρυφή μνήμη διευθυνσιοδοτούμενη από δεδομένα (Content Addressable Memory - CAM). Ένα δομοστοιχείο της CAM είναι παρόμοιο με αυτό της RAM με έναν ενσωματωμένο συγκριτή, έτσι ώστε η βάση δεδομένων αποθηκευμένων στοιχείων της CAM να μπορεί να εκτελέσει μια παράλληλη έρευνα για να τοποθετηθεί μια διεύθυνση σε κάθε τοποθεσία (πλήρους συσχέτισης).

3.2.9 Πολιτική αντικατάστασης κρυφής μνήμης

Η πολιτική αντικατάστασης της κρυφής μνήμης είναι η τεχνική για την επιλογή της γραμμής της κρυφής μνήμης που θα αντικατασταθεί όταν μια συσχετιστική κρυφή μνήμη είναι πλήρης. Ασφαλώς η πολιτική αυτή δε χρειάζεται σε μια άμεσης απεικόνισης κρυφής μνήμης, αφού υπάρχει αντιστοίχηση πάντα 1 προς 1 (1 διεύθυνση εξωτερικής μνήμης πάντα αντιστοιχεί σε μια γραμμή της κρυφής μνήμης). Υπάρχουν τρεις συνηθισμένες πολιτικές αντικατάστασης. Μια πολιτική τυχαίας αντικατάστασης (random) επιλέγει τυχαία το σετ που θα αντικατασταθεί. Όσο απλή κι αν είναι στην υλοποίηση της, αυτή η πολιτική δεν κάνει τίποτα για να αποτρέψει την αντικατάσταση κάποιου στοιχείου που είναι πιθανό να χρησιμοποιηθεί σύντομα ξανά. Η πολιτική αντικατάστασης του λιγότερο χρησιμοποιούμενου στοιχείου (least recently used) αντικαθιστά το στοιχείο που δεν έχει προσπελαστεί για το μεγαλύτερο χρονικό διάστημα, υποθέτοντας βέβαια ότι αυτό σημαίνει δεν είναι πιθανό να προσπελαστεί στο άμεσο μέλλον. Η πολιτική αυτή εξασφαλίζει ένα σημαντικό λόγο ευστοχίας έναντι αστοχίας, αλλά απαιτεί ακριβό υλικό για να παρακολουθεί τους χρόνους προσπέλασης των δομικών στοιχείων. Η πολιτική κατά την οποία αυτό που είχε έρθει πρώτο, θα αποχωρήσει και πρώτο (FIFO), χρησιμοποιεί μια ουρά μεγέθους και σπρώχνει κάθε διεύθυνση στοιχείου στην ουρά, όταν η διεύθυνση αυτή προσπελαύνεται, και έπειτα επιλέγει από την κορυφή το στοιχείο που θα αντικαταστήσει μετακινώντας τα υπόλοιπα στοιχεία της ουράς.

3.2.10 Τεχνικές εγγραφής στην κρυφή μνήμη

Όταν εγγράφουμε στην κρυφή μνήμη πρέπει να ενημερώνουμε την εξωτερική μνήμη ανά τακτά χρονικά διαστήματα. Η ενημέρωση αυτή αποτελεί ένα ζήτημα για τα δεδομένα της κρυφής μνήμης από τη στιγμή που η κρυφή μνήμη εξ ορισμού είναι μόνο για ανάγνωση. Υπάρχουν δύο τεχνικές: (i) άμεσης εγγραφής ή εγγραφής διαμέσου (write-through) και εγγραφής με καθυστέρηση (write-back).

Στην τεχνική άμεσης εγγραφής, όποτε εγγράφουμε στην κρυφή μνήμη εγγράφουμε επίσης και στην κυρία μνήμη, γεγονός που απαιτεί από τον επεξεργαστή αναμονή έως ότου η εγγραφή στην κύρια μνήμη ολοκληρωθεί. Όντας εύκολη να υλοποιηθεί, η τεχνική αυτή μπορεί να έχει ως αποτέλεσμα διάφορες περιττές εγγραφές στην κύρια μνήμη. Για παράδειγμα, υποθέτουμε ότι ένα πρόγραμμα εγγράφει σε ένα στοιχείο της κρυφής μνήμης, μετά το διαβάζει και μετά το ξαναγράφει με το δομικό στοιχείο να παραμένει στην κρυφή μνήμη κατά τη διάρκεια αυτών των τριών προσπελάσεων. Δεν θα υπήρχε ανάγκη να ανανεωθεί η κύρια μνήμη μετά την πρώτη εγγραφή, από τη στιγμή που η δεύτερη εγγραφή επικαλύπτει την πρώτη. Όμως, επειδή χρησιμοποιείται η εγγραφή διαμέσου, τότε μεταφέρεται σε κάθε εγγραφή η τιμή στην αργή εξωτερική μνήμη.

Η τεχνική της εγγραφής με καθυστέρηση μειώνει τον αριθμό των εγγραφών στην κύρια μνήμη εγγράφοντας ένα στοιχείο στην κύρια μνήμη μόνο όταν το στοιχείο αντικαθίσταται (από την πολιτική αντικατάστασης). Διαφορετικά παραμένει στη κρυφή μνήμη. Η τεχνική αυτή απαιτεί τη σύνδεση ενός επιπλέον bit, που ονομάζεται ακάθαρτο bit (dirty), σε κάθε tag. Μόλις γίνει μια εγγραφή, τότε το bit ακαθαρσίας από 0 γίνεται 1, που σημαίνει ότι αν ποτέ αυτή η γραμμή της κρυφής μνήμης απομακρυνθεί θα πρέπει να μεταφερθούν τα δεδομένα στην εξωτερική μνήμη.

Σε συστήματα που αποτελούνται από πολλούς επεξεργαστές, η τεχνική write-back εγκυμονεί κινδύνους ασυνέπειας της εικόνας που έχουν οι επεξεργαστές για κάθε διεύθυνση μνήμης, επειδή υπάρχει η δυνατότητα κάποιος επεξεργαστής να έχει τροποποιήσει κάποια διεύθυνση μνήμης και να μην έχει μεταφέρει την αλλαγή στην εξωτερική μνήμη. Αν ένας άλλος επεξεργαστής διαβάσει από την εξωτερική μνήμη, τότε θα δει άλλη τιμή. Παρόμοιο πρόβλημα εμφανίζεται και με την εγγραφή διαμέσου, αφού ένας επεξεργαστής μπορεί να έχει διαβάσει και μεταφέρει μια διεύθυνση της εξωτερικής μνήμης στην κρυφή του μνήμη, ένας άλλος επεξεργαστής να τροποποιήσει τα δεδομένα στην εξωτερική μνήμη, αλλά ο πρώτος να συνεχίζει να χρησιμοποιεί την παλαιά τιμή των δεδομένων που έχει η κρυφή μνήμη. Ασφαλώς, αυτά τα προβλήματα έχουν λυθεί στους σημερινούς πολυπύρηνους επεξεργαστές με διάφορα επιπρόσθετα πρωτόκολλα που αναλύονται στο σχετικό μάθημα των 'Συστημάτων Παράλληλης και Κατανεμημένης Επεξεργασίας'.

Συνοψίζοντας λοιπόν, οι μνήμες αποθηκεύουν δεδομένα για να χρησιμοποιηθούν από τους επεξεργαστές. Η ROM τυπικά μόνο διαβάζεται από ένα ενσωματωμένο σύστημα. Μπορεί να προγραμματιστεί κατά τη διαδικασία της κατασκευής (προγραμματισμός με χρήση μάσκας) ή από τον χρήστη (προγραμματιζόμενη ROM ή PROM). Η PROM μπορεί να έχει δυνατότητα διαγραφής χρησιμοποιώντας ακτίνες UV (EPROM), ή ηλεκτρικά διαγράψιμη (EEPROM). Η RAM από την άλλη, είναι μνήμη που μπορεί να διαβαστεί ή να εγγραφεί από ένα ενσωματωμένο σύστημα. Η στατική RAM χρησιμοποιεί flip-flop για να αποθηκεύσει κάθε bit, ενώ η δυναμική RAM χρησιμοποιεί ένα τρανζίστορ κι ένα πυκνωτή, με αποτέλεσμα λιγότερα τρανζίστορ αλλά και με την ανάγκη ανανέωσης του φορτίου του πυκνωτή και πιο αργή απόδοση. Η ψευδοστατική RAM είναι μια δυναμική RAM (DRAM) με ενσωματωμένο ελεγκτή ανανέωσης. Οι σχεδιαστές πρέπει όχι μόνο να επιλέγουν τα κατάλληλα είδη μνημών για ένα δεδομένο ενσωματωμένο σύστημα, αλλά πρέπει και να συνδέουν συχνά τις μικρότερες μνήμες στις μεγαλύτερες, σε μια ιεραρχία μνήμης. Με τη χρήση της ιεραρχίας της μνήμης μπορούμε να βελτιώσουμε την απόδοση του συστήματος κρατώντας αντίγραφα των συχνά προσπελασμένων οδηγιών/δεδομένων σε μικρές και γρήγορες μνήμες κοντά στον επεξεργαστή. Η κρυφή μνήμη είναι μία μικρή και γρήγορη μνήμη μεταξύ ενός επεξεργαστή και της κύριας μνήμης. Αρ-

κετά χαρακτηριστικά σχεδιασμού της κρυφής μνήμης επηρεάζουν σημαντικά την ταχύτητα και το κόστος της κρυφής μνήμης, συμπεριλαμβανομένων και της χαρτογράφησης, των πολιτικών αντικατάστασης και των τεχνικών εγγραφής.

3.2.11 Ιεραρχία μνήμης

Ένας σύγχρονος μικροεπεξεργαστής μπορεί να εκτελέσει τις εντολές με υψηλό ρυθμό. Για να εκμεταλλευτεί πλήρως αυτή τη δυνατή απόδοση ο επεξεργαστής, πρέπει να συνδεθεί με ένα σύστημα μνήμης το οποίο είναι και πολύ μεγάλο και πολύ γρήγορο. Εάν η μνήμη είναι πάρα πολύ μικρή, δε θα είναι σε θέση να κρατήσει αρκετά προγράμματα, έτσι ώστε να κρατήσει τον επεξεργαστή απασχολημένο. Εάν είναι πάρα πολύ αργή, η μνήμη δεν θα είναι σε θέση να παρέχει εντολές τόσο γρήγορα όσο ο επεξεργαστής μπορεί να τις εκτελέσει.

Δυστυχώς, όσο μεγαλύτερη είναι μια μνήμη τόσο πιο αργή είναι. Δεν είναι επομένως δυνατό, να σχεδιαστεί μια μόνο μνήμη που να είναι και αρκετά μεγάλη και αρκετά γρήγορη, για να κρατήσει απασχολημένο έναν επεξεργαστή υψηλής απόδοσης.

Εντούτοις, είναι δυνατό να κατασκευαστεί ένα σύνθετο σύστημα μνήμης που συνδυάζει μια μικρή, γρήγορη μνήμη και μια μεγάλη, αργή κύρια μνήμη για να παρουσιάσει μια εξωτερική συμπεριφορά, η οποία με τις χαρακτηριστικές στατιστικές προγράμματος εμφανίζεται να συμπεριφέρεται, τις περισσότερες φορές, όπως μια μεγάλη, γρήγορη μνήμη. Το μικρό, γρήγορο συστατικό μνήμης είναι η κρυφή μνήμη (cache), η οποία αυτόματα διατηρεί τα αντίγραφα των εντολών και των δεδομένων που ο επεξεργαστής χρησιμοποιεί πολύ συχνά.

Η αποτελεσματικότητα της κρυφής μνήμης εξαρτάται από τις ιδιότητες της χωρικής και χρονικής τοποθεσίας του προγράμματος.

Αυτή η αρχή μνήμης δύο-επιπέδων μπορεί να επεκταθεί σε μια ιεραρχία μνήμης πολλών επιπέδων, και ο σκληρός δίσκος του υπολογιστή μπορεί να αναγνωρισθεί ως τμήμα αυτής της ιεραρχίας. Με την κατάλληλη υποστήριξη διαχείρισης μνήμης, το μέγεθος ενός προγράμματος είναι περιορισμένο όχι από την κύρια μνήμη του υπολογιστή, αλλά από το μέγεθος του σκληρού δίσκου, το οποίο μπορεί να είναι πολύ μεγαλύτερο από αυτό της κύριας μνήμης.

3.2.12 Μέγεθος και ταχύτητα μνήμης

Μία χαρακτηριστική ιεραρχία μνήμης υπολογιστών περιλαμβάνει διάφορα επίπεδα, με κάθε επίπεδο να κατέχει χαρακτηριστικό μέγεθος και ταχύτητα.

- Οι καταχωρητές του επεξεργαστή μπορούν να εκληφθούν ως η κορυφή της ιεραρχίας μνήμης. Ο επεξεργαστής RISC τυπικά θα έχει περίπου τριάντα δύο 32 bit καταχωρητές. Αν έχει 32 καταχωρητές των 4 Byte θα έχει

ένα σύνολο 128 bytes, με έναν χρόνο πρόσβασης μερικών νανοδευτερολέπτων (nanoseconds).

- Η κρυφή μνήμη σε ένα ΕΣ πάνω στο τσιπ θα έχει χωρητικότητα από οκτώ έως 32 kbytes, με χρόνο πρόσβασης περίπου δέκα νανοδευτερόλεπτα.
- Τα υψηλής απόδοσης υπολογιστικά συστήματα γραφείου μπορούν να έχουν παραπάνω από ένα επίπεδο κρυφής μνήμης. Το δεύτερο επίπεδο θα έχει χωρητικότητα μερικών εκατοντάδων kbytes και χρόνο πρόσβασης μερικών δεκάδων νανοδευτερολέπτων.
- Η κύρια μνήμη θα είναι από κάποια KB έως δεκάδες MB δυναμικής RAM, με χρόνο πρόσβασης περίπου 100 νανοδευτερόλεπτα.
- Μετά τη μνήμη RAM υπάρχει ο χώρος μόνιμης αποθήκευσης σε δίσκο ή σε μνήμη Flash. Αυτές είναι εκατοντάδες MB μέχρι μερικά GB με ένα χρόνο πρόσβασης μερικών δεκάδων χιλιοστών του δευτερολέπτου (miliseconds).

Σημειώστε ότι η διαφορά απόδοσης μεταξύ της κύριας μνήμης και του χώρου μόνιμης αποθήκευσης, είναι πολύ μεγαλύτερη από τη διαφορά μεταξύ οποιονδήποτε άλλων παρακείμενων επιπέδων, ακόμα και όταν δεν υπάρχει καμία δευτερεύουσα κρυφή μνήμη στο σύστημα. Τα δεδομένα που φυλάσσονται στους καταχωρητές είναι υπό τον άμεσο έλεγχο του μεταγλωττιστή ή του συμβολομεταφραστή, αλλά τα περιεχόμενα των υπόλοιπων επιπέδων της ιεραρχίας διαχειρίζονται συνήθως αυτόματα. Οι κρυφές μνήμες είναι αποτελεσματικά αόρατες στο πρόγραμμα εφαρμογής, με τμήματα ή σελίδες εντολών και δεδομένα που μεταναστεύουν πάνω-κάτω στην ιεραρχία, υπό τον έλεγχο του υλικού. Μερικά συστήματα έχουν διάφορες μονάδες μεταφοράς δεδομένων από το δίσκο στη μνήμη RAM και ονομάζονται σελίδες. Η σελιδοποίηση μεταξύ της κύριας μνήμης και του δίσκου ελέγχεται από το λειτουργικό σύστημα, και παραμένει διαφανής στο πρόγραμμα εφαρμογής. Καθώς η διαφορά απόδοσης μεταξύ της κύριας μνήμης και δίσκου είναι τόσο μεγάλη, απαιτούνται ακόμη πιο περίπλοκοι αλγόριθμοι για να καθορίσουν πότε πρέπει να μεταναστεύσουν δεδομένα μεταξύ των επιπέδων.

Ένα ενσωματωμένο σύστημα συνήθως δεν θα έχει ένα μεγάλο δίσκο και για αυτό σπάνια χρησιμοποιείται η σελιδοποίηση. Ωστόσο, πολλά ενσωματωμένα συστήματα ενσωματώνουν κρυφές μνήμες, και τα τσιπ των επεξεργαστών διαθέτουν μια ποικιλία οργανώσεων κρυφής μνήμης.

Η γρήγορη μνήμη είναι ακριβότερη ανά bit από την αργή μνήμη, γι' αυτό η ιεραρχία μνήμης στοχεύει επίσης να δώσει μια απόδοση, κοντά στη γρηγορότερη μνήμη με ένα μέσο κόστος ανά bit που πλησιάζει αυτό της πιο αργής μνήμης.

3.2.13 Μνήμη on-chip

Κάποια μορφή μνήμης on-chip (εντός ολοκληρωμένου κυκλώματος ή επιπλήνθια μνήμη) είναι απαραίτητη, εάν ένας μικροεπεξεργαστής απαιτεί μια εξωτερική μνήμη. Με τις σημερινές ταχύτητες ρολογιού, μόνο η on-chip μνήμη μπορεί να υποστηρίξει ταχύτητες πρόσβασης σε κατάσταση μηδενικής αναμονής (χωρίς να περιμένει ο επεξεργαστής), και θα δώσει επίσης την καλύτερη αποδοτικότητα ενέργειας και μειωμένη ηλεκτρομαγνητική παρέμβαση από την off-chip μνήμη.

On-chip RAM Σε πολλά ενσωματωμένα συστήματα προτιμάται η απλή on-chip RAM (ονομάζεται και scratch pad), έναντι της κρυφής μνήμης για διάφορα πλεονεκτήματα:

- Είναι απλούστερη, φτηνότερη, και χρησιμοποιεί λιγότερη ενέργεια. Θα δούμε στα ακόλουθα τμήματα ότι η κρυφή μνήμη μεταφέρει ένα σημαντικό πλεονέκτημα στα πλαίσια της λογικής ότι είναι απαραίτητη, προκειμένου να επιτρέψει αποδοτική λειτουργία.

- Υφίσταται επίσης ένα σημαντικό κόστος σχεδίασης, εάν μια κατάλληλη off-the-shelf κρυφή μνήμη δεν είναι διαθέσιμη. Ο σχεδιασμός μιας κρυφής μνήμης είναι αρκετά σύνθετος.

- Έχει περισσότερη ντετερμινιστική (αιτιοκρατική) συμπεριφορά.

- Οι κρυφές μνήμες έχουν σύνθετες συμπεριφορές που μπορούν να κάνουν δύσκολη την πρόβλεψη του πόσο καλά θα λειτουργήσουν κάτω από συγκεκριμένες περιστάσεις. Ειδικότερα, μπορεί να είναι δύσκολο να εγγυηθεί τη διακοπή του χρόνου απόκρισης (interrupt response time), δηλαδή το χρόνο που απαιτείται για να εξυπηρετηθεί μια διακοπή του επεξεργαστή.

Το μειονέκτημα με την on-chip RAM έναντι της κρυφής μνήμης είναι ότι απαιτεί ρητή διαχείριση από τον προγραμματιστή, ενώ η κρυφή μνήμη είναι συνήθως διαφανής στον προγραμματιστή. Όπου το μείγμα προγράμματος είναι καθορισμένο με σαφήνεια και υπό τον έλεγχο του προγραμματιστή, η on-chip RAM μπορεί αποτελεσματικά να χρησιμοποιηθεί ως μία κρυφή μνήμη ελεγχόμενη από το λογισμικό. Όπου το μείγμα εφαρμογής δεν μπορεί να προβλεφθεί αυτή η διεργασία ελέγχου γίνεται πολύ δύσκολη.

Ως εκ τούτου μια κρυφή μνήμη προτιμάται συνήθως σε οποιοδήποτε γενικού σκοπού σύστημα, όπου η εφαρμογή του μείγματος είναι άγνωστη. Στα ΕΣ όμως, που προσδιορίζονται για ειδικού σκοπού εφαρμογές, μπορεί να χρησιμοποιηθεί αυτού του είδους η μνήμη αντί για cache.

Ένα σημαντικό πλεονέκτημα της on-chip RAM είναι ότι επιτρέπει στον προγραμματιστή να διαθέσει χώρο αυτήν χρησιμοποιώντας γνώσεις μελλοντικών αναγκών, δηλαδή να μεταφέρει σε αυτή από πριν δεδομένα που θα χρησιμοποιηθούν στο μέλλον [21]. Μια κρυφή μνήμη έχει γνώση μόνο της συμπεριφοράς στο παρελθόν του προγράμματος, και δεν μπορεί επομένως ποτέ να προετοιμαστεί εκ των προτέρων για κρίσιμες μελλοντικές διεργασίες. Αυτή είναι μια διαφορά, η οποία είναι πιο πιθανό να είναι σημαντική όταν κρίσιμες διεργασίες πρέπει να συναντήσουν αυστηρούς περιορισμούς πραγματικού χρόνου.

Ο σχεδιαστής του συστήματος πρέπει να αποφασίσει ποια είναι η σωστή προσέγγιση για ένα συγκεκριμένο σύστημα, λαμβάνοντας όλους αυτούς τους παράγοντες υπόψη. Οποιαδήποτε μορφή on-chip μνήμης επιλεχθεί, πρέπει να διευκρινιστεί με μεγάλη προσοχή. Πρέπει να είναι αρκετά γρήγορη ώστε να κρατήσει τον επεξεργαστή απασχολημένο και αρκετά μεγάλη, ώστε να περιέχει κρίσιμες ρουτίνες, αλλά ούτε πολύ γρήγορη (γιατί θα καταναλώσει πάρα πολλή ενέργεια), ούτε πολύ μεγάλη (γιατί θα καταλάβει πολύ μεγάλη περιοχή του τσιπ).

Κρυφές Μνήμες

Οι πρώτοι επεξεργαστές RISC εισήχθησαν σε μία εποχή που τυποποιημένα τσιπ μνήμης ήταν γρηγορότερα από τους σύγχρονούς μικροεπεξεργαστές, αλλά αυτή η κατάσταση δεν παρέμεινε για πολύ. Επόμενες εξελίξεις στην τεχνολογία διαδικασίας ημιαγωγών, κατέστησαν μικροεπεξεργαστές γρηγορότερους, έχουν εφαρμοστεί για να βελτιώσουν τα τσιπ μνήμης. Τα τυποποιημένα μέρη DRAM έχουν γίνει λίγο γρηγορότερα, αλλά κυρίως έχουν αναπτυχθεί με στόχο να προσφέρουν μια πολύ υψηλότερη χωρητικότητα.

Ταχύτητες επεξεργαστών και μνήμης

Το 1980 ένα χαρακτηριστικό τσιπ DRAM θα μπορούσε να κρατήσει 4 Kbits δεδομένων, με τα 16 Kbit τσιπ να φτάνουν το 1981 και το 1982, αντίστοιχα. Αυτά τα μέρη θα ανακύκλωναν στα 3 ή 4 MHz για τις τυχαίες προσβάσεις, και για δύο φορές αυτό το ποσοστό για τις τοπικές προσβάσεις (στην τροποποίηση σελίδων). Οι μικροεπεξεργαστές εκείνο το καιρό μπορούσαν να ζητήσουν περίπου δύο εκατομμύρια προσβάσεις μνήμης ανά δευτερόλεπτο.

Το 2000 τα τσιπ DRAM είχαν χωρητικότητα 256 Mbits ανά τσιπ, με τις τυχαίες προσβάσεις να λειτουργούν περίπου στα 30 MHz. Οι μικροεπεξεργαστές μπορούσαν να ζητήσουν αρκετές εκατοντάδες εκατομμυρίων προσβάσεις μνήμης ανά δευτερόλεπτο. Εάν ο επεξεργαστής ήταν τόσο πολύ γρηγορότερος από τη μνήμη, μπορούσε να αποδώσει τα μέγιστα, μόνο με τη βοήθεια μιας κρυφής μνήμης. Μια κρυφή μνήμη είναι μια μικρή, πολύ γρήγορη μνήμη, που διατηρεί αντίγραφα των πρόσφατα χρησιμοποιημένων τιμών μνήμης. Λειτουργεί διαφανώς στον προγραμματιστή, αποφασίζοντας αυτόματα ποιες τιμές θα κρατήσει και ποιες θα παραγράψει. Στην εποχή μας συνήθως εφαρμόζεται πάνω στο ίδιο τσιπ με τον επεξεργαστή (στο παρελθόν ήταν σε διαφορετικά τσιπ). Οι κρυ-

φές μνήμες λειτουργούν επειδή τα προγράμματα επιδεικνύουν κανονικά την ιδιοκτησία της τοπικότητας, το οποίο σημαίνει ότι σε οποιοδήποτε χρόνο τείνουν να εκτελούν τις ίδιες εντολές πολλές φορές (παραδείγματος χάριν σε έναν βρόχο) στις ίδιες περιοχές δεδομένων (για περίπτωση μια στοιβάδα).

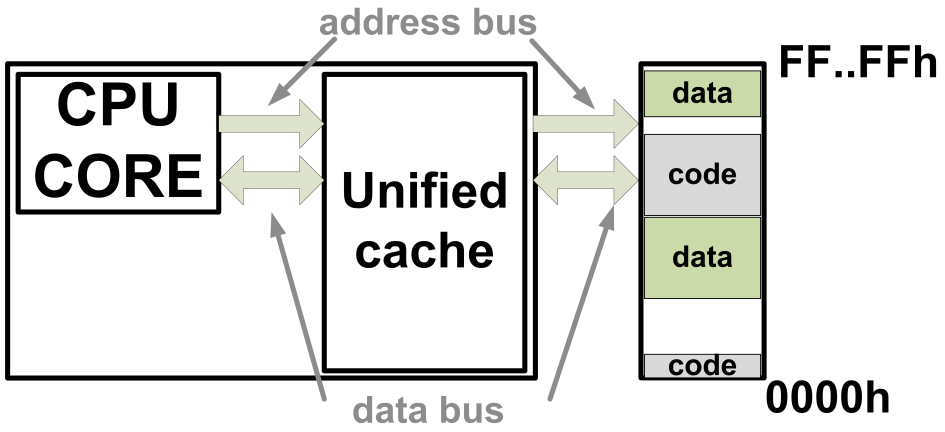
Ενοποιημένες και κρυφές μνήμες του Χάρβαρντ

Οι κρυφές μνήμες μπορούν να κατασκευαστούν με πολλούς τρόπους. Στο πιο υψηλό επίπεδο ένας επεξεργαστής μπορεί να έχει μία από τις ακόλουθες δύο διατάξεις:

- Μια ενοποιημένη (unified) κρυφή μνήμη. Αυτή είναι μια ενιαία κρυφή μνήμη τόσο για τις εντολές όσο και για τα δεδομένα, όπως διευκρινίζεται στο σχήμα 3.14.

- Ξεχωριστές κρυφές μνήμες εντολών και δεδομένων (split). Αυτή η διάταξη καλείται μερικές φορές ως τροποποιημένη αρχιτεκτονική του Χάρβαρντ όπως παρουσιάζεται παρακάτω.

Και οι δύο αυτές οι διατάξεις είναι χρήσιμες. Η ενοποιημένη κρυφή μνήμη αυτόματα ρυθμίζει το ποσοστό της κρυφής μνήμης που χρησιμοποιείται από τις εντολές σύμφωνα με τις τρέχουσες απαιτήσεις του προγράμματος, δίνοντας καλύτερη απόδοση από ότι ένας προκαθορισμένος διαχωρισμός. Από την άλλη μεριά, οι χωριστές κρυφές μνήμες επιτρέπουν τις εντολές φόρτωσης και αποθήκευσης να εκτελεστούν μέσα σε έναν ενιαίο κύκλο ρολογιού.



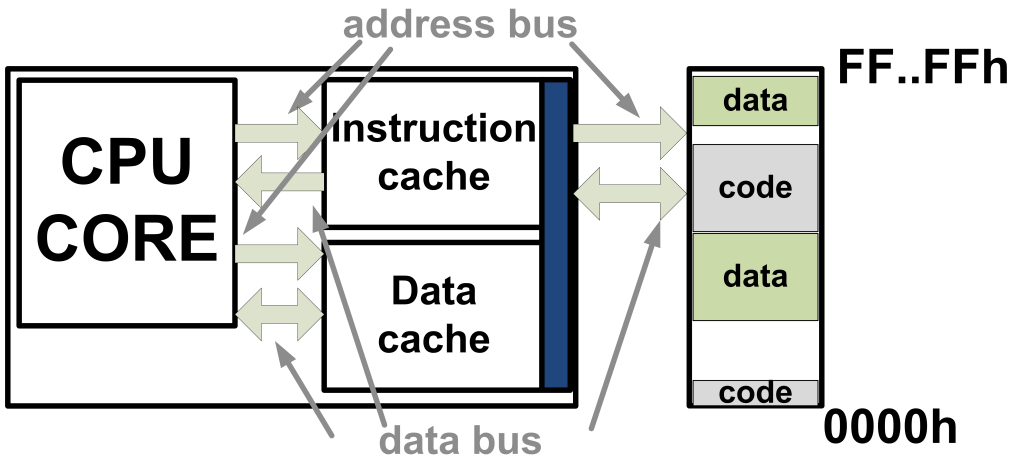
Σχήμα 3.14: Μία ενοποιημένη κρυφή μνήμη εντολών και δεδομένων.

Μετρικές απόδοσης κρυφής μνήμης

Δεδομένου ότι ο επεξεργαστής μπορεί να λειτουργήσει στην υψηλή συχνότητα ρολογιού του μόνον όταν τα αντικείμενα μνήμης που απαιτεί διατηρούνται

στην κρυφή μνήμη, η συνολική απόδοση συστήματος εξαρτάται έντονα από το ποσοστό των προσβάσεων μνήμης το οποίο δεν μπορεί να ικανοποιηθεί από την κρυφή μνήμη. Η πρόσβαση σε ένα αντικείμενο το οποίο βρίσκεται στην κρυφή μνήμη καλείται επιτυχία, και η πρόσβαση σε ένα αντικείμενο που δεν βρίσκεται στην κρυφή μνήμη καλείται αποτυχία. Το ποσοστό όλων των προσβάσεων μνήμης που ικανοποιούνται από την κρυφή μνήμη είναι το ποσοστό επιτυχιών, εκφρασμένο συνήθως ως ποσοστό, και το ποσοστό αυτών που δεν ικανοποιούνται είναι το ποσοστό αποτυχιών.

Το ποσοστό αποτυχιών, μιας καλά σχεδιασμένης κρυφής μνήμης, πρέπει να είναι μόνο μερικά τοις εκατό, εάν ένας σύγχρονος επεξεργαστής πρόκειται να εκπληρώσει τη δυνατότητά του. Το ποσοστό αποτυχίας εξαρτάται από έναν αριθμό παραμέτρων της κρυφής μνήμης, συμπεριλαμβανομένου του μεγέθους της (ο αριθμός των bytes της μνήμης μέσα στην κρυφή μνήμη) και της οργάνωσης της.



Σχήμα 3.15: Ξεχωριστά δεδομένα και εντολές κρυφής μνήμης.

3.2.14 Διαχείριση μνήμης

Όπως έγινε κατανοητό, ένας επεξεργαστής διαχειρίζεται τον κώδικα προγράμματος μέσα σε ένα ενσωματωμένο σύστημα μέσω των διεργασιών (αν υπάρχει λειτουργικό σύστημα, ΛΣ). Ο πυρήνας πρέπει επίσης να έχει κάποιο σύστημα φόρτωσης και εκτέλεσης διεργασιών μέσα στο σύστημα, εφόσον η CPU εκτελεί μόνο τον κώδικα της διεργασίας που είναι μέσα στην κρυφή μνήμη ή στη RAM. Με τις πολλαπλές διεργασίες να μοιράζονται τον ίδιο χώρο, ένα ΛΣ χρειάζεται ένα σύστημα με μηχανισμό προστασίας για να προστατέψει τον κώδικα της διεργασίας από άλλες ανεξάρτητες διεργασίες. Επιπλέον, εφόσον ένα

ΛΣ πρέπει να τοποθετηθεί στην ίδια μνήμη με τις διεργασίες τις οποίες διαχειρίζεται, ο μηχανισμός προστασίας πρέπει να περιλαμβάνει τη διαχείριση του δικού του κώδικα στη μνήμη, και να το προστατέψει από τον κώδικα των διεργασιών που διαχειρίζεται. Αυτές οι λειτουργίες, και άλλες πολλές, είναι καθήκον του Διαχειριστή Μνήμης ενός ΛΣ. Δεν έχουν όλα τα ΕΣ ΛΣ, γιατί η ύπαρξη ενός ΛΣ επιβαρύνει το σύστημα με την εκτέλεση κώδικα που απαιτείται για το ίδιο το ΛΣ. Όμως, υπάρχουν περιπτώσεις που πρέπει να χρησιμοποιηθεί ένα ΛΣ σε ένα ΕΣ, όπως θα συζητηθεί στο σχετικό κεφάλαιο με τα ΛΣ.

Γενικά, τα καθήκοντα του διαχειριστή μνήμης πυρήνα σε ένα ΛΣ περιλαμβάνουν:

- Διαχείριση της χαρτογράφησης μεταξύ της λογικής μνήμης που βλέπουν οι διεργασίες και της φυσικής μνήμης RAM, αφού κάθε διεργασία εκτελείται σε ένα δικό της απομονωμένο χώρο διευθύνσεων (που ονομάζεται λογικός χώρος διευθύνσεων).

- Καθορισμό των διαδικασιών που θα φορτωθούν στο διαθέσιμο χώρο μνήμης, όπως και σε ποιες διευθύνσεις θα φορτωθούν.

- Προσδιορισμός και ελευθέρωση της μνήμης για διεργασίες που αποτελούν το σύστημα.

- Υποστήριξη προσδιορισμού και ελευθέρωσης μνήμης των προγραμμάτων (μέσα στη διεργασία), όπως η γλώσσα C με λειτουργίες ‘alloc’ και ‘dealloc’, ή ειδικές ρουτίνες προσδιορισμού και ελευθέρωσης προσωρινής μνήμης.

- Παρακολούθηση της χρήσης μνήμης σε ένα τμήμα συστήματος, ώστε να μπορεί να το απελευθερώνει σε περίπτωση που μια διαδικασία τερματίσει.

- Διαβεβαίωση για τη συνεκτικότητα της κρυφής μνήμης (για συστήματα πολλών επεξεργαστών με κρυφή μνήμη), ώστε όλοι οι επεξεργαστές να βλέπουν την ίδια εικόνα για μια διεύθυνση της εξωτερικής μνήμης.

- Διαβεβαίωση προστασίας της μνήμης διεργασιών, ώστε η μια διεργασία να μη μπορεί να διαβάσει ή να καταστρέψει τα δεδομένα ή τις εντολές άλλης διεργασίας.

Η φυσική μνήμη αποτελείται από δισδιάστατες σειρές φτιαγμένες από κελιά που διευθυνσιοδοτούνται από μια μοναδική γραμμή και στήλη, στην οποία κάθε κελί μπορεί να αποθηκεύσει 1 bit. Το ΛΣ συμπεριφέρεται στη μνήμη ως μια μεγάλη μονοδιάστατη σειρά, καλούμενη χάρτης μνήμης (memory map). Υπάρχει ο λογικός (ή ιδεατός) και ο φυσικός χώρος διευθύνσεων μνήμης. Οι διεργασίες

βλέπουν τον ιδεατό χώρο διευθύνσεων ενώ οι φυσικές διευθύνσεις τοποθετούνται στο δίαυλο διευθύνσεων της μνήμης. Η αντιστοίχηση γίνεται είτε από το ίδιο το ΛΣ, είτε από τον επεξεργαστή, αν διαθέτει μια μονάδα διαχείρισης μνήμης (memory management unit).

Ο τρόπος με τον οποίο τα ΛΣ διαχειρίζονται το χώρο της λογικής μνήμης διαφέρει ανάμεσα στα ΛΣ. Γενικά, οι πυρήνες των ΛΣ εκτελούν κώδικα πυρήνα (kernel) σε ξεχωριστό χώρο μνήμης από τις διεργασίες που εκτελούν κώδικα υψηλότερου επιπέδου (δηλαδή κώδικα επιπέδου εφαρμογής), και ονομάζεται κώδικας χρήστη (user). Η διαφοροποίηση έγκειται στο γεγονός ότι ο κώδικας που εκτελείται σε επίπεδο πυρήνα δεν έχει κανένα περιορισμό και μπορεί να προσπελάσει οποιοδήποτε περιφερειακό ή διεύθυνση μνήμης. Αντιθέτως, ο κώδικας που εκτελείται σε επίπεδο χρήστη, έχει πολλούς περιορισμούς που υπάρχουν για την καλύτερη ασφάλεια και σταθερότητα του συστήματος. Αν ο κώδικας που εκτελείται σε επίπεδο χρήστη θελήσει κάποια ενέργεια που απαιτεί αυξημένα δικαιώματα, θα ζητήσει από τον πυρήνα να μεσολαβήσει και να δράσει εκ μέρους αυτού, επιστρέφοντας σε αυτόν τα αποτελέσματα. Κάθε ένας από αυτούς τους χώρους μνήμης ((i) πυρήνα, συμπεριλαμβανομένου και του κώδικα πυρήνα του ΛΣ και (ii) χρήστη, συμπεριλαμβανομένων των διεργασιών υψηλού επιπέδου) υπόκεινται σε διαχείριση με διαφορετικό τρόπο. Στην πραγματικότητα, οι περισσότερες διεργασίες τυπικά εκτελούνται με μια από τις δύο μεθόδους: πυρήνα και χρήστη, ανάλογα με τις ρουτίνες που εκτελούνται. Οι ρουτίνες πυρήνα εκτελούνται σε μεθόδους πυρήνα (αναφερόμενες επίσης ως μέθοδοι-επόπτες -supervisors), σε διαφορετικό χώρο μνήμης και επίπεδο από ότι τα υψηλότερα επίπεδα λογισμικού, όπως οι εφαρμογές. Τυπικά, αυτά τα υψηλότερα επίπεδα λογισμικού εκτελούνται σε μεθόδους χρήστη και μπορούν μόνο να εισχωρήσουν σε οτιδήποτε εκτελείται σε μέθοδο πυρήνα μέσω κλήσεων συστήματος (operating system calls). Ο πυρήνας διαχειρίζεται τη μνήμη και για τον εαυτό του και για τις διεργασίες χρήστη.

3.2.15 Χώρος μνήμης χρηστών

Επειδή οι πολλαπλές διεργασίες μοιράζονται την ίδια φυσική μνήμη όταν φορτώνονται στη RAM για επεξεργασία, πρέπει επίσης να υπάρχει κάποιος μηχανισμός προστασίας, έτσι ώστε οι διεργασίες να μην μπορούν ακούσια ή εκούσια να επηρεάσουν η μια την άλλη όταν εναλλάσσονται μέσα και έξω από έναν χώρο μιας φυσικής μνήμης. Αυτά τα ζητήματα τυπικά λύνονται από το λειτουργικό σύστημα μέσω της 'ανταλλαγής' μνήμης, όπου τα διαμερίσματα της μνήμης ανταλλάσσονται μέσα και έξω από τη μνήμη κατά το χρόνο εκτέλεσης. Τα πιο κοινά διαμερίσματα μνήμης που χρησιμοποιούνται για ανταλλαγή είναι τα τμήματα (τεμαχισμός των διεργασιών) και οι σελίδες (τεμαχισμός της λογικής μνήμης). Η κατάτμηση και η σελιδοποίηση όχι μόνο απλοποιούν την

ανταλλαγή, αλλά επιτρέπουν την επαναχρησιμοποίηση κώδικα και την προστασία μνήμης, καθώς επίσης αποτελούν τη βάση για την εικονική μνήμη. Η εικονική μνήμη είναι ένας μηχανισμός που ελέγχεται από το ΛΣ για να επιτρέψει τον περιορισμένο χώρο μνήμης μίας συσκευής να μοιραστεί από τις πολλαπλές ανταγωνιστικές διεργασίες χρήστη.

3.2.16 Κατάτμηση

Μια διεργασία ενθυλακώνει όλες τις πληροφορίες που σχετίζονται με την εκτέλεση ενός προγράμματος, συμπεριλαμβανομένων του πηγαίου κώδικα, της στοίβας, των δεδομένων, κτλ. Όλοι οι διαφορετικοί τύποι πληροφοριών σε μια διεργασία χωρίζονται σε λογικές μονάδες μνήμης μεταβλητού μεγέθους, αποκαλούμενα τμήματα. Ένα τμήμα είναι ένα σύνολο λογικών διευθύνσεων που περιέχουν τον ίδιο τύπο πληροφοριών. Οι διευθύνσεις τμημάτων είναι λογικές διευθύνσεις που αρχίζουν από το 0, και αποτελούνται από έναν αριθμό τμήματος, που υποδεικνύει τη διεύθυνση βάσεων του τμήματος, και ένα τμήμα έναρξης που καθορίζει την πραγματική φυσική διεύθυνση μνήμης. Τα τμήματα προστατεύονται ανεξάρτητα, που σημαίνει ότι έχουν προσδιορίσει τα χαρακτηριστικά της προσβασιμότητας, όπως η κοινόχρηστη ιδιότητα (shared), όπου οι άλλες διεργασίες μπορούν να έχουν πρόσβαση σε αυτό το τμήμα, Μόνο για Ανάγνωση ή Ανάγνωση και Εγγραφή.

Τα περισσότερα ΛΣ τυπικά επιτρέπουν στις διεργασίες να χρησιμοποιούν πλήρη ή μερικό συνδυασμό των πέντε τύπων των πληροφοριών μέσα στα τμήματα: τμήμα κειμένου ή κώδικα (code / text segment), τμήμα δεδομένων (data segment), τμήμα συμβόλων bss (block starting symbol segment), και το τμήμα στοίβας (stack segment). Ένα τμήμα κειμένου είναι μια περιοχή μνήμης που περιλαμβάνει τον πηγαίο κώδικα. Ένα τμήμα δεδομένων είναι μια περιοχή μνήμης που περιλαμβάνει τις αρχικοποιημένες μεταβλητές του πηγαίου κώδικα (δεδομένα). Ένα τμήμα bbs είναι μια περιοχή μνήμης προσδιορισμένη με στατικό τρόπο, που περιλαμβάνει τις μεταβλητές του πηγαίου κώδικα που δεν έχουν αρχικοποιηθεί (δεδομένα). Τα τμήματα δεδομένων, κειμένου και bbs καθορίζονται κατά το χρόνο μετάφρασης, και είναι υπό αυτή τη μορφή στατικά τμήματα. Αυτά τα τρία τμήματα είναι αυτά που τυπικά αποτελούν τμήμα ενός εκτελέσιμου αρχείου. Τα εκτελέσιμα αρχεία μπορεί να διαφέρουν ως προς τα τμήματα από τα οποία συντίθενται, αλλά γενικά περιλαμβάνουν μια επιγραφή, και διαφορετικά κομμάτια τα οποία αναπαριστούν τους τύπους των τμημάτων, συμπεριλαμβανομένου του ονόματος, τις άδειες κτλ, όπου ένα τμήμα μπορεί να φτιαχτεί από ένα ή περισσότερα κομμάτια. Το ΛΣ δημιουργεί την εικόνα μιας διεργασίας μέσω της χαρτογράφησης μνήμης των περιεχομένων του εκτελέσιμου αρχείου, που σημαίνει τη φόρτωση και διερμηνεία των τμημάτων (περιοχών) που αντικατοπτρίζουν στην εκτελέσιμη μνήμη. Υπάρχουν διάφορα σχή-

ματα εκτελέσιμων αρχείων που υποστηρίζονται από τα ενσωματωμένα ΛΣ, τα πιο κοινά συμπεριλαμβάνουν:

- ELF (Executable and Linkable Format - εκτελέσιμο και διασυνδεδεμένο αρχείο): βασισμένο στο UNIX, περιλαμβάνει έναν συνδυασμό από κατάλληλη επικεφαλίδα (ELF), τον πίνακα επικεφαλίδας προγράμματος, τον πίνακα επικεφαλίδας τμήματος, την περιοχή ELF και το τμήμα ELF. Τα Linux, FreeBSD και τα vxWorks(WRS) είναι μερικά παραδείγματα των ΛΣ που υποστηρίζουν ELF.

- Κλάση Java Byte: Ένα αρχείο κλάσης περιγράφει με λεπτομέρειες μια κλάση της Java υπό μορφή πλήθους bytes των 8 bit. Αντί για τμήματα, τα στοιχεία του αρχείου κλάσης καλούνται αντικείμενα. Η μορφοποίηση αρχείου κλάσης Java περιλαμβάνει την περιγραφή της κλάσης, όπως επίσης και το πώς αυτή η κλάση συνδέεται με τις υπόλοιπες κλάσεις. Τα κύρια περιεχόμενα ενός αρχείου κλάσης είναι ένας πίνακας συμβόλων (με σταθερές), δήλωση των πεδίων, κλήσεις μεθόδων (κώδικας) και συμβολικές αναφορές (όπου υπάρχουν άλλες αναφορές κλάσεων). Το ΛΣ Jbed RT είναι ένα παράδειγμα που υποστηρίζει τη μορφοποίηση κώδικα Java Byte.

- COFF (Common Object File Format - κοινό σχήμα αρχείων αντικειμένου): Μια μορφοποίηση αρχείου κλάσης το οποίο (μεταξύ άλλων) καθορίζει ένα αρχείο εικόνας το οποίο περιλαμβάνει τις επικεφαλίδες αρχείων που περιέχουν μια υπογραφή αρχείου, επικεφαλίδα COFF, μια προαιρετική επικεφαλίδα, και επίσης αρχεία αντικειμένου που περιέχουν μόνο την επικεφαλίδα COFF. Το ΛΣ WinCE της Microsoft είναι ένα παράδειγμα ενσωματωμένου OS που υποστηρίζει τη μορφοποίηση αρχείου εκτέλεσης COFF.

Ένα τμήμα στοίβας είναι ένα τμήμα της μνήμης που είναι δομημένο ως ουρά τελευταίο μέσα, πρώτα έξω (Last In First Out - LIFO), όπου το στοιχείο είναι 'ωθημένο' επάνω στη στοίβα, ή 'απωθημένο' από την στοίβα (η ώθηση και η απώθηση είναι οι μόνες δύο διαδικασίες συνδεδεμένες με την στοίβα).

Η στοίβα χρησιμοποιείται ως μια απλή και αποδοτική μέθοδος στα πλαίσια ενός προγράμματος για τη διάθεση και την απελευθέρωση της μνήμης για δεδομένα που είναι προβλέψιμα (δηλαδή τοπικές μεταβλητές, διάβαση παραμέτρων, κ.λπ.). Σε μία στοίβα, ο χρησιμοποιούμενος και ελεύθερος χώρος μνήμης βρίσκεται κατά συνέπεια μέσα στο χώρο μνήμης. Εντούτοις, δεδομένου ότι 'η ώθηση' και 'η απώθηση' είναι οι μόνες δύο διαδικασίες που συνδέονται με μία στοίβα, μία στοίβα μπορεί να έχει περιορισμένες χρήσεις.

3.3 Ο επεξεργαστής σε ένα ΕΣ

Η καρδιά σε κάθε υπολογιστικό σύστημα είναι το δομοστοιχείο που κάνει τους υπολογισμούς. Στους υπολογιστές γενικού σκοπού δεν υπάρχει ευελιξία ως προς αυτό: υπάρχουν ένας ή περισσότεροι επεξεργαστές γενικής αρχιτεκτονικής 64 bit συμβατής με Intel x86-64, που σημαίνει ότι μπορούν να αντεπεξέλθουν σε οποιαδήποτε φόρτο εργασίας τους ζητηθεί. Αντιθέτως, στα ΕΣ υπάρχει μια μεγάλη ποικιλία υλοποιήσεων, κάποιες από τις οποίες είναι αντιδιαμέτρου αντίθετες μεταξύ τους. Για παράδειγμα τα ΕΣ μπορούν να χρησιμοποιήσουν έναν γενικής χρήσης επεξεργαστή (π.χ. Intel ATOM), ειδικούς επεξεργαστές ψηφιακής επεξεργασίας σημάτων (π.χ. Texas Instrument TIC6201), μικροεπεξεργαστές 8 bit (π.χ. AVR ATMEGA328P), σύνθετους επεξεργαστές 32 bit (π.χ. ARM Cortex A15), απλούς επεξεργαστές 32 bit (MIPS R3000), επεξεργαστές μαλακού πυρήνα σε επαναδιαμορφώσιμη λογική FPGA (soft-cores, π.χ. Altera NIOS II) ή ακόμη και εξειδικευμένα κυκλώματα υλικού (ASIC), τα οποία δεν έχουν καν επεξεργαστή, αλλά υλοποιούν μια συγκεκριμένη λειτουργία χρησιμοποιώντας μια μηχανή πεπερασμένων καταστάσεων (finite state machine). Μάλιστα, οι επεξεργαστές δεν είναι συμβατοί μεταξύ τους, όπως στους υπολογιστές γενικού σκοπού ή στους φορητούς υπολογιστές ή στους διακομιστές (που όλοι είναι συμβατοί με τις προδιαγραφές της αρχιτεκτονικής x86-64, και έτσι ένα πρόγραμμα μπορεί να εκτελεστεί σε όλους τους συμβατούς επεξεργαστές). Το οικοσύστημα των ΕΣ φαίνεται τόσο χαοτικό σε κάποιον που μόλις τώρα το γνωρίζει, και ο σχεδιασμός ενός ΕΣ μπορεί να παρομοιαστεί με ένα δαιδαλώδες λαβύρινθο με πολλά μονοπάτια, που κάθε μονοπάτι οδηγεί σε άλλες επιλογές. Στο επόμενο κεφάλαιο, αναλύουμε τους προγραμματιζόμενους επεξεργαστές· τα επεξεργαστικά στοιχεία σε FPGA και σε ASIC αναλύονται σε άλλο κεφάλαιο.

3.4 Η αρχιτεκτονική αποθηκευμένου προγράμματος

Σε αυτή την ενότητα θα περιγράψουμε την ισχύουσα αρχιτεκτονική στα προγραμματιζόμενα υπολογιστικά συστήματα, που φέρει κοινά στοιχεία, είτε αφορά τα ΕΣ, είτε άλλα συστήματα. Η αρχιτεκτονική αυτή ονομάζεται αποθηκευμένου προγράμματος (stored-program), γιατί το πρόγραμμα βρίσκεται αποθηκευμένο σε μια μορφή μόνιμης αποθήκευσης (π.χ. σε ένα δίσκο ή σε έναν οδηγό Flash), μεταφέρεται στη μνήμη RAM, και στη συνέχεια η κάθε εντολή του προγράμματος εισέρχεται μέσα στον επεξεργαστή, ο οποίος την εκτελεί. Μόλις την εκτελέσει, τότε φέρνει την επόμενη εντολή κ.ο.κ.

Από τα παραπάνω καταλαβαίνουμε ότι ο επεξεργαστής εκτελεί συνεχώς

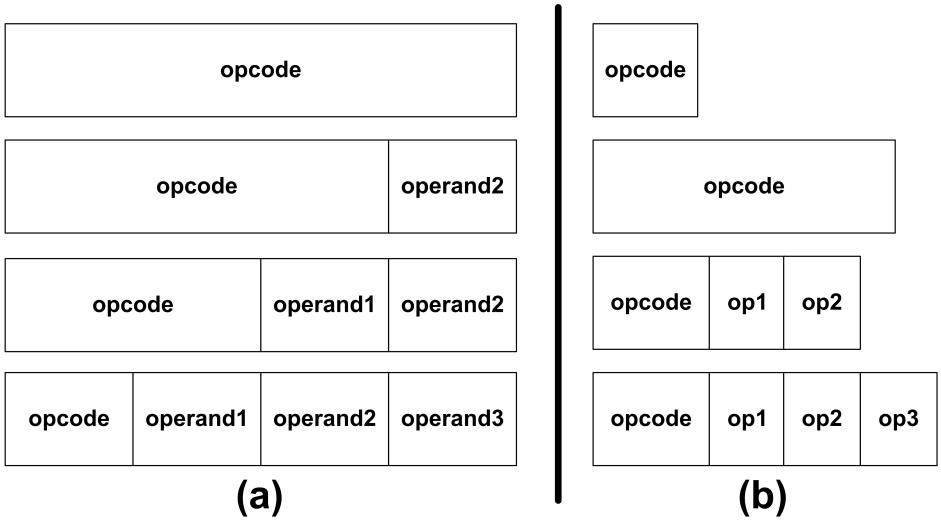
έναν αέναο βρόχο, που αποτελείται από τα βήματα της προσκόμισης της εντολής και της εκτέλεσης. Με μια καλύτερη ανάλυση βρίσκουμε ότι ο επεξεργαστής εκτελεί συνεχώς τέσσερις λειτουργίες:

1. Προσκόμιση εντολής (fetch)
2. Αποκωδικοποίηση εντολής (decode)
3. Εκτέλεση εντολής (execute)
4. Αποθήκευση των αποτελεσμάτων (store)

Έτσι, ο επεξεργαστής προσκομίζει από την εξωτερική μνήμη, σε έναν εσωτερικό καταχωρητή που ονομάζεται καταχωρητής εντολής (Instruction Register) την εντολή που θα ξεκινήσει την εκτέλεση. Η εντολή αυτή είναι κωδικοποιημένη με '1' και '0' σε συγκεκριμένες θέσεις.

Οι επεξεργαστές χωρίζονται σε 2 κατηγορίες ως προς το μήκος των εντολών που υποστηρίζουν (Εικόνα 3.16): οι εντολές σταθερού μήκους bit, που σημαίνει ότι όλες οι εντολές θα έχουν πάντα το ίδιο μήκος (π.χ. 32 bit) και οι εντολές μεταβλητού μήκους bit, που σημαίνει ότι άλλες εντολές θα είναι 1 Byte, άλλες εντολές 2 Byte, κ.ο.κ. (σε κάποιους επεξεργαστές έφτανε μέχρι και 31 Byte) [22]. Οι εντολές με σταθερό μήκος bit είναι πολύ πιο εύκολο να αποκωδικοποιηθούν και άρα απαιτούν λιγότερο υλικό στον επεξεργαστή για αυτή τη λειτουργία. Δεν είναι παράξενο λοιπόν, που η πλειονότητα των επεξεργαστών που χρησιμοποιούνται στα ΕΣ, έχει υιοθετήσει εντολές σταθερού μήκους. Από την άλλη, οι εντολές μεταβλητού μήκους επιτρέπουν μεγαλύτερη ευελιξία και είναι πιο φιλικές για τον προγραμματισμό, αφού επιτρέπουν συνήθως και πολύπλοκες λειτουργίες με μια εντολή (π.χ. έλεγχος σε μια περιοχή μνήμης για την ύπαρξη ενός string). Αυτές οι πολύπλοκες λειτουργίες μπορούν να υλοποιηθούν με πολλές εντολές στους υπολογιστές σταθερού μήκους. Βέβαια, η ύπαρξη πολύπλοκων εντολών συνεπάγεται και την έντονη διακύμανση στον απαιτούμενο χρόνο εκτέλεσης κάθε εντολής: κάποιες εντολές θα απαιτήσουν 1 κύκλο, κάποιες 10 κύκλους, σε αντιδιαστολή με τις εντολές σταθερού μήκους που έχουν στη συντριπτική πλειονότητα χρόνο εκτέλεσης του 1 κύκλου ρολογιού. Οι ιδιότητες αυτές (σταθερό μήκος εντολής, χρόνος εκτέλεσης εντολής ίσο με 1 κύκλο ρολογιού, λίγες εντολές και όχι πολύπλοκες, απλοποιημένη λογική αποκωδικοποίησης) χαρακτηρίζουν τους υπολογιστές RISC (reduced instruction set computing, επεξεργαστές με μειωμένο σύνολο εντολών), όπως είναι οι επεξεργαστές ARM και MIPS. Στον αντίποδα βρίσκονται οι CISC (complex instruction set computing, επεξεργαστές πολύπλοκου σετ εντολών), που είναι κυρίως οι οικογένειες IA32 και x86-64. Εκτός από τις παραπάνω σημαντικές διαφορές, υπάρχουν και άλλες όπως το ότι οι εντολές RISC απαιτούν 1 κύκλο εκτέλεσης, θα

πρέπει όλοι οι παράμετροι να είναι σε καταχωρητές, σε αντίθεση με τους CISC που επιτρέπουν και εντολές με απευθείας πράξη πάνω σε δεδομένα θέσης μνήμης, και ότι τα προγράμματα RISC είναι μεγαλύτερα σε μέγεθος αφού απαιτούν πολλές απλές εντολές για να ολοκληρωθούν. Πριν από το 1970 όλοι οι υπολογιστές ήταν CISC, αλλά με την ανάπτυξη των ενσωματωμένων συστημάτων, όλοι οι νέοι επεξεργαστές είναι RISC και οι επεξεργαστές που ήταν CISC (οικογένεια επεξεργαστών Intel x86) αποκωδικοποιούνται σε μικρο-εντολές RISC. Επίσης, οι σημερινοί RISC επεξεργαστές έχουν ενισχυθεί με κάποιες παραπάνω εντολές (τύπου CISC) για τη διευκόλυνση κάποιων λειτουργιών, χωρίς να σημαίνει ότι δεν είναι RISC.



Σχήμα 3.16: Οι εντολές των επεξεργαστών έχουν ποικίλες μορφές. Στο σχήμα εμφανίζονται οι εντολές σταθερού μήκους bit (a) και οι εντολές μεταβλητού μήκους bit (β) με τις διάφορες μορφές, ως προς τις παραμέτρους

Εκτός από το μήκος των εντολών, ένα άλλο στοιχείο είναι ο αριθμός των παραμέτρων που δέχεται μια εντολή. Για παράδειγμα μια εντολή πρόσθεσης σε έναν επεξεργαστή είναι: *ADD AX,BX*, που σημαίνει να προστεθούν οι τιμές των καταχωρητών AX και BX και το αποτέλεσμα να αποθηκευτεί στον καταχωρητή AX. Σε έναν άλλο επεξεργαστή, η εντολή πρόσθεσης είναι διαφορετική: *ADDS R1,R2,R3*, που σημαίνει να προστεθούν οι τιμές των καταχωρητών R2 και R3 και το αποτέλεσμα να αποθηκευτεί στον καταχωρητή R1. Όπως φαίνεται ο κάθε επεξεργαστής έχει τη δικιά του σύνταξη και τα δικά του ονόματα στους καταχωρητές και τις εντολές. Υπάρχουν γενικά εντολές που δε παίρνουν καμία παράμετρο, ή παίρνουν 1 ή 2 ή 3 παραμέτρους. Ο κάθε επεξεργαστής υποστηρίζει το δικό του συντακτικό (η Εικόνα 3.16 δείχνει τη δομή των εντολών για 0 ή 1 ή 2 ή 3 παραμέτρους).

3.4.1 Κώδικας μηχανής

Όλα τα προγράμματα που εξετάζει ένας επεξεργαστής εκφράζονται ως ψηφία 1 και 0 στη μνήμη του συστήματος. Η μνήμη αποθηκεύει μόνο 1 και 0, οπότε θα πρέπει με κάποιον τρόπο να καθορίζεται τι σημαίνει μια λέξη, π.χ. 11100011. Αυτή η λέξη θα μπορούσε να είναι εντολή προς τον επεξεργαστή (κώδικας μηχανής) ή δεδομένα (ένας χαρακτήρας ASCII ή μια ακέραια τιμή χωρίς πρόσημο ή μια ακέραια τιμή με πρόσημο ή τμήμα μιας μεγαλύτερης λέξης). Το τι σημαίνει κάθε φορά, συσχετίζεται με το που βρίσκεται η συγκεκριμένη τιμή. Αν βρίσκεται σε ένα τμήμα μνήμης που έχει σημανθεί με κατάλληλο τρόπο ως κώδικας (code), τότε θα εκτελεστεί, ενώ αν είναι σε μια περιοχή μνήμης που έχει σημανθεί ως δεδομένο θα επεξεργαστεί κατάλληλα. Σε αυτή τη λειτουργία βασίζονται κάποιοι κακόβουλοι χρήστες που τοποθετούν δεδομένα ειδικής κωδικοποίησης στη μνήμη του συστήματος και στη συνέχεια καταφέρουν να κάνουν τον επεξεργαστή να τα εκτελέσει ως κώδικα και άρα να εκτελέσει αυτά που οι ίδιοι με προσοχή έχουν τοποθετήσει.

Ο κάθε επεξεργαστής καταλαβαίνει μόνο το δικό του κώδικα μηχανής. Κάθε κώδικας μηχανής διαιρείται σε πεδία όπως opcode (κώδικας πράξης) και operand(s) (παράμετροι). Οι μηχανικοί βρίσκουν τις λίστες με κώδικες μηχανής πολύ πολύπλοκες για να τα γράψουν και ακόμα ποιο πολύπλοκες να τις καταλάβουν. Για να αντιμετωπίσουν αυτό το μεγάλο πρόβλημα, χρησιμοποιείται μια σημειολογική γλώσσα που να γεφυρώνει το χάσμα ανάμεσα σε υπολογιστή και άνθρωπο την αποκαλούμενη Συμβολική γλώσσα (Assembly). Προκειμένου να προγραμματιστεί ένας οποιοδήποτε επεξεργαστής, μπορεί να χρησιμοποιηθεί μια γλώσσα υψηλού επιπέδου, όπως η C, που θα μεταγλωττιστεί από τα κατάλληλα εργαλεία, δημιουργώντας τη συμβολική γλώσσα (assembly) που θα μεταφραστεί τον κώδικα μηχανής, και στη συνέχεια την ακολουθία των bit 0 και 1 που καταλαβαίνει ο προγραμματιστής. Εναλλακτικά, κάποιος επεξεργαστής μπορεί να προγραμματιστεί κατευθείαν με τη συμβολική γλώσσα. Αν και η μνήμη του συστήματος αποθηκεύει τιμές στο δυαδικό σύστημα, συνήθως αυτές απεικονίζονται στο δεκαεξαδικό σύστημα, ώστε να έχουν μια πιο συμπαγή μορφή. Η Εικόνα 3.17 παρουσιάζει ένα τμήμα ενός προγράμματος που έχει γραφεί σε συμβολική γλώσσα, και έχει μεταφραστεί στον αντίστοιχο κώδικα μηχανής για τον επεξεργαστή 8086. Κάθε συμβολική εντολή καταλαμβάνει κάποια Byte που τοποθετούνται σε διαδοχικές θέσεις στη μνήμη (στήλη (α)).

Ένας επεξεργαστής υποστηρίζει (i) τις δικές του εντολές συμβολικής γλώσσας, με (ii) τη δικιά του σύνταξη και τους (iii) δικούς του κανόνες χρήσης της συμβολικής γλώσσας. Αυτά τα 3 στοιχεία καθορίζουν το σετ εντολών του επεξεργαστή (Instruction Set). Όπως φαίνεται στην Εικόνα 3.17 κάποιες εντολές απαιτούν 3 Byte, κάποιες 2 κτλ. Ο αριθμός των Byte που καταλαμβάνει μια εντολή εξαρτάται από αυτό που κάνει. Αυτό ισχύει για τους επεξεργαστές

Σχήμα 3.17: Ο κώδικας μηχανής που έχει προέλθει από τη συμβολική γλώσσα (γ), εκφρασμένος στο δεκαεξαδικό σύστημα (β), τοποθετείται σε διαδοχικές διευθύνσεις μνήμης (α).

πολύπλοκου σετ εντολών (Complex Instruction Set Computing - CISC) αλλά όχι για τους επεξεργαστές ελαττωμένου σετ εντολών (Reduced Instruction Set Computing - RISC). Οι εντολές εισέρχονται στον επεξεργαστή και αποκωδικοποιούνται. Εντολές των επεξεργαστών RISC είναι απλές στην κωδικοποίηση ενώ των CISC, εμφανίζουν μεγάλη πολυπλοκότητα λόγω της διακύμανσης του μεγέθους.

Οι επεξεργαστές χρησιμοποιούν διαφορετικές συμβολικές γλώσσες. Καμία δεν είναι όμοια με κάποια άλλη Το προηγούμενο παράδειγμα γράφτηκε σε Assembly 8086. Αυτή η Assembly ισχύει μόνο στην οικογένεια 8086 των επεξεργαστών της Intel. Άλλες συμβολικές γλώσσες, επειδή είναι βασισμένες σε πολύ διαφορετικό υλικό επεξεργαστών, έχουν διαφορετική σύνταξη. Μερικά παραδείγματα των διαφορετικών εκδόσεων της ίδιας λειτουργίας δίνονται στον Πίνακα 3.1. Σε κάθε περίπτωση, ένας καταχωρητής φορτώνεται με μια τιμή 0x41. Σημειώστε επίσης τους διαφορετικούς τρόπους στη σημείωση δεκαεξαδικού.

Επεξεργαστής	Εντολή
Intel x86	mov al, 41h
Motorola 6800/68HC11	LDAA #$41
Motorola 680x0	move.b #$41,D0
PIC16xx	movlw 0x41
Motorola 56000	move #$0041,A
Intel 80960	lda 0x41,r4

Πίνακας 3.1: Ο κάθε επεξεργαστής χρησιμοποιεί μια δικιά του συμβολική γλώσσα για την επίτευξη της ίδιας λειτουργίας.

Στη συμβολική γλώσσα του κάθε επεξεργαστή, υπάρχουν εντολές που κα-

λύπτουν διάφορες λειτουργίες που συναντώνται σε υπολογιστικά συστήματα, όπως:

Λογικές: ολίσθηση, περιστροφή με διάφορους τρόπους (δεξιά/αριστερή, με κρατούμενο ή χωρίς)

Αριθμητικές Ακεραίων: πρόσθεση, αφαίρεση, πολλαπλασιασμός, διαίρεση

Αριθμητικές Πραγματικών: πρόσθεση, αφαίρεση, πολλαπλασιασμός, διαίρεση , ύψωση σε δύναμη, ρίζα

Εισόδου/Εξόδου: πρόσβαση σε συσκευές εισόδου εξόδου

Πρόσβασης Μνήμης: ανάγνωση ή εγγραφή στη μνήμη, λειτουργίες σε περιοχές μνήμης

Ελέγχου Ροής: διακλαδώσεις υπό συνθήκη ή χωρίς, πράξεις με σημαίες

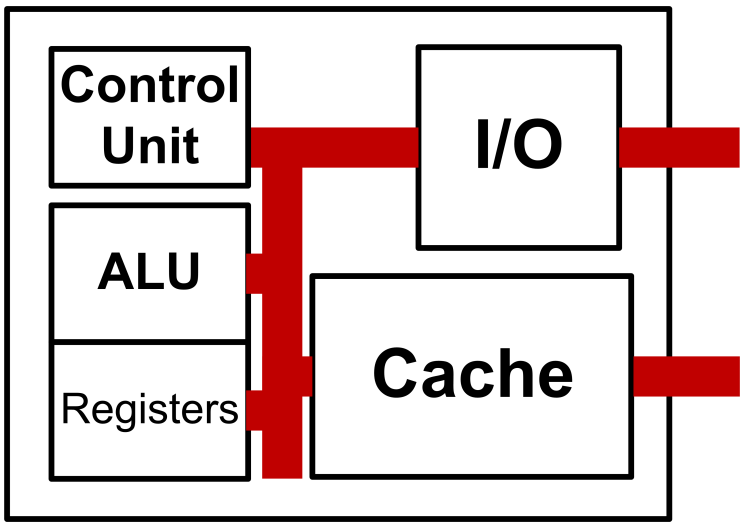
Διανυσματικές Πράξεις: Πράξεις διανυσματικής επεξεργασίας (vector processing)

Οι παραπάνω εντολές, αν και βρίσκονται σε επεξεργαστές γενικού τύπου, δε βρίσκονται όλες σε επεξεργαστές ΕΣ, αφού τα ΕΣ είναι εξειδικευμένα για κάποια εφαρμογή. Αν αυτή η εφαρμογή δε χρειάζεται κάποιες από τις ανωτέρω λειτουργίες, τότε δε θα υλοποιηθεί σε υλικό για να υπάρχει μειωμένο κόστος και κατανάλωση ενέργειας. Να σημειωθεί, ότι κάποιοι επεξεργαστές, όπως ο ARM, είναι τόσο πολύ παραμετροποιήσιμοι, όπου αποτελούνται από δομικά στοιχεία που προσθέτονται η αφαιρούνται κατά τη διάρκεια της κατασκευής ενός συγκεκριμένου επεξεργαστή. Αν για παράδειγμα, δεν απαιτείται μια λειτουργία πραγματικών αριθμών, δε θα τοποθετηθεί η μονάδα FPU, αν δε χρειάζεται η λειτουργία διανυσματικών πράξεων, θα παραληφθεί και η αντίστοιχη μονάδα (NEON). Για αυτό και δεν υπάρχει ένας επεξεργαστής ARM, αλλά πολλοί διαφορετικοί επεξεργαστές που προκύπτουν με συνδυασμό των ανωτέρων λειτουργιών (και ασφαλώς και άλλων δυνατοτήτων, όπως βαθμού διασωλήνωσης, όπως αναγράφεται στην επόμενη ενότητα).

3.4.2 Διασωλήνωση

Όπως είναι κατανοητό ο κώδικας μηχανής για κάθε εντολή που είναι μια αριθμητική τιμή, απαιτεί να αποκωδικοποιηθεί. Η αποκωδικοποίηση σημαίνει τη δημιουργία των κατάλληλων σημάτων ελέγχου προς τις μονάδες του επεξεργαστή προκειμένου να υλοποιήσουν την κατάλληλη λειτουργία. Για παράδειγμα, αν η εντολή προσδιορίζει ότι θα πρέπει να γίνει η πράξη της άθροισης,

τότε κατά την αποκωδικοποίηση θα δημιουργηθούν σήματα προς τον πρώτο καταχωρητή πηγή και προς τον δεύτερο καταχωρητή πηγή, για να τοποθετήσουν τα δεδομένα στις κατάλληλες γραμμές, προς την αριθμητική λογική μονάδα, για να κάνει τη συγκεκριμένη πράξη από το πλήθος των πράξεων που υποστηρίζει, και προς τον καταχωρητή προορισμού, για να δεχτεί και να αποθηκεύσει το αποτέλεσμα. Η Εικόνα 3.18 παρουσιάζει τις βασικές μονάδες του επεξεργαστή, δηλαδή τη μονάδα ελέγχου (control unit), τους καταχωρητές (registers), τη μονάδα αριθμητικών και λογικών πράξεων (arithmetic & logical unit, ALU), την κρυφή μνήμη και τη διεπαφή εισόδου/εξόδου (I/O). Παρατηρήστε ότι η κρυφή μνήμη συνδέεται άμεσα με μια εξωτερική μνήμη και με τον επεξεργαστή, ενώ και η I/O (που συνήθως υπάρχουν πολλοί ελεγκτές I/O) έχει τους δικούς τις ακροδέκτες στο τσιπ. Υπάρχουν πολλοί τρόποι δημιουργίας σημάτων ελέγχου, αλλά δε θα τους αναπτύξουμε αφού αποτελούν ύλη της αρχιτεκτονικής υπολογιστών.



Σχήμα 3.18: Στο εσωτερικό του επεξεργαστή, τα πιο σημαντικά στοιχεία είναι η μονάδα ελέγχου, η αριθμητική λογική μονάδα και οι καταχωρητές.

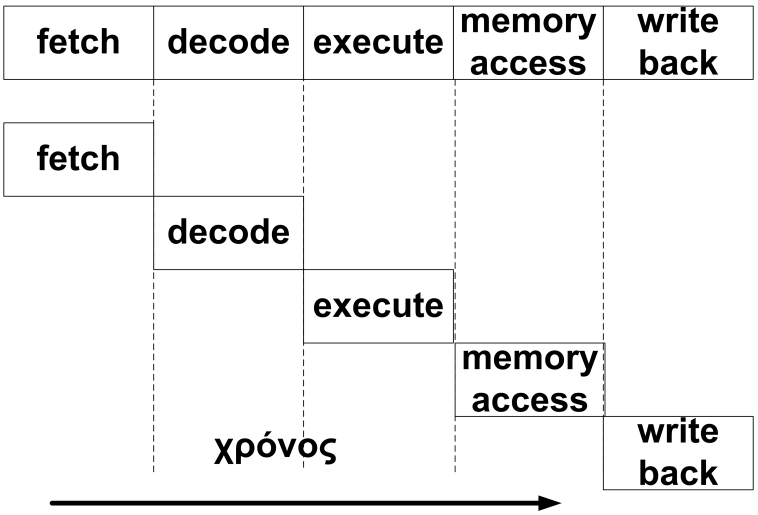
Όπως έχει συζητηθεί, η εκτέλεση μιας εντολής περνάει από 4 στάδια. Προκειμένου να επιτευχθεί χρόνος εκτέλεσης 1 κύκλου ανά εντολή, στους επεξεργαστές RISC έχει διαιρεθεί το τέταρτο στάδιο σε πρόσβαση μνήμης (memory access) και εγγραφή στους καταχωρητές (Write Back), δηλαδή τα 5 στάδια είναι:

1. Προσκόμιση εντολής (fetch)
2. Αποκωδικοποίηση εντολής (decode)
3. Εκτέλεση εντολής (execute)

4. Πρόσβαση στη μνήμη (εγγραφή/ανάγνωση) (memory access)

5. Έγγραφή στους καταχωρητές (Write Back)

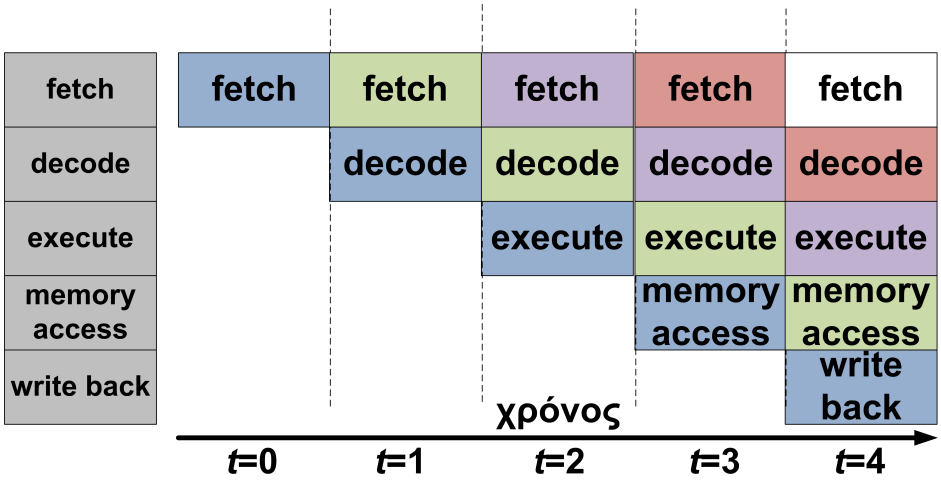
Σε μια απλή σχεδίαση, όσο μια εντολή ήταν στο πρώτο στάδιο, τα υπόλοιπα στάδια δε θα έκαναν κάτι χρήσιμο (Εικόνα 3.19). Για αυτό το λόγο, αποφασίστηκε να βελτιωθεί η λειτουργία του επεξεργαστή, με παρόμοιο τρόπο όπως στην αυτοκινητοβιομηχανία, όπου το όχημα κατασκευάζεται διαδοχικά κατά τη διέλευση από τη γραμμή παραγωγής, και συνεχώς πάνω στη γραμμή παραγωγής υπάρχουν πολλαπλά οχήματα σε διαφορετικές φάσεις ολοκλήρωσης.



Σχήμα 3.19: Κατά την εκτέλεση μιας εντολής σε έναν απλό επεξεργαστή, υπάρχουν μονάδες υλικού που δε χρησιμοποιούνται συνεχώς.

Η βελτίωση της απόδοσης του επεξεργαστή επιτυγχάνεται με παρόμοιο τρόπο με την αυτοκινητοβιομηχανία. Δηλαδή, με την συνεχή χρήση όλων των υπομονάδων του υπολογιστή, με την ταυτόχρονη (ή παράλληλη) ενασχόληση με πολλές εντολές, όπου η κάθε εντολή βρίσκεται σε διαφορετικό στάδιο ολοκλήρωσης. Η Εικόνα 3.20 παρουσιάζει την εξέλιξη κατά τον άξονα του χρόνου της εκτέλεσης εντολών σε ένα επεξεργαστή. Τη χρονική στιγμή 0, εισέρχεται η πρώτη εντολή στο στάδιο Fetch. Όταν αυτή η εντολή προωθηθεί στο στάδιο Decode, τότε μια άλλη εντολή θα προσκομιστεί και θα βρίσκεται στο στάδιο Fetch. Τη χρονική στιγμή t=3, η πρώτη εντολή αφού έχει περάσει από τα στάδια Fetch και Decode, θα βρεθεί στο στάδιο execute, ενώ η 2 εντολή θα βρεθεί στο στάδιο Decode μετά την επιτυχή ολοκλήρωση του σταδίου προσκόμισης. Με αυτόν τον τρόπο κάποια στιγμή και τα 5 στάδια της διασωλήνωσης θα είναι σε χρήση με 5 διαφορετικές εντολές. Οι σύγχρονοι επεξεργαστές για τα

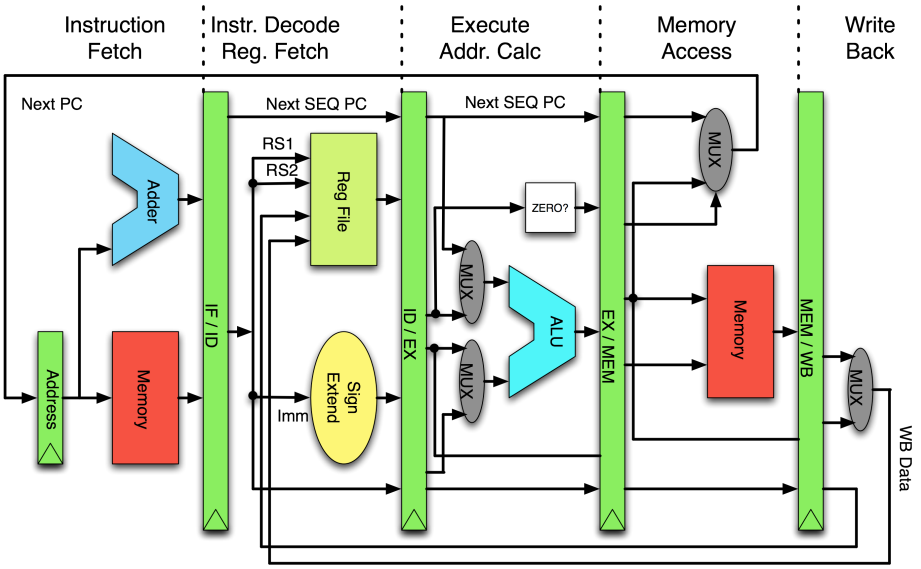
ενσωματωμένα συστήματα έχουν διάφορα επίπεδα διασωλήνωσης. Υπάρχουν επεξεργαστές με αυτά τα τυπικά 5 στάδια διασωλήνωσης (όπως ο MIPS R3000), αλλά και επεξεργαστές με 15 στάδια διασωλήνωσης όπως ο ARM Cortex A15, όπου η διαδικασία fetch έχει διαιρεθεί σε 5 στάδια, η Decode σε 7 στάδια και τα άλλα 3 αφορούν την εκτέλεση, η δε εγγραφή των αποτελεσμάτων απαιτεί 4 ακόμη κύκλους. Επιπρόσθετα, ο ίδιος ο επεξεργαστής έχει δυο διαφορετικές γραμμές διασωλήνωσης: μια γραμμή για τις πράξεις με ακέραιους αριθμούς (integer pipeline), και μια για τις πράξεις με πραγματικούς αριθμούς (floating point unit pipeline).



Σχήμα 3.20: Σε ένα διασωληνωμένο επεξεργαστή, υπάρχουν ταυτόχρονα πολλές εντολές σε διαφορετικά στάδια ολοκλήρωσης.

Μια αναλυτική εικόνα της διασωλήνωσης στον πιο απλό επεξεργαστή ενσωματωμένων συστημάτων MIPS, παρουσιάζεται στην Εικόνα 3.21. Φαίνονται τα 5 στάδια της διασωλήνωσης και για κάθε στάδιο τα βασικά στοιχεία που το συνθέτουν. Στο στάδιο Instruction Fetch, φαίνεται ο καταχωρητής που φέρει την επόμενη διεύθυνση της εντολής, η οποία διευθυνσιοδοτεί τη μνήμη εντολών (memory), ώστε να εξαχθεί η λέξη που θα προωθηθεί προς την αποκωδικοποίηση. Επίσης σε αυτό το στάδιο, έχει προστεθεί και ένας αθροιστής (adder), γιατί η κάθε διεύθυνση απέχει από την επόμενη κατά +4 (υπό κανονικές συνθήκες), αφού κάθε εντολή είναι 32 bit. Στο επόμενο στάδιο γίνεται η αποκωδικοποίηση της εντολής, όπου ενεργοποιούνται οι κατάλληλοι καταχωρητές (registers) από τη συστοιχία καταχωρητών (register file). Σε αυτό το στάδιο υπάρχει και η επέκταση πρόσημου (sign extension) σε περίπτωση που απαιτείται να γίνει μια πράξη, όπου τα μήκη των λέξεων δεν είναι ίδια. Στο στάδιο εκτέλεσης, υπάρχει η ALU όπου σε κάθε είσοδο έχει έναν πολυπλέκτη, που επιλέγει κάθε φορά

την πηγή που θα τοποθετηθεί στην είσοδο της αριθμητικής λογικής μονάδας, επίσης υπάρχει και ένας συγκριτής για τον έλεγχο των συνθηκών. Στο τέταρτο στάδιο, υπάρχει ένας πολυπλέκτης και επιτυγχάνεται η πρόσβαση στη μνήμη δεδομένων, ενώ στο τελευταίο στάδιο υπάρχει η αποθήκευση των ενημερωμένων τιμών στους καταχωρητές (αν η εντολή απαιτεί κάτι τέτοιο). Η διαδρομή που ακολουθεί μια εντολή από την προσκόμιση ως και την εγγραφή, ονομάζεται διαδρομή δεδομένων (datapath) επεξεργαστή.



Σχήμα 3.21: Μια αναλυτική εικόνα της αρχιτεκτονικής της διασωλήνωσης του επεξεργαστή MIPS. Εικόνα από wikipedia.org.

Εκτός από τα βασικά αυτά δομικά στοιχεία, παρατηρείται ότι κάθε στάδιο της διασωλήνωσης διαχωρίζεται από τους καταχωρητές διασωλήνωσης, οι οποίοι διαφυλάσσουν ότι το κρίσιμο μονοπάτι περιορίζεται ανάμεσα στα στοιχεία του κάθε σταδίου. Οι καταχωρητές αυτοί, εκτός από το διαχωρισμό των σταδίων μεταφέρουν και δεδομένα και σήματα ελέγχου, οπότε η καθυστέρηση είναι ενιαία για όλες τις εντολές, τα δεδομένα και τα σήματα ελέγχου. Ασφαλώς, αυτός είναι ένας απλός επεξεργαστής ΕΣ (που όμως χρησιμοποιείται σε πάρα πολλές συσκευές κυρίως δικτυακές modem/routers), και διαφορετικοί επεξεργαστές έχουν πολύ πιο σύνθετες δομές διασωλήνωσης.

Η εισαγωγή της διασωλήνωσης σε ένα σύστημα, δεν αυξάνει την ταχύτητα εκτέλεσης μιας εντολής, αφού έτσι και αλλιώς η εντολή διέρχεται από τα ίδια στάδια. Αυξάνει όμως την απόδοση του όλου συστήματος, και για k στάδια διασωλήνωσης η απόδοση (ρυθμός εντολών στη μονάδα του χρόνου) αυξάνεται κατά k. Όμως, η χρήση της διασωλήνωσης αυξάνει την πολυπλοκότητα

του συστήματος και δημιουργεί μια νέα εστία κινδύνων, που ονομάζονται κίνδυνοι διασωλήνωσης. Οι κυριότεροι κίνδυνοι είναι: (i) δομικοί κίνδυνοι, (ii) κίνδυνοι ελέγχου και (iii) κίνδυνοι δεδομένων. Οι δομικοί κίνδυνοι υπάρχουν όταν το ίδιο δομικό στοιχείο απαιτείται να χρησιμοποιηθεί από 2 διαφορετικά στάδια της διασωλήνωσης. Για παράδειγμα, στην Εικόνα 3.21, το αρχείο καταχωρητών μπορεί να χρησιμοποιηθεί και στο στάδιο 2 και στο στάδιο 5. Αυτός ο κίνδυνος μπορεί να εξαλειφθεί με την προσθήκη υλικού (π.χ. μιας επιπρόσθετης θύρας εγγραφής/ανάγνωσης στο αρχείο των καταχωρητών) ή με την προσθήκη καθυστέρησης στο ανάλογο στάδιο (και κατ' επέκταση σε όλη τη διασωλήνωση, αφού αν δεν προχωρήσει ένα στάδιο στο επόμενο δε μπορεί να συνεχίσει κανένα). Οι κίνδυνοι ελέγχου δημιουργούνται όταν υπάρχει διακλάδωση ή κλήση συνάρτησης, όπου σε αυτή την περίπτωση δεν πρέπει να φορτωθεί ο καταχωρητής εντολών με την εντολή που έπεται (+4 Byte στο MIPS), αλλά με την εντολή που βρίσκεται πολλά Byte μακρυά και αφορά την πρώτη εντολή της συνάρτησης ή της εναλλακτικής διαδρομής. Αυτό μπορεί να λυθεί πάλι με την προσθήκη υλικού που θα εκτελεί μια λειτουργία πρόβλεψης επόμενης εντολής (με συνηθισμένα ποσοστά επιτυχίας 90%) ή με καθυστέρηση έως ότου υπολογιστεί η διεύθυνση της νέας εντολής στο στάδιο της εκτέλεσης. Τέλος, οι κίνδυνοι δεδομένων προκύπτουν όταν απαιτούνται δεδομένα που έχουν υπολογιστεί αλλά ακόμη δεν έχουν γραφεί (γιατί είναι στο στάδιο 4 ή 5). Ομοίως, μια λύση είναι η καθυστέρηση μέχρι να αποθηκευτούν στο αρχείο των καταχωρητών ή με την προσθήκη υλικού που θα δημιουργήσει μονοπάτια συντόμευσης, ώστε να μεταφέρουν τα δεδομένα άμεσα εκεί που χρειάζονται, παρακάμπτοντας προσωρινά την τυπική λειτουργιά.

Μερικοί επεξεργαστές για να αυξήσουν τον ρυθμό απόδοσης, έχουν ταυτόχρονα πολλές λειτουργικές μονάδες (π.χ. 3 ALU, 2 FPU κτλ). Όταν υπάρχει ανάγκη για χρήση μιας από αυτές τις μονάδες, τότε χρησιμοποιείται ένας χρονοδρομολογητής που αποφασίζει ποια μονάδα θα ασχοληθεί με κάθε συγκεκριμένη εντολή, ενώ μέχρι να ελευθερωθεί μια μονάδα που απαιτεί μια εντολή, την καθυστερεί και δεν την προωθεί. Για να προωθηθεί μια εντολή από την αναμονή στην εκτέλεση, θα πρέπει να τηρούνται οι εξής προϋποθέσεις: θα πρέπει όλες οι εξαρτήσεις εισόδου να έχουν υπολογιστεί, και η μονάδα που απαιτείται να είναι ελεύθερη προς χρήση. Συνήθως, ο χρονοδρομολογητής τοποθετεί τις εντολές σε ουρές αναμονής ανά λειτουργική μονάδα, που ονομάζονται σταθμοί αναμονής (reservation stations). Προκειμένου να επιτευχθεί η μέγιστη απόδοση, ο χρονοδρομολογητής είναι κατασκευασμένος σε υλικό και τα πάντα γίνονται και υπολογίζονται κατά το χρόνο εκτέλεσης. Αυτού του είδους οι επεξεργαστές ονομάζονται υπερβαθμωτοί (superscalar). Αρκετοί υπερβαθμωτοί επεξεργαστές επιτρέπουν την καλύτερη εκμετάλλευση των πολλαπλών λειτουργικών μονάδων με την εκτέλεση εντολών εκτός σειράς (out of order execution). Ενώ οι τυπικοί επεξεργαστές εκκινούν την εκτέλεση μιας εντολής σε κάθε κύκλο ρολογιού

(single issue), οι πιο σύνθετοι επεξεργαστές μπορούν και εκκινούν έως και 4 εντολές κάθε κύκλο (multiple issue), αφού έχουν διασφαλίσει ότι επιτρέπεται από τις εξαρτήσεις δεδομένων και τη διαθεσιμότητα των λειτουργικών μονάδων. Οι επεξεργαστές εκτέλεσης εκτός σειράς, έχουν επιπρόσθετο υλικό για τη σειριοποίηση των αποτελεσμάτων στο τέλος (commitment stage) και την ολοκλήρωση της εκτέλεσης (retirement stage).

Η αντιδιαμετρική αρχιτεκτονική, ως προς τους υπερβαθμωτούς επεξεργαστές, είναι οι επεξεργαστές πολύς μεγάλης λέξης εντολής (very long instruction word, VLIW). Οι επεξεργαστές αυτοί έχουν πολλαπλές λειτουργικές μονάδες, αλλά δεν έχουν υλικό για τη εύρεση των εξαρτήσεων και την εκτέλεση εκτός σειράς. Όλες οι λειτουργίες κωδικοποιούνται σε μια μεγάλη εντολή (48 bit έως και 128 bit), όπου το opcode φέρει κωδικοποιημένες τις λειτουργίες που επιτελεί η κάθε λειτουργική μονάδα. Αυτό σημαίνει ότι ο έλεγχος εξαρτήσεων γίνεται κατά το στάδιο της μεταγλώττισης και όχι κατά το χρόνο εκτέλεσης. Τα πειραματικά αποτελέσματα έχουν δείξει ότι καλύτερα αποτελέσματα υπάρχουν όταν οι αποφάσεις λαμβάνονται κατά το χρόνο της εκτέλεσης, αλλά με αυξημένο τίμημα στο υλικό και στην κατανάλωση ενέργειας.

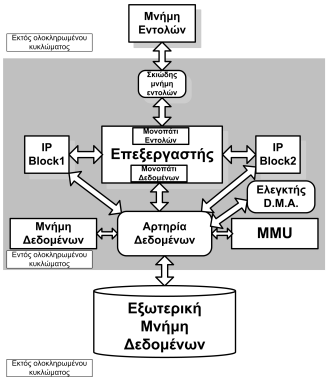
Τα ΕΣ κυμαίνονταν σε ένα πολύ μεγάλο εύρος αρχιτεκτονικών, οπότε σε ΕΣ υψηλών επιδόσεων βρίσκονται και υπερβαθμωτοί επεξεργαστές και επεξεργαστές VLSI. Στην πλειονότητα των περιπτώσεων όμως, που απαιτείται ένα οικονομικό και μικρό σύστημα με χαμηλό ενεργειακό προφίλ, επιλέγονται απλοί επεξεργαστές χωρίς τις επιπρόσθετες αυτές δυνατότητες, που συναντώνται σε κάθε σύγχρονο επεξεργαστή υψηλών επιδόσεων όπως Intel Core I7.

3.5 Περιγραφή της αρχιτεκτονικής-στόχου για ένα ΕΣ

Κάθε μεθοδολογία σχεδιασμού βασίζεται σε κάποια αρχιτεκτονική. Η επιλογή της αρχιτεκτονικής πρέπει να γίνει προσεκτικά, διαφορετικά μπορεί να μην είναι πρακτικά υλοποιήσιμη. Η αρχιτεκτονική θα πρέπει να έχει όσο το δυνατόν χαμηλότερο κόστος, να μπορεί να υλοποιηθεί πραγματικά (δηλαδή να μην είναι η μεθοδολογία μόνο θεωρητική), και να είναι παραμετροποιήσιμη, δηλαδή να παρέχει στο σχεδιαστή ευελιξία στο να τροποποιεί κάποια χαρακτηριστικά που ίσως ταιριάζουν καλύτερα στο πρόβλημά του. Πριν αρχίσουμε την έρευνά μας, προβήκαμε σε μια ανάλυση των αρχιτεκτονικών των ενσωματωμένων συστημάτων διαφόρων κατασκευαστών, σημειώσαμε τα κοινά χαρακτηριστικά που έχουν, τα θετικά και αρνητικά σημεία τους, και αναπτύξαμε τη δικιά μας αρχιτεκτονική.

Έτσι, λοιπόν, η μεθοδολογία σχεδιασμού ΕΣ που παρουσιάζουμε βασίζεται

σε μια γενική αρχιτεκτονική, η οποία συναντάται αρκετά συχνά στα ενσωματωμένα συστήματα (Σχήμα 3.22). Όπως φαίνεται στο Σχήμα 3.22, η αρχιτεκτονική αποτελείται από δομοστοιχεία εντός και εκτός ολοκληρωμένου κυκλώματος. Η εξωτερική μνήμη δεδομένων και η μνήμη εντολών βρίσκονται εκτός ολοκληρωμένου κυκλώματος, ενώ η μνήμη δεδομένων (ή επιπλήνθια μνήμη), η μονάδα διαχείρισης μνήμης (Memory Management Unit, MMU), ο ελεγκτής απευθείας πρόσβασης στη μνήμη (Direct Memory Access, DMA), η σκιώδης (ή κρυφή) μνήμη εντολών, ο επεξεργαστής και τα εξειδικευμένα στοιχεία για την εφαρμογή (IP-blocks) βρίσκονται πάνω στο ολοκληρωμένο κύκλωμα. Τα δομοστοιχεία αυτά θα τα αναπτύξουμε στη συνέχεια, μαζί με το λόγο για τον οποίον έχουν επιλεγεί. Πρέπει να σημειωθεί βέβαια ότι ο τύπος του συγκεκριμένου δομοστοιχείου, για παράδειγμα αν ο τύπος της μνήμης θα είναι SRAM ή DRAM, δεν είναι ρητά καθορισμένος, αλλά καθορίζεται από το σχεδιαστή, ή από τη διαθεσιμότητα των υλικών.



Σχήμα 3.22: Η μεθοδολογία μας χρησιμοποιεί μια γενική αρχιτεκτονική ενσωματωμένων συστημάτων και δεν περιορίζεται σε συγκεκριμένα δομοστοιχεία.

3.5.1 Υπάρχει πρόβλημα με την επίδοση των σημερινών συστημάτων

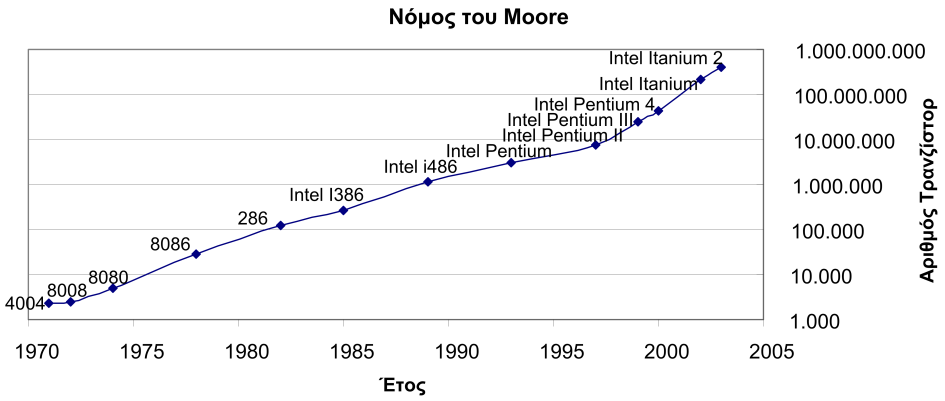
Με τον όρο επίδοση ενός συστήματος αναφερόμαστε συνήθως σε δύο μεγέθη, ανάλογα με το επίπεδο σχεδιαστικής αφαίρεσης που βρισκόμαστε. Στο υψηλό επίπεδο συστήματος (system level) χρησιμοποιούμε το μέγεθος της επεξεργαστικής ισχύος που μετράται σε εκατομμύρια εντολών ανά δευτερόλεπτο (Millions of Instructions Per Second-MIPS). Αν και είναι ευρέως διαδεδομένο αυτό το μέγεθος ως μέτρο σύγκρισης επιδόσεων, εντούτοις δε χαρακτηρίζει όλο το σύστημα, αλλά μόνο τον επεξεργαστή του. Υπάρχουν σε ένα σύστημα αρκετοί παράγοντες που επιδρούν αρνητικά στην επίδοση, με συνέπεια να υπάρχει

μια μεγάλη απόκλιση από αυτήν την ανώτερη θεωρητική τιμή. Σε πιο χαμηλό επίπεδο, δηλαδή στο επίπεδο κυκλώματος, χρησιμοποιούμε τα μεγέθη του κρίσιμου μονοπατιού (critical path) και της παροχής δεδομένων (throughput). Το πρώτο μετρικό υποδηλώνει το μέγιστο χρόνο που απαιτείται για την επεξεργασία κάποιων δεδομένων, και είναι ο χρόνος από τη στιγμή που τα δεδομένα θα εισέλθουν στο κύκλωμα μέχρι τη στιγμή που η έξοδος θα είναι διαθέσιμη στην αρτηρία δεδομένων εξόδου (output data bus). Το δεύτερο μετρικό υποδηλώνει το χρόνο που χρειάζεται το κύκλωμα για να επεξεργαστεί διαδοχικά δεδομένα. Αν το κύκλωμα είναι σχεδιασμένο με τέτοιο τρόπο ώστε, ενώ δεν έχει ολοκληρώσει ακόμη την επεξεργασία κάποιων δεδομένων, να δέχεται και άλλα (νέα) δεδομένα, τότε έχει μεγαλύτερη παροχή δεδομένων από ένα κύκλωμα που πρέπει πρώτα να επεξεργαστεί τα δεδομένα και να τα εξάγει στην αρτηρία δεδομένων, προτού δεχτεί καινούργια δεδομένα. Αν και θα μπορούσαμε να χρησιμοποιήσουμε την παροχή δεδομένων για να χαρακτηρίσουμε την επίδοση συστημάτων, εντούτοις στους βιομηχανικούς και ακαδημαϊκούς κύκλους χρησιμοποιείται το μετρικό MIPS των επεξεργαστών των συστημάτων.

Εκτός από αυτά τα μετρικά, πολύ συχνά χρησιμοποιείται και το μετρικό της συχνότητας λειτουργίας του επεξεργαστή. Η συχνότητα λειτουργίας του επεξεργαστή ορίζεται ως η συχνότητα λειτουργίας του ρολογιού με το οποίο χρονίζεται ο επεξεργαστής. Ασφαλώς, η χρήση του όρου αυτού είναι δόκιμη μόνο στην περίπτωση που έχουμε ρολόι στο σύστημά μας, δηλαδή όταν αναφερόμαστε σε ένα 'σύγχρονο' (synchronous) ψηφιακό σύστημα, σε αντιδιαστολή προς ένα 'ασύγχρονο' (asynchronous) σύστημα. Στην περίπτωση που έχουμε στο κύκλωμα παραπάνω από ένα ρολόι αναφέρουμε όλες τις συχνότητες στις οποίες λειτουργεί το σύστημα (π.χ. ρολόι πυρήνα 833 Mhz, ρολόι αρτηρίας δεδομένων 133 Mhz κτλ). Το μετρικό αυτό είναι υποκειμενικό και δεν μπορεί να χρησιμοποιηθεί για τη σύγκριση επεξεργαστών διαφορετικών οικογενειών. Αυτό οφείλεται στο γεγονός ότι ένας επεξεργαστής με χαμηλή συχνότητα μπορεί να εκτελεί περισσότερες παράλληλες εντολές (περισσότερα MIPS) από ότι ένας επεξεργαστής με υψηλότερη συχνότητα. Είναι γνωστό εξάλλου, ότι κάποιες εταιρείες που κατασκευάζουν επεξεργαστές διαφημίζουν τα προϊόντα τους με τις ισοδύναμες συχνότητες επεξεργασίας και όχι με τις πραγματικές. Για παράδειγμα, η εταιρεία Advanced Micro Devices (AMD) διαφημίζει ότι ο επεξεργαστής AMD 3200 Mhz είναι PR 4000+ Mhz, δηλαδή ότι είναι ισοδύναμος με έναν Intel Pentium συχνότητας 4000 Mhz.

Πάντως, όποιο μετρικό και να χρησιμοποιήσουμε για να συγκρίνουμε τους επεξεργαστές, που έχουν παρουσιαστεί τις τελευταίες δύο δεκαετίες, θα διαπιστώσουμε ότι η επίδοση ενός συστήματος ακολουθεί το νόμο του Moore [23]. Ο Moore έκανε μια πολύ σημαντική παρατήρηση και πρόβλεψη το 1965, όταν εργαζόταν ως μηχανικός στην εταιρεία Fairchild. Παρατήρησε λοιπόν ότι η πυκνότητα των ολοκληρωμένων κυκλωμάτων που υλοποιούνται στο πυρίτιο ακο-

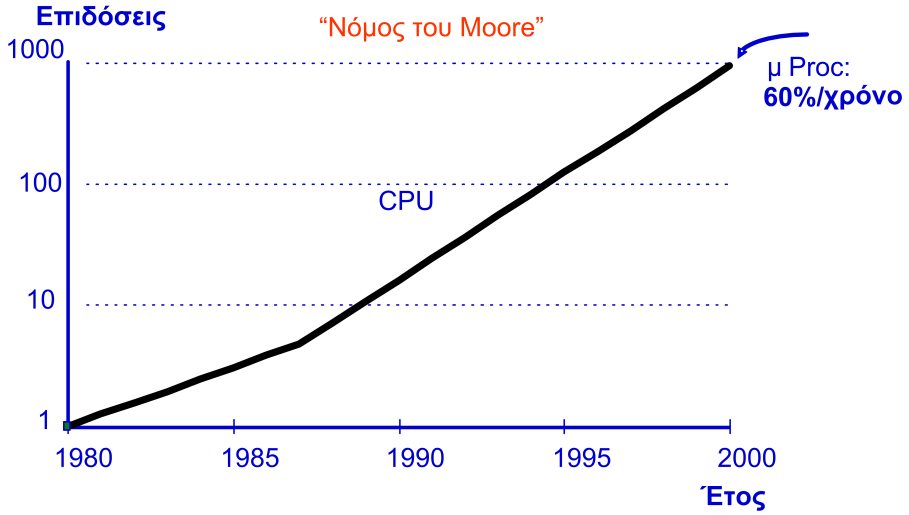
λουθεί μια αυξητικά εκθετική πορεία με συνέπεια κάθε 18 μήνες να διπλασιάζεται. Έτσι, ο Moore πρότεινε μια μαθηματική σχέση, η οποία έκτοτε καθιερώθηκε ως νόμος του Moore. Αν και έχουν περάσει 40 χρόνια από τότε, ο νόμος του Moore ισχύει ακόμη και σήμερα. Οι εταιρείες κατασκευής ολοκληρωμένων κυκλωμάτων, όπως η AMD και η INTEL, παρουσιάζουν συνεχώς νέα μοντέλα και νέες αρχιτεκτονικές επεξεργαστών, που όμως ακολουθούν πιστά το νόμο αυτό (Σχήμα 3.23).



Σχήμα 3.23: Η πρόβλεψη του Gordon Moore το 1965, για το διπλασιασμό του αριθμού των τρανζίστορ στο πυρίτιο κάθε 18 μήνες, συνεχίζει να ισχύει ακόμη και σήμερα.

Μια από τις συνέπειες του νόμου του Moore είναι ότι με την αύξηση της πυκνότητας ολοκλήρωσης επηρεάζεται θετικά και η συχνότητα λειτουργίας του επεξεργαστή και, κατ' επέκταση, οι επιδόσεις του. Συγκεκριμένα, αν σημειώσουμε τις επιδόσεις των επεξεργαστών ως προς τη χρονιά που παρουσιάστηκαν στην αγορά, θα διαπιστώσουμε ότι ακολουθούν μια παράλληλη αυξητική πορεία με την αύξηση της πυκνότητας ολοκλήρωσης, που σημαίνει ότι οι επιδόσεις των επεξεργαστών αυξάνουν κατά 60% ετησίως (Νόμος του Moore για τις επιδόσεις επεξεργαστών, Σχήμα 3.24). Αν και υπάρχουν ερευνητές που υποστήριξαν ότι ο νόμος αυτός δεν μπορεί να συνεχίσει να ισχύει γιατί υπάρχει ένα όριο στην τεχνολογία κατασκευής των επεξεργαστών, εντούτοις η συνεχής εκμετάλλευση της παραλληλίας σε επίπεδο εντολών επεξεργαστή (η εκτέλεση αρκετών εντολών ταυτόχρονα), οι μικρο-αρχιτεκτονικές ανακαλύψεις, η σμίκρυνση των γεωμετρικών διαστάσεων της τεχνολογίας, οι νέες αρχιτεκτονικές και οι πιο βαθιές διαδοχικές διοχετεύσεις έχουν οδηγήσει στο να έχουν οι επεξεργαστές πολύ υψηλές συχνότητες λειτουργίας. Από ότι φαίνεται μάλιστα, ο

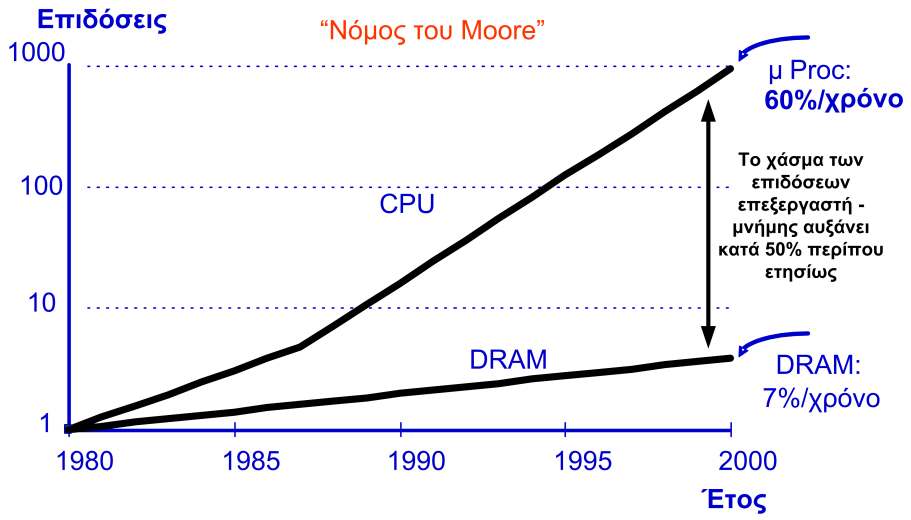
νόμος αυτός θα συνεχίσει να ισχύει για αρκετά χρόνια ακόμη· υπάρχουν προβλέψεις ότι θα ισχύει ως και τη δεύτερη δεκαετία του 2000.



Σχήμα 3.24: Μια διαφορετική ερμηνεία του νόμου του Moore είναι ότι η επίδοση των επεξεργαστών αυξάνει κατά 60% κάθε χρόνο.

Κοιτώντας κάποιος τα δεδομένα αυτά θα μπορούσε να συμπεράνει ότι ίσως δεν υπάρχει πρόβλημα στην επίδοση των συστημάτων, αφού οι επεξεργαστές που αποτελούν την καρδιά του συστήματος, έχουν ολοένα και μεγαλύτερες συχνότητες επεξεργασίας. Δυστυχώς όμως, ενώ οι τεχνολογικές εξελίξεις στους επεξεργαστές τους έκαναν να έχουν συχνότητες λειτουργίας που ακολουθούσαν μια εκθετική πορεία, οι συχνότητες λειτουργίας της μνήμης, ενός άλλου βασικού και αναπόσπαστου μέρους ενός συστήματος, υστέρησαν σημαντικά.

Η μνήμη είναι αναπόσπαστο στοιχείο ενός συστήματος και δε μπορεί να λειτουργήσει ένα σύστημα χωρίς καμία μορφή μνήμης. Η μνήμη χρησιμοποιείται για την αποθήκευση ή ανάκτηση δεδομένων κατά τη λειτουργία του συστήματος. Όπως κάθε κύκλωμα ή στοιχείο, έτσι και οι μνήμες ακολούθησαν μια αναπτυξιακή πορεία κατά τα τελευταία 40 χρόνια. Η βελτίωσή τους φαίνεται τόσο στο μέγεθος (η μνήμη 1 KB του παρελθόντος αντικαταστάθηκε από μνήμες 1 GB) όσο και στη συχνότητα λειτουργίας της μνήμης (η συχνότητα λειτουργίας των 8 Mhz έγινε 233 Mhz). Εντούτοις, η βελτίωση στη συχνότητα λειτουργίας δεν ακολούθησε τους ταχύτατους ρυθμούς ανάπτυξης των επεξεργαστών, με συνέπεια τη δημιουργία ενός ολοένα αυξανόμενου χάσματος ανάμεσα σε αυτήν και τον επεξεργαστή (Σχήμα 3.25). Έχει υπολογιστεί ότι ενώ οι επιδόσεις των επεξεργαστών αυξάνονται κατά 60% ετησίως, οι επιδόσεις των μνημών αυξάνονται μόνο κατά 7%, που σημαίνει ότι το χάσμα ανάμεσά τους αυξάνει κατά 50% περίπου ετησίως.

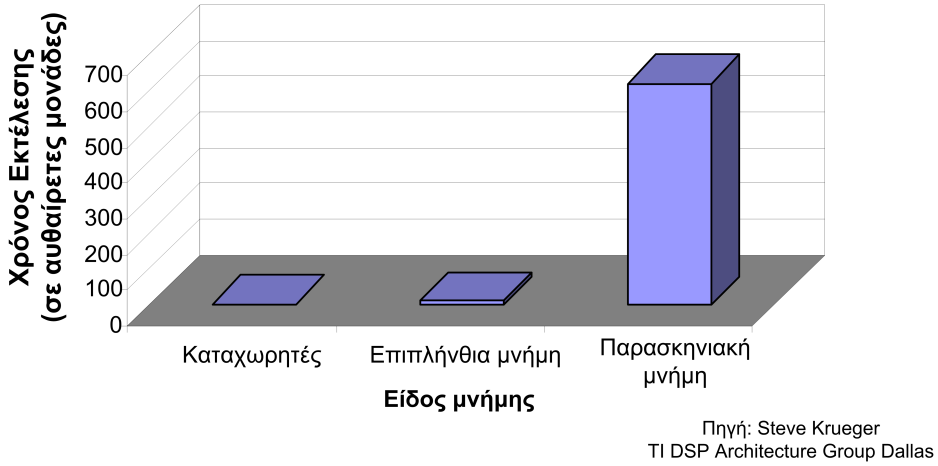


Σχήμα 3.25: Ο διαφορετικός ρυθμός ανάπτυξης και βελτίωσης των επεξεργαστών δημιουργεί ένα ολοένα αυξανόμενο χάσμα ανάμεσα σε αυτούς και τις μνήμες.

Προκειμένου να αντιμετωπιστεί κάπως το πρόβλημα της καθυστέρησης στην πρόσβαση της μνήμης, έχει υιοθετηθεί η χρήση μνήμης πάνω στο ολοκληρωμένο κύκλωμα (on-chip memory), δηλαδή η χρήση μνήμης πάνω στο ολοκληρωμένο κύκλωμα του επεξεργαστή. Η μνήμη αυτή είναι αρκετά πιο γρήγορη από τη μνήμη παρασκηνίου (background memory), δηλαδή τη μνήμη που βρίσκεται εκτός ολοκληρωμένου κυκλώματος (εξωτερική μνήμη), αφού συνήθως λειτουργεί στη συχνότητα του επεξεργαστή. Η μνήμη αυτή, όμως, έχει πολύ μεγάλο κόστος υλοποίησης, και για αυτό το λόγο συνήθως είναι πολύ μικρή σε μέγεθος (της τάξης των μερικών εκατοντάδων KB). Έτσι, απαιτούνται πάλι προσβάσεις στην εξωτερική μνήμη, οι οποίες έχουν σημαντικό αντίκτυπο στην επίδοση, αλλά και στην κατανάλωση ισχύος, όπως θα συζητηθεί στη συνέχεια.

Οι συνέπειες του χάσματος αυτού είναι σημαντικές στην επίδοση των σημερινών συστημάτων, ιδιαίτερα όταν τα συστήματα αυτά εκτελούν εφαρμογές με έντονη χρήση της μνήμης. Έτσι, ενώ για παράδειγμα απαιτείται ένας κύκλος ρολογιού για πρόσβαση στη μνήμη καταχωρητών του επεξεργαστή και ένας με δύο κύκλους για πρόσβαση στη μνήμη εντός ολοκληρωμένου κυκλώματος, η εξωτερική μνήμη απαιτεί από 10 έως 20 κύκλους ρολογιού για πρόσβαση. Κατά τη διάρκεια που γίνεται η πρόσβαση στην εξωτερική μνήμη, ο επεξεργαστής περιμένει (stall) να αναγνωστούν ή να γραφούν τα δεδομένα. Το Σχήμα 3.26 αναπαριστά τις επιπτώσεις στο χρόνο εκτέλεσης μιας τυπικής εφαρμογής πολυμέσων (αποκωδικοποίηση MPEG4), όταν οι προσβάσεις που αυτή απαιτεί, γίνονται εξ' ολοκλήρου στη μνήμη καταχωρητών ή τη μνήμη εντός ολοκλη-

ρωμένου κυκλώματος ή την παρασκηνιακή μνήμη. Είναι φανερό, λοιπόν, ότι η αρκετά χαμηλότερη συχνότητα λειτουργίας της μνήμης ενός συστήματος αποτελεί κυρίαρχο ανασχετικό παράγοντα επίτευξης υψηλής επίδοσης. Δεν είναι υπερβολή, μάλιστα, να συμφωνήσουμε με την άποψη αρκετών ερευνητών ότι τα σημερινά συστήματα "ξοδεύουν" πάνω από το μισό του χρόνου εκτέλεσης μιας εφαρμογής περιμένοντας να ολοκληρωθούν οι προσβάσεις στην παρασκηνιακή μνήμη.



Σχήμα 3.26: Η καθυστέρηση πρόσβασης στην παρασκηνιακή μνήμη έχει δραματικές επιπτώσεις πάνω στην επίδοση ενός συστήματος.

Όμως, το πρόβλημα δεν περιορίζεται μόνο στην επίδοση ενός συστήματος. Κατά την τελευταία δεκαετία έχει εμφανιστεί ένα ακόμη μετρικό, που χαρακτηρίζει ένα σύστημα, και αυτό είναι η καταναλισκόμενη ενέργεια. Η τεχνολογική ανάπτυξη και η δραματική βελτίωση που έχει συντελεστεί στα ολοκληρωμένα κυκλώματα έχει επηρεάσει αρνητικά την κατανάλωση ενέργειας, όπως θα αναπτυχθεί στη συνέχεια.

3.5.2 Υπάρχει πρόβλημα με την κατανάλωση ενέργειας ενός συστήματος

Μέχρι την προηγούμενη δεκαετία, οι σχεδιαστές είχαν ως στόχο τη μεγιστοποίηση της επίδοσης, με όσο το δυνατόν μικρότερο κόστος. Έτσι, σχεδόν παντού οι μηχανικοί έδιναν ιδιαίτερη έμφαση στο να παρουσιάζουν προϊόντα ολοένα και πιο 'γρήγορα', παραμελώντας τελείως την κατανάλωση ενέργειας. Από τα μέσα περίπου της δεκαετίας του 1990, όμως, αυτό έχει αλλάξει δραματικά. Η κατανάλωση ενέργειας αποτελεί ένα ισοδύναμο (ή και πιο σημαντικό)

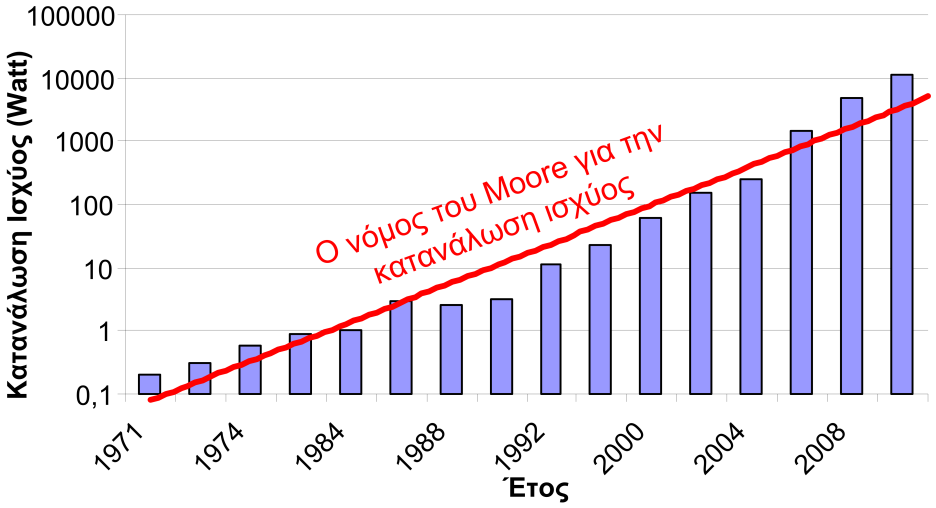
μετρικό χαρακτηρισμού ενός συστήματος που πρέπει να λαμβάνεται σοβαρά υπόψη σε όλα τα στάδια του σχεδιασμού του. Σήμερα, είναι αδιανόητο να μην αναφέρεται η κατανάλωση ενέργειας ενός συστήματος (ή επεξεργαστή), μαζί με τα υπόλοιπα χαρακτηριστικά, όταν αυτό παρουσιάζεται στο καταναλωτικό κοινό.

Οι λόγοι που έκαναν να αναδυθεί αυτό το μετρικό από την αφάνεια, κατά την τελευταία δεκαετία, και το έκαναν εξίσου σημαντικό με την επίδοση και το κόστος είναι πολλοί. Ο πιο σημαντικός λόγος από όλους είναι η εκτεταμένη χρήση των φορητών συσκευών. Προσωπικοί φορητοί υπολογιστές, ψηφιακοί βοηθοί, κινητά τηλέφωνα, φορητές μονάδες αναπαραγωγής βίντεο, φορητά τερματικά πολυμέσων και άλλες συσκευές έχουν υιοθετηθεί από τους περισσότερους καταναλωτές και έχουν γίνει αναπόσπαστα στοιχεία της καθημερινής τους ζωής. Η μέχρι τώρα εξέλιξή τους, μάλιστα, προδιαγράφει ένα λαμπρό μέλλον για τις φορητές συσκευές με ολοένα αυξανόμενα ποσοστά διεισδυτικότητας στην καταναλωτική αγορά. Θα αναφερθούμε περαιτέρω στις φορητές συσκευές και γενικότερα στα ενσωματωμένα συστήματα, στο δεύτερο κεφάλαιο.

Οι επιδόσεις των σημερινών επεξεργαστών, όπως είδαμε, διπλασιάζονται κάθε 18 μήνες (Σχήμα 3.24). Μια από τις συνέπειες αυτού του γεγονότος είναι ότι και η κατανάλωση ενέργειας αυτών αυξάνεται εκθετικά (Σχήμα 3.27). Αν και αρχικά οι μικροεπεξεργαστές κατανάλωναν λίγα Watt-hour, στην εποχή μας υπάρχουν επεξεργαστές που έχουν κατανάλωση ισχύος πάνω από 80 Watt. Τέτοια επίπεδα κάνουν επιτακτική τη χρήση συστημάτων ψύξης, η οποία γίνεται ολοένα πιο δύσκολη και ακριβή. Αν συνεχιστεί ο ρυθμός αυτός αύξησης της κατανάλωσης, τότε σε λίγα χρόνια θα έχουμε καταναλώσεις εκατοντάδων Watt-hour, που ασφαλώς δεν επιτρέπεται στα περισσότερα συστήματα.

Σχετικά με τις φορητές συσκευές, το πρόβλημα της κατανάλωσης ενέργειας είναι πολύ πιο σοβαρό από ότι στα επιτραπέζια συστήματα. Οι σημερινές μπαταρίες ιόντων λιθίου έχουν μια ενεργειακή πυκνότητα περίπου 100 Wh/Kg. Αυτό σημαίνει ότι αν θέλαμε να χρησιμοποιήσουμε έναν επεξεργαστή που έχει απαίτηση ισχύος 50 Watt, θα απαιτούνταν μια μπαταρία 2 Kg, που δεν ενδείκνυται για μια φορητή συσκευή. Προκειμένου να αντιμετωπιστεί εν μέρει, αυτό το πρόβλημα, οι εταιρείες έχουν παρουσιάσει στο καταναλωτικό κοινό επεξεργαστές χαμηλής κατανάλωσης ενέργειας, οι οποίοι όμως έχουν πολύ μικρές επιδόσεις και δεν μπορούν να αντεπεξέλθουν στην πολυπλοκότητα των σημερινών εφαρμογών.

Προφανώς, η λύση στο πρόβλημα της κατανάλωσης ισχύος με μείωση των επιδόσεων δεν είναι αποτελεσματική. Οι χρήστες των φορητών συσκευών απαιτούν την εκτέλεση των εφαρμογών τους με όσο το δυνατόν μικρότερη κατανάλωση ισχύος, και όσο γίνεται πιο γρήγορα. Όταν δεν ικανοποιούνται οι επεξεργαστικές απαιτήσεις, ο χρήστης θα έχει κάποια προβλήματα (π.χ. να σταματάει η αναπαραγωγή κάποιας ταινίας, ή να διακόπτεται η επικοινωνία, ή να υπάρ-



Σχήμα 3.27: Άλλη μια ερμηνεία του νόμου του Moore είναι ότι η κατανάλωση ενέργειας των συστημάτων αυξάνει εκθετικά με το ημερολογιακό έτος.

χουν αρκετές καθυστερήσεις στην επεξεργασία των δεδομένων του). Οι χρήστες αυτών των φορητών συστημάτων δεν μπορούν να ανεχθούν τέτοιες προβληματικές λειτουργίες, και έτσι σύντομα αντικαθιστούν αυτές τις συσκευές με άλλες που καλύπτουν αυτές τις απαιτήσεις.

Αλλά, η μεγάλη κατανάλωση ισχύος δεν είναι το μόνο πρόβλημα στη φορητότητα. Η κατανάλωση ισχύος συνδέεται άμεσα με το κόστος ψύξης και ενσωμάτωσης της στο σύστημα. Από τη στιγμή που η θερμότητα πρέπει να αποβάλλεται στο περιβάλλον μέσω της συσκευασίας και της μονάδας ψύξης, το κόστος γίνεται ιδιαίτερα υψηλό έως και απαγορευτικό. Ένας επεξεργαστής, με μεγαλύτερη κατανάλωση ισχύος από έναν άλλο, αποβάλει περισσότερη θερμότητα στο περιβάλλον και χρειάζεται μια μεγαλύτερη μονάδα ψύξης. Εκτός από το μέγεθός της, η μεγαλύτερη μονάδα ψύξης προκαλεί και περισσότερο θόρυβο (ηχορύπανση) λόγω των μηχανικών μερών που περιλαμβάνει. Συνεπώς, υπάρχει ένα ξεκάθαρο πλεονέκτημα στη μείωση της κατανάλωσης ισχύος, τόσο για να μειωθεί το κόστος όσο και για να μειωθεί ο θόρυβος που θα παράγει η συσκευή.

Εκτός από τα παραπάνω οφέλη, υπάρχει και το θέμα της αξιοπιστίας (reliability). Τα συστήματα με μεγάλη κατανάλωση ισχύος αναπτύσσουν υψηλές θερμοκρασίες. Οι υψηλές θερμοκρασίες επιδρούν αρνητικά στα κυκλώματα πολύ μεγάλης κλίμακας ολοκλήρωσης (Very Large Scale Integration, VLSI) και οδηγούν σε διάφορες καταστάσεις αστοχίας. Μια αύξηση της θερμοκρασίας κατά 10° C διπλασιάζει το ποσοστό αστοχίας του κυκλώματος. Επίσης, η λειτουργία ενός ολοκληρωμένου κυκλώματος σε υψηλή θερμοκρασία μειώνει και τη διάρκεια ζωής του. Έτσι, λοιπόν, η μικρή κατανάλωση ισχύος δεν αφορά μό-

νον τις φορητές συσκευές, αλλά κάθε υπολογιστικό σύστημα, από το πιο απλό έως το πιο πολύπλοκο.

Οι δύο σημαντικότεροι παράγοντες στη διαμόρφωση της συνολικής κατανάλωσης ισχύος ενός συστήματος είναι η κατανάλωση ισχύος του επεξεργαστή και της μνήμης. Οι προσβάσεις στην εξωτερική μνήμη εκτός του ότι είναι αργές (όπως αναφέρθηκε προηγουμένως) έχουν και αρκετά υψηλό ενεργειακό τίμημα. Ο λόγος είναι απλός: Προκειμένου να αναγνωσθεί μια εξωτερική μνήμη πρέπει να γίνει πρόσβαση στην αρτηρία δεδομένων και διευθύνσεων της μνήμης· δηλαδή, πρέπει να μεταφερθούν σήματα από μέσα από το ολοκληρωμένο κύκλωμα προς τα έξω από αυτό και αντιστρόφως. Για να γίνει αυτό, πρέπει να χρησιμοποιηθούν ενισχυτές σημάτων. Έτσι, δε μας εκπλήσσει το γεγονός που υποστηρίζουν ερευνητές ότι, δηλαδή, η κατανάλωση ενέργειας της ιεραρχίας μνήμης αποτελεί το 60% - 80% της ολικής κατανάλωσης ισχύος ενός ενσωματωμένου συστήματος, που εκτελεί εφαρμογές με εντατική χρήση της εκτός ολοκληρωμένου κυκλώματος μνήμης.

Όπως έγινε φανερό από τις προηγούμενες ενότητες, υπάρχει ένα σημαντικό πρόβλημα στο σχεδιασμό ενσωματωμένων συστημάτων ως προς την επίδοση (ταχύτητα) και την κατανάλωση ενέργειας. Το πρόβλημα εστιάζεται από τη μια μεριά στον κυρίαρχο ανασχετικό παράγοντα της μνήμης του ενσωματωμένου συστήματος και από την άλλη μεριά στην ικανοποίηση των επεξεργαστικών απαιτήσεων των σύγχρονων εφαρμογών. Το πρόβλημα αυτό έχει γίνει αντιληπτό από τους ερευνητές εδώ και χρόνια και για αυτό το λόγο έχει παρουσιαστεί μια πληθώρα μεθοδολογιών, τεχνικών, ή τρόπων για την αποτελεσματική αντιμετώπισή του. Συγκεκριμένα, η μεγάλη καθυστέρηση και το υψηλό κόστος ενέργειας της εκτός ολοκληρωμένου κυκλώματος μνήμης, έχει οδηγήσει ερευνητές να προτείνουν διάφορους τρόπους για την καλύτερη εκμετάλλευση, είτε της μνήμης δεδομένων είτε της μνήμης εντολών, ενώ οι υψηλές επεξεργαστικές απαιτήσεις αντιμετωπίζονται κυρίως με τη χρήση εξειδικευμένων κυκλωμάτων, ως προς τις λειτουργίες της εφαρμογής. Όμως, όπως φαίνεται από τη σχετική βιβλιογραφία, οι ερευνητικές εργασίες αντιμετωπίζουν μερικώς το πρόβλημα και παρουσιάζουν ελλείψεις, ή δε λαμβάνουν υπόψη χαρακτηριστικά τα οποία θα μπορούσαν να χρησιμοποιηθούν για την περαιτέρω βελτίωση του συστήματος, ή παρουσιάζουν μια βέλτιστη λύση που δεν ικανοποιεί κάθε σχεδιαστή.

3.5.3 Το χάσμα ανάμεσα στον επεξεργαστή και στη μνήμη δεδομένων

Αν και η υπολογιστική ισχύς των σημερινών ενσωματωμένων πυρήνων είναι αρκετές εκατομμύρια εντολές το δευτερόλεπτο (millions instructions per second, MIPS), επιτρέποντας την εκτέλεση πολύπλοκων εφαρμογών με υψη-

λές επεξεργαστικές απαιτήσεις, εντούτοις η ιεραρχία μνήμης αποτελεί ανασχετικό παράγοντα στην επίτευξη υψηλών επιδόσεων, λόγω του μεγάλου χρόνου προσπέλασης σε αυτήν. Έχει αναφερθεί ότι ακόμα και οι σημερινοί επεξεργαστές ψηφιακών σημάτων εμφανίζονται αναποτελεσματικοί στην εκτέλεση συγκεκριμένων λειτουργιών των εφαρμογών MPEG. Επιπρόσθετα, έχει βρεθεί ότι το υψηλό ενεργειακό κόστος ανά πρόσβαση στη μνήμη είναι υπεύθυνο για τη μεγάλη κατανάλωση ισχύος σε εφαρμογές που κυριαρχούνται από μεταφορές δεδομένων από και προς την παρασκηνιακή μνήμη.

Το σημαντικό πρόβλημα του χάσματος επιδόσεων ανάμεσα στον επεξεργαστή και στη μνήμη έχει οδηγήσει αρκετούς ερευνητές στην προσπάθεια ανάπτυξης μεθοδολογιών για να το γεφυρώσουν. Μια προφανής λύση στο πρόβλημά μας θα μπορούσε να είναι η χρήση μνημών με πολλές θύρες. Όμως, μια τέτοια οργάνωση μνήμης αυξάνει σε απαγορευτικά όρια την ολική κατανάλωση ενέργειας, και έτσι η λύση αυτή δεν είναι πρακτική. Οι μνήμες με πολλές θύρες εισόδου / εξόδου έχουν μεγάλο κόστος επιφάνειας και κατανάλωσης ενέργειας, και συνήθως τις βρίσκουμε σε πολύ μικρά μεγέθη. Βέβαια, υπάρχουν μερικές περιπτώσεις στις οποίες είναι αναγκαία η χρήση μνημών με πολλές θύρες, επειδή διαφορετικά δε θα ικανοποιούνταν οι απαιτήσεις επεξεργασίας πραγματικού χρόνου.

Από την άλλη μεριά, προκειμένου να αντιμετωπιστούν οι προκλήσεις ως προς την κατανάλωση ενέργειας και επίδοσης των ενσωματωμένων συστημάτων, ερευνητές έχουν προτείνει τη χρήση πολυεπίπεδων αρχιτεκτονικών μνήμης. Γι' αυτό το λόγο, σχεδόν κάθε πλατφόρμα ενσωματωμένου συστήματος, σήμερα, έχει στην ιεραρχία μνήμης δεδομένων περισσότερα από ένα επίπεδα. Η βελτίωση των χαρακτηριστικών, που επιφέρει η χρήση πολυεπίπεδων αρχιτεκτονικών, βασίζεται στο γεγονός ότι τόσο η καταναλισκόμενη ενέργεια όσο και η επίδοση εξαρτώνται από το μέγεθος της μνήμης. Έτσι, με την εκμετάλλευση της ιεραρχίας της μνήμης, μπορούν να αποκομιστούν αρκετά οφέλη με τη μετατόπιση ενός αριθμού προσβάσεων από την παρασκηνιακή μνήμη στη μνήμη εντός ολοκληρωμένου κυκλώματος. Ασφαλώς, προκειμένου να γίνει καλύτερη εκμετάλλευση της μνήμης, πρέπει να ληφθούν υπόψη οι βελτιστοποιήσεις που μπορούν να γίνουν επί-τόπου (in-place optimizations). Αν και μελλοντικά θα αναλύσουμε τον όρο αυτό, σε αυτό το σημείο θα αναφέρουμε ότι *βελτιστοποιήσεις επί-τόπου* σημαίνει ότι χρησιμοποιούμε την ίδια περιοχή μνήμης για δύο ή περισσότερους πίνακες που δεν έχουν επικαλυπτόμενη διάρκεια ζωής. Στις τρέχουσες υλοποιήσεις η χρήση μνήμης εντός του ολοκληρωμένου κυκλώματος βασίζεται είτε σε υλικούς ελεγκτές που δημιουργούν αντίγραφα από το ένα επίπεδο στο άλλο, βασιζόμενοι σε τοπικά χαρακτηριστικά, είτε σε αλγοριθμικούς μετασχηματισμούς που εκμεταλλεύονται τα χαρακτηριστικά επαναχρησιμοποίησης δεδομένων. Συγκεκριμένα, για τη δεύτερη κατηγορία, οι μετασχηματισμοί εισάγουν αντίγραφα δεδομένων από απομακρυσμένα επίπεδα σε επί-

πεδα κοντά στον επεξεργαστή. Οι τεχνικές που έχουν προταθεί μέχρι σήμερα για τους μετασχηματισμούς χρησιμοποιούν μόνο ένα μικρό μέρος των βελτιστοποιήσεων που μπορούν να επιτευχθούν και, μάλιστα, σε μερικές προσεγγίσεις κάποιες βελτιστοποιήσεις αγνοούνται τελείως. Η ιεραρχία μνήμης μπορεί να εμπεριέχει πρόχειρες (ελεγχόμενες από το πρόγραμμα) μνήμες, ή σκιώδεις μνήμες. Προκειμένου μια εφαρμογή να απεικονιστεί αποτελεσματικά, θα πρέπει να γίνει διεξοδική διερεύνηση διαφορετικών υλοποιήσεων, κάτι το οποίον είναι αρκετά χρονοβόρο, εάν γίνει χειρωνακτικά.

Όσον αφορά τις κρυφές μνήμες, κάποιες εργασίες προτείνουν τη βελτίωση των παραμέτρων των σκιωδών μνημών με αύξηση του μεγέθους ή με ρυθμίσεις του μεγέθους μπλοκ, ή με την εύρεση του καταλληλότερου βαθμού συσχετικότητας αυτών των μνημών. Επίσης, ερευνητές έχουν προτείνει τη χρήση ελεγχόμενης από το πρόγραμμα προμετάκλησης, ή τη χρήση προγραμμάτων βελτιστοποιημένων για συγκεκριμένες σκιώδεις μνήμες. Τέλος, άλλες προσεγγίσεις χρησιμοποιούν πολυνηματικές (multithreading) τεχνικές, εκτέλεση εκτός σειράς και χρήση ειδικών τρόπων λειτουργίας μνήμης. Αν και το πρόβλημα έχει σε ένα μικρό βαθμό αντιμετωπιστεί με αυτές τις τεχνικές, εντούτοις το χάσμα επιδόσεων ανάμεσα στον επεξεργαστή και στη μνήμη παραμένει.

3.5.4 Δομοστοιχεία εκτός ολοκληρωμένου κυκλώματος

Τα στοιχεία εκτός ολοκληρωμένου κυκλώματος είναι η εξωτερική μνήμη και η μνήμη εντολών. Επιλέξαμε μια εξωτερική μνήμη, επειδή τα υπολογιστικά συστήματα σήμερα διεκπεραιώνουν πολύπλοκες εφαρμογές, τα δεδομένα των οποίων δεν μπορούν να αποθηκευτούν στη μνήμη εντός ολοκληρωμένου κυκλώματος. Συνεπώς, είναι αναγκαίο να υπάρχει μια μεγάλη εξωτερική μνήμη για να διατηρεί τα πολυδιάστατα σήματα. Στην πραγματικότητα, η εξωτερική μνήμη δεδομένων μπορεί να αποτελείται από μονάδες διαφορετικών τύπων και χαρακτηριστικών. Για παράδειγμα, η εξωτερική μνήμη μπορεί να έχει ένα επίπεδο στο οποίο βρίσκεται η παρασκηνιακή μνήμη τύπου RAM και τεχνολογίας DRAM, και ένα ακόμη επίπεδο στο οποίο βρίσκεται ένας σκληρός δίσκος (Σχήμα 3.28). Όπως θα δούμε στην επόμενη ενότητα, και η μνήμη δεδομένων εντός ολοκληρωμένου κυκλώματος χρησιμοποιεί μια παρόμοια ιεραρχία. Έτσι, λοιπόν, το σύστημά μας έχει μια πολυεπίπεδη ιεραρχία μνήμης δεδομένων. Η παρασκηνιακή μνήμη επιλέγεται συνήθως να είναι τεχνολογίας DRAM, λόγω του χαμηλού κόστους που έχει συγκρινόμενη με τη μνήμη SRAM. Η μεθοδολογία μας επιτρέπει τη χρήση ποικίλων DRAM, αρκεί ο σχεδιαστής να μπορεί να προσδιορίσει τα χαρακτηριστικά τους (κύκλοι πρόσβασης, κατανάλωση ισχύος, κ.α.), όπως SDRAM - synchronous DRAM, RDRAM - rambus DRAM, EDO DRAM - extended data out RAM κ.α. Αναλόγως της μνήμης που θα επιλεγεί, θα πρέπει να ληφθούν υπόψη και οι τρόποι με τους οποίους αυτή μπορεί να χρησιμοποιηθεί.

Για παράδειγμα, οι σύγχρονες μνήμες DRAM υποστηρίζουν ρυθμό πρόσβασης ριπής (burst), ή συστοιχίας (bank) κ.α., χαρακτηριστικά τα οποία βελτιώνουν τις επιδόσεις και την κατανάλωση ισχύος, αν χρησιμοποιηθούν σωστά. Ο σχεδιαστής απλώς πρέπει να ορίσει τα χαρακτηριστικά αυτά σε αρχεία ρυθμίσεων, τα οποία θα περιγράψουμε σε επόμενο κεφάλαιο.

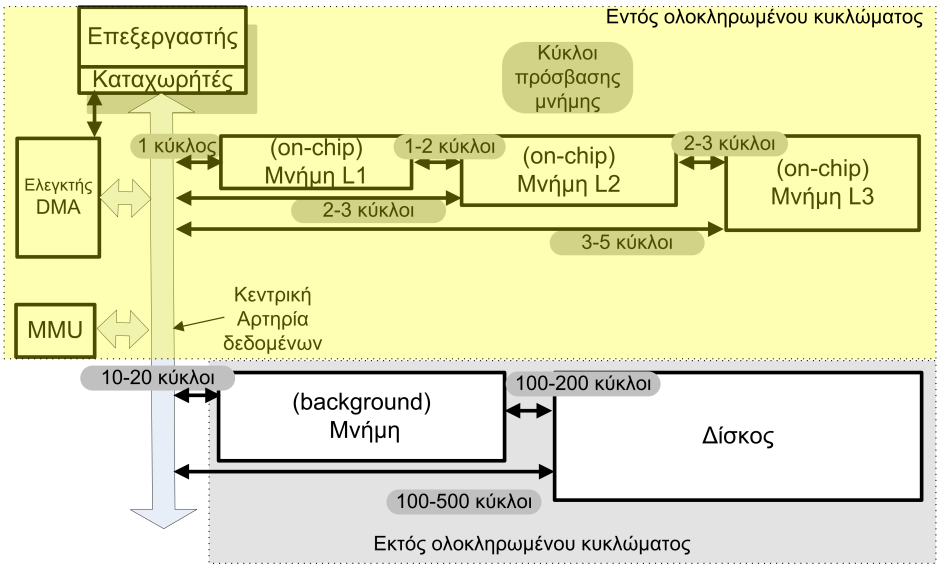
Το γεγονός ότι η εξωτερική μνήμη βρίσκεται εκτός ολοκληρωμένου κυκλώματος συνεπάγεται ότι οι προσβάσεις στη μνήμη αυτή αποτελούν τον κυρίαρχο παράγοντα διαμόρφωσης της ολικής κατανάλωσης ισχύος και επίδοσης [24]. Το πρόβλημα αυτό το αντιμετωπίζουμε επιτυχώς με μια ιεραρχία μνήμης πάνω στο ολοκληρωμένο κύκλωμα, όπως θα αναλυθεί σύντομα.

Κάθε προγραμματιζόμενος επεξεργαστής συνοδεύεται από μια μνήμη εντολών (Instruction Memory), η οποία είναι τύπου μόνο ανάγνωσης (ROM-Read Only Memory και έχει το λογισμικό του επεξεργαστή. Η μνήμη εντολών συνήθως τοποθετείται εκτός ολοκληρωμένου κυκλώματος, ώστε να είναι δυνατή η εύκολη αντικατάστασή της με μια αναβαθμισμένη έκδοση στην περίπτωση που αυτό χρειαστεί. Η αντικατάσταση γίνεται εύκολα με απομάκρυνση της μνήμης και τοποθέτηση μιας άλλης μνήμης που έχει προγραμματιστεί με την καινούργια έκδοση του προγράμματος. Τα δεδομένα που μεταφέρει η μνήμη αυτή δε χάνονται ποτέ. Αν και θα μπορούσε να χρησιμοποιηθεί επανεγγράψιμη ROM (π.χ. Electrically Erasable Programmable ROM, EEPROM), για λόγους κόστους, αξιοπιστίας και ταχύτητας πρόσβασης της μνήμης, τα πραγματικά ενσωματωμένα συστήματα χρησιμοποιούν μια μνήμη ROM, η οποία προγραμματίζεται μια μόνο φορά και συνήθως δεν αντικαθίσταται ποτέ.

Επειδή η μνήμη εντολών είναι εξωτερική συνεπάγεται μια καθυστέρηση στην πρόσβαση της μνήμης από τον επεξεργαστή, αφού ως γνωστόν η πρόσβαση δομοστοιχείων εκτός ολοκληρωμένου κυκλώματος είναι αρκετά αργή και τουλάχιστον μια τάξη μεγέθους βραδύτερη από προσβάσεις σε στοιχεία εντός ολοκληρωμένου κυκλώματος. Όπως θα δούμε στην επόμενη ενότητα, αντιμετωπίζουμε το πρόβλημα αυτό με μια σκιώδη μνήμη εντολών.

3.5.5 Δομοστοιχεία εντός ολοκληρωμένου κυκλώματος

Ένα από τα στοιχεία της αρχιτεκτονικής πάνω στο ολοκληρωμένο κύκλωμα είναι η μνήμη δεδομένων. Η μνήμη αυτή τις περισσότερες φορές είναι τύπου SRAM λόγω των μεγάλων ταχυτήτων πρόσβασης που προσφέρει, με τίμημα το κόστος κατασκευής της, αν και οι ερευνητές έχουν αρχίσει να τοποθετούν πάνω στο ολοκληρωμένο κύκλωμα μνήμη τύπου DRAM [25], [26]. Η μνήμη δεδομένων μπορεί να είναι είτε μια σκιώδης μνήμη, είτε μια πρόχειρη (scratchpad) μνήμη. Η σκιώδης μνήμη αποτελεί τη διασύνδεση ανάμεσα στον επεξεργαστή και στην παρασκηνιακή μνήμη, δεν έχει δικό της χώρο διευθύνσεων και δεν είναι άμεσα ελέγξιμη από το σχεδιαστή. Στην περίπτωση που χρησιμοποιείται



Σχήμα 3.28: Η αρχιτεκτονική μας χρησιμοποιεί μια πολυεπίπεδη ιεραρχία μνήμης.

σκιώδης μνήμη, όλες οι προσβάσεις στην παρασκηνιακή μνήμη γίνονται διαμέσου της σκιώδους μνήμης, αντικαθιστώντας δεδομένα που βρίσκονται μέσα σ' αυτήν. Έτσι, τα δεδομένα της σκιώδους μνήμης αλλάζουν δυναμικά κατά τη λειτουργία του συστήματος. Η λειτουργία της σκιώδους μνήμης δεδομένων είναι αρκετά πολύπλοκη και, έτσι, προτείνουμε να μην τη χρησιμοποιείται σε εφαρμογές χαμηλής κατανάλωσης ενέργειας.

Η πρόχειρη μνήμη, αντιθέτως, μπορεί να ελεγχθεί άμεσα από το σχεδιαστή. Χρησιμοποιεί συγκεκριμένες διευθύνσεις μνήμης, που σημαίνει ότι ο σχεδιαστής μπορεί ρητά να τοποθετήσει δεδομένα σ' αυτήν τη μνήμη, αποθηκεύοντας τα σε κάποια διεύθυνση μνήμης που αντιστοιχεί σ' αυτή την πρόχειρη μνήμη. Αν και η σκιώδης μνήμη και η πρόχειρη μνήμη είναι ίδιας τεχνολογίας (SRAM), εντούτοις η πρόχειρη μνήμη εγγυάται πάντα χρόνο πρόσβασης ενός κύκλου, που δεν μπορεί να εγγυηθεί η σκιώδης μνήμη, αφού μπορεί να υπάρξει αστοχία. Συνήθως κατά τη διάρκεια της μεταγλώττισης ο σχεδιαστής ορίζει ρητά τα δεδομένα που θα τοποθετηθούν πάνω στην πρόχειρη μνήμη.

Πάντως ανεξαρτήτως της μνήμης εντός ολοκληρωμένου κυκλώματος που θα επιλέξουμε, αυτή θα τοποθετηθεί στο σύστημα τηρώντας και διασφαλίζοντας την ιεραρχία μνήμης δεδομένων (Σχήμα 3.28). Η πιο μικρή και γρήγορη μνήμη θα βρίσκεται πολύ κοντά στον επεξεργαστή (μνήμη πάνω στο ολοκληρωμένο κύκλωμα L1, ή απλώς L1) και θα μπορεί να υποστηρίξει πρόσβαση ενός κύκλου· δηλαδή, μέσα σε ένα κύκλο ο επεξεργαστής θα μπορεί είτε να διαβάσει είτε να γράψει κάποιο στοιχείο σ' αυτήν. Στην ιεραρχία μνήμης μπορεί να

έχουμε και άλλα επίπεδα (π.χ. μνήμες εντός ολοκληρωμένου κυκλώματος L2 και L3, ή απλώς L2, L3, με χρόνους πρόσβασης από τον επεξεργαστή ίσους με 2-3 κύκλους και 3-5 κύκλους, αντίστοιχα). Επίσης, η ιεραρχία μνήμης είναι πλήρως διασυνδεδεμένη, που σημαίνει ότι υπάρχουν συνδέσεις μεταξύ των γειτονικών επιπέδων, ώστε σε περίπτωση που απαιτείται να μεταφερθούν δεδομένα από γειτονικά επίπεδα να μην υπάρχει επιβάρυνση της κεντρικής αρτηρίας δεδομένων.

Πριν συνεχίσουμε την περιγραφή της επιλεγμένης αρχιτεκτονικής θα εξηγήσουμε γιατί χρησιμοποιούμε την πολυεπίπεδη ιεραρχία μνήμης δεδομένων. Ο λόγος που το κάνουμε είναι για να βελτιστοποιήσουμε την επίδοση και την κατανάλωση ισχύος του συστήματος. Έτσι, η μνήμη διαιρείται σε επίπεδα διάφορων μεγεθών. Η επίδοση και η κατανάλωση ισχύος μιας μνήμης εξαρτώνται από το μέγεθος της μνήμης. Όσο πιο μικρό μέγεθος έχει η μνήμη τόσο πιο υψηλή είναι η επίδοση και πιο μικρή η κατανάλωση ισχύος. Όσο πιο συχνά χρησιμοποιούνται κάποια δεδομένα, τόσο πιο κοντά τοποθετούνται στον επεξεργαστή και αντίστροφα. Είναι σαφές ότι η επιλογή των δεδομένων που θα τοποθετηθούν στα διάφορα επίπεδα είναι μια κρίσιμη απόφαση και επηρεάζει κάθε παράμετρο του συστήματος. Η συστηματική μεθοδολογία που παρουσιάζουμε λύνει με επιτυχία αυτό το πρόβλημα.

Ένα ακόμη δομοστοιχείο, που υπάρχει εντός ολοκληρωμένου κυκλώματος, είναι η μονάδα διαχείρισης μνήμης. Η μονάδα αυτή χρησιμοποιείται καταρχήν για να κάνει αναζωογόνηση (refresh) των δεδομένων της μνήμης, γιατί διαφορετικά αυτά θα εξασθενήσουν και θα χαθούν. Επίσης, στην περίπτωση που υπάρχουν πολλαπλοί επεξεργαστές, χρησιμοποιείται για την κοινή και συνεπή χρήση της μνήμης, καθώς και του ποιος επεξεργαστής έχει πρόσβαση σε κάποιο επίπεδο της μνήμης. Τέλος, χρησιμοποιείται για τη μετατροπή των λογικών (εικονικών) διευθύνσεων μνήμης του επεξεργαστή σε πραγματικές διευθύνσεις της μνήμης, όταν χρησιμοποιείται διευθυνσιοδότηση απεικόνισης μνήμης (memory mapped). Ως γνωστόν, όλη η ιεραρχία μνήμης δεδομένων διευθυνσιοδοτείται από ένα συνεχές πεδίο εικονικών (virtual) διευθύνσεων μνήμης (π.χ. 0-64 MB). Στις διευθύνσεις αυτές αντιστοιχούν πραγματικές διευθύνσεις εσωτερικών και εξωτερικών μνημών. Έτσι, για παράδειγμα οι διευθύνσεις 0-1 KB αντιστοιχούν στην L1, οι διευθύνσεις 1 KB - 3 KB αντιστοιχούν στην L2 κ.ο.κ. Η μονάδα MMU είναι υπεύθυνη για τη μετατροπή των εικονικών διευθύνσεων στις πραγματικές διευθύνσεις της μνήμης. Οι εικονικές διευθύνσεις 1-3 KB αντιστοιχούν σε πραγματικές διευθύνσεις X1-X2 KB, τα οποία εξαρτώνται από τη μνήμη κ.ο.κ. Η μονάδα αυτή συναντάται σήμερα σχεδόν σε κάθε ενσωματωμένο σύστημα.

Μια μονάδα, που επίσης χρησιμοποιείται στη μεταφορά δεδομένων, είναι η μονάδα απευθείας πρόσβασης στη μνήμη (DMA-Direct Memory Access). Ως γνωστόν, στα περισσότερα ενσωματωμένα συστήματα οι μεταφορές δεδομένων από (ή σε) μνήμη εκτός ολοκληρωμένου κυκλώματος μπορούν να επιτευχθούν με

διάφορους τρόπους. Ο ευκολότερος και αρκετά διαδεδομένος (αλλά μη αποδοτικός) είναι όταν ο επεξεργαστής εποπτεύει, ελέγχει και διενεργεί τη μεταφορά. Το μειονέκτημα είναι ότι κατά τη διάρκεια που γίνεται η μεταφορά μ' αυτόν τον τρόπο, ο επεξεργαστής περιμένει και δεν εκτελεί κανέναν υπολογισμό, μέχρι να ολοκληρωθεί η μεταφορά δεδομένων, η οποία συνήθως απαιτεί αρκετούς κύκλους ρολογιού. Ένας εναλλακτικός τρόπος μεταφοράς είναι η χρήση του ειδικού τρόπου λειτουργίας 'Απευθείας Πρόσβασης στη Μνήμη' (Direct Memory Access-DMA). Όταν χρησιμοποιείται αυτός ο τρόπος, ο επεξεργαστής δεν εποπτεύει τη μεταφορά δεδομένων, αλλά αυτό ανατίθεται σε έναν ελεγκτή ή μηχανή που ονομάζεται ελεγκτής ή μηχανή DMA (DMA Controller). Η χρήση του ελεγκτή DMA συνεπάγεται ότι ο επεξεργαστής μπορεί να χρησιμοποιηθεί για την εκτέλεση άλλων λειτουργιών, εφόσον δεν υπάρχουν εξαρτήσεις δεδομένων. Συνήθως, τα ενσωματωμένα συστήματα έχουν παραπάνω από ένα κανάλι (ή προτεραιότητα) για να διενεργούν τις μεταφορές DMA. Η αρχιτεκτονική μας, λοιπόν, έχει έναν ελεγκτή DMA, ο οποίος υποστηρίζει περισσότερα από ένα κανάλια. Συνήθως, οι ελεγκτές DMA έχουν 4 κανάλια, όπως ο επεξεργαστής TI C6201 [27].

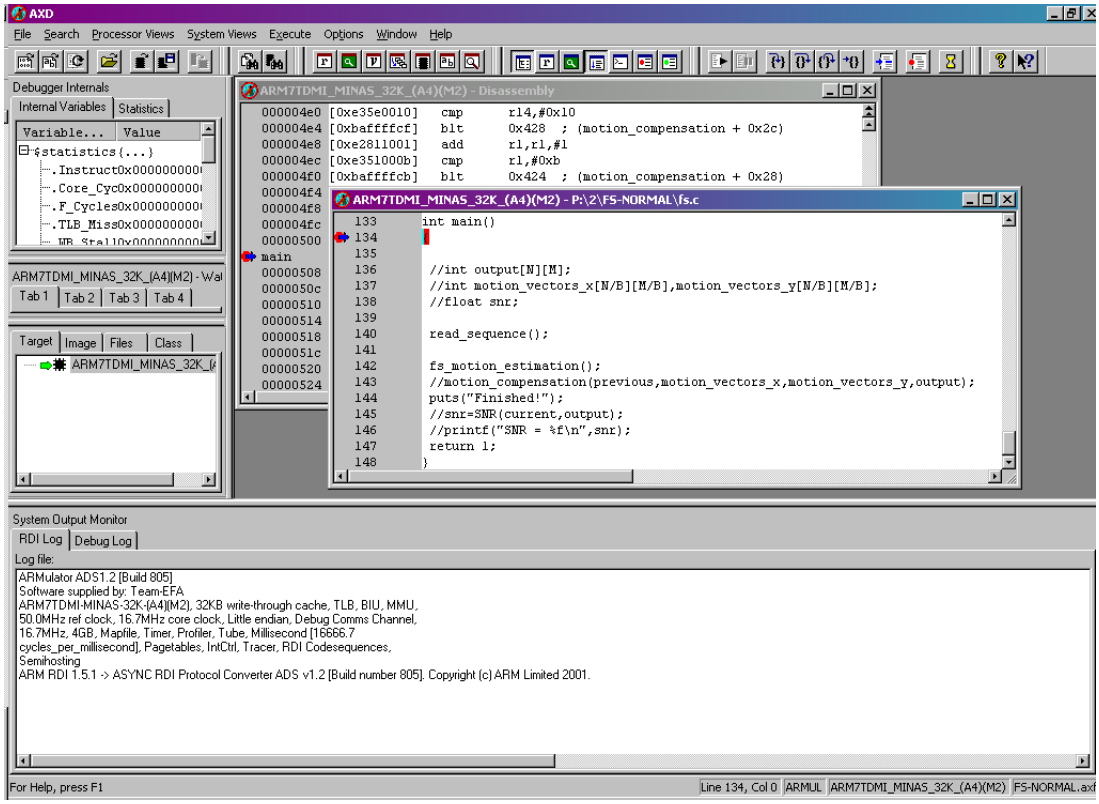
Η 'καρδιά' του ενσωματωμένου συστήματος είναι ο επεξεργαστής. Αν και η αρχιτεκτονική μας μπορεί εύκολα να επεκταθεί και σε πολλαπλούς επεξεργαστές, εμείς έχουμε διεξάγει τα πειράματά μας με έναν προγραμματιζόμενο επεξεργαστή. Η αξιολόγηση της συγκεκριμένης μεθοδολογίας έγινε χρησιμοποιώντας επεξεργαστές από τις εταιρείες Advanced RISC Machines (ARM) Ltd και Texas Instruments, για δύο λόγους. Πρώτον, οι δύο αυτές εταιρείες έχουν αθροιστικά το μεγαλύτερο μερίδιο στην αγορά των ενσωματωμένων συστημάτων και δεύτερον είχαμε στη διάθεση μας συμβολομεταφραστές και προσομοιωτές επιπέδου εντολών για τους επεξεργαστές τους. Πρέπει να τονίσουμε ότι η αρχιτεκτονική μας είναι αρκετά γενική, ώστε να μπορεί να χρησιμοποιηθεί οποιοσδήποτε επεξεργαστής, αρκεί να υπάρχει στη διάθεση του σχεδιαστή ο αντίστοιχος προσομοιωτής.

Ειδικότερα, από την εταιρεία ARM Ltd επιλέξαμε τον επεξεργαστή ARM7TDMI της οικογένειας ARM7 [28]. Η επιλογή μας βασίστηκε στο γεγονός ότι οι επεξεργαστές της οικογένειας ARM7 είναι αρκετά διαδεδομένοι στην αγορά των φορητών ενσωματωμένων συστημάτων, γιατί έχουν τον καλύτερο λόγο επίδοσης προς κατανάλωση ισχύος (MIPS/Watt) μαζί με τον καλύτερο λόγο επίδοσης προς κόστος (MIPS/$). Όσον αφορά τις τεχνικές λεπτομέρειες, ο επεξεργαστής αυτός είναι ένας επεξεργαστής τύπου μειωμένου συνόλου εντολών (Reduced Instruction Set Computer - RISC), δηλαδή ενός απλοποιημένου συνόλου εντολών με μέγεθος λέξης 32 bit και με 16 καταχωρητές χρήστη 32 bit σε σύνολο 37 καταχωρητών. Ο επεξεργαστής αυτός υποστηρίζει διαδοχική διοχέτευση τριών βαθμίδων και είναι αρχιτεκτονικής Von Neuman· δηλαδή, αρχιτεκτονικής με διαφορετική αρτηρία δεδομένων και διαφορετική αρτηρία

εντολών. Η λέξη TDMI που βρίσκεται μετά το όνομα ARM7 είναι το ακρωνύμιο των λέξεων Thumb instruction set with Debug, Multiplier and Interrupt· δηλαδή, ότι έχει ένα σετ εντολών τύπου Thumb, υποστηρίζει εύκολη εκσφαλμάτωση, με ικανότητα εκτέλεσης γρήγορων πολλαπλασιασμών, και υποστηρίζει σημαίες διακοπτών. Η επέκταση Thumb σημαίνει ότι το σετ εντολών του επεξεργαστή έχει συμπιεστεί από 32 bit σε 16 bit, με συνέπεια να υπάρχει έως και 40% μείωση στο μέγεθος της μνήμης εντολών που χρησιμοποιεί αυτός ο επεξεργαστής. Ασφαλώς, η συμπίεση των εντολών σημαίνει ότι θα πρέπει να υπάρχει μια μονάδα αποσυμπίεσης εντολών μέσα στον επεξεργαστή, ώστε η εντολή 16 bit στη μνήμη εντολών να αποσυμπιέζεται στην εντολή 32 bit μέσα στον επεξεργαστή. Τα δεδομένα βέβαια δε συμπιέζονται και παραμένουν με μήκος 32 bit. Ο χρόνος αποσυμπίεσης είναι αμελητέος, αφού γίνεται μέσα στον πυρήνα του επεξεργαστή. Η τυπική συχνότητα λειτουργίας του επεξεργαστή είναι 33 Mhz. Ο επεξεργαστής αυτός διευθυνσιοδοτεί λέξεις 32 bit, άρα μπορεί να χειριστεί μνήμη έως 4GB. Το μήκος λέξης δεδομένων που υποστηρίζει είναι 32, ή 16, ή 8 bit. Τέλος, ο ARM7TDMI υποστηρίζει την τεχνική φόρτωσης - αποθήκευσης (load-store) που σημαίνει ότι τα δεδομένα μπορεί να υποστούν επεξεργασία μόνον όταν έχουν μεταφερθεί σε καταχωρητές και τοποθετούνται μόνο σε καταχωρητές (και στη συνέχεια στη μνήμη). Δηλαδή, δεν υπάρχει η δυνατότητα επεξεργασίας δεδομένων απευθείας στη μνήμη.

Προκειμένου να αξιολογήσουμε τον επεξεργαστή ARM7TDMI χρησιμοποιήσαμε το αναπτυξιακό πρόγραμμα ARM Software Development Toolkit [29] (Σχήμα 3.29). Το εργαλείο αυτό αποτελείται από μια σειρά προγραμμάτων μεταγλώττισης, εκσφαλμάτωσης (debugging) και προσομοίωσης προκειμένου να αξιολογηθεί η επίδοση μιας εφαρμογής σε μια αρχιτεκτονική με κάποιον επεξεργαστή ARM. Από όλους τους διαθέσιμους επεξεργαστές εμείς προτιμήσαμε τον αρκετά διαδεδομένο επεξεργαστή ARM7TDMI. Ο χρήστης μπορεί να δηλώσει μια ιεραρχία μνήμης, τους χρόνους πρόσβασης που απαιτούνται σε κάθε επίπεδό της και να μετρήσει τους κύκλους ρολογιού που απαιτήθηκαν από την εφαρμογή, καθώς και την κατανομή τους στις διάφορες συναρτήσεις της εφαρμογής. Η προσομοίωση γίνεται σε επίπεδο εντολών, οπότε η επίδοση που μετράται συμπίπτει με την πραγματική επίδοση χωρίς προσομοίωση. Το πρόγραμμα αυτό υποστηρίζει πρόχειρες μνήμες καθώς και σκιώδεις μνήμες.

Εκτός από τον επεξεργαστή της εταιρείας Advanced RISC Machines, επιλέξαμε έναν ακόμη αρκετά διαδεδομένο επεξεργαστή ενσωματωμένων συστημάτων, τον TI C6201 [27]. Οι λόγοι που τον επιλέξαμε σχετίζονται αρχικά με το γεγονός ότι για ένα μικρό χρονικό διάστημα υπήρχε στη διάθεσή μας μια αναπτυξιακή πλακέτα με δύο επεξεργαστές TI C6201, για την καλύτερη αξιολόγηση της μεθοδολογίας, και είχαμε στη διάθεση μας το αναπτυξιακό πρόγραμμα για αυτόν τον επεξεργαστή. Όσον αφορά τις τεχνικές πληροφορίες, ο επεξεργαστής αυτός χρονίζεται στα 200 Mhz, έχει 5 ns μέγιστο χρόνο εκτέλεσης εντολής, εί-

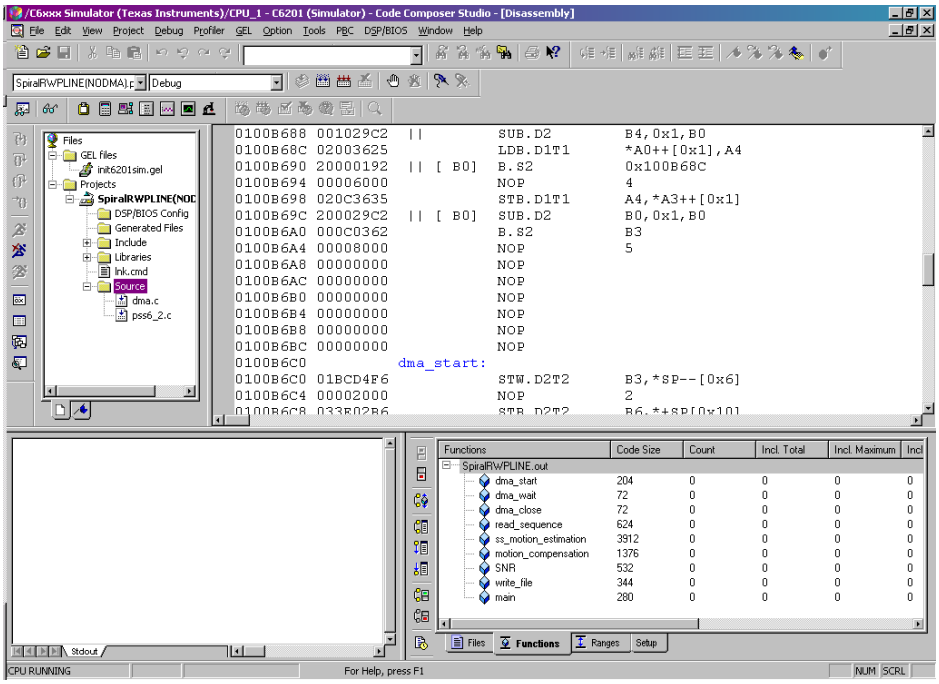


Σχήμα 3.29: Η αξιολόγηση της μεθοδολογίας μας χρησιμοποιώντας τον επεξεργαστή ARM7TDMI έγινε με τη βοήθεια του προγράμματος ARM Software Development Toolkit

ναι επεξεργαστής 32 bit, μπορεί να δεχτεί έως και 8 εντολές ανά κύκλο και έχει 1500 MIPS. Μάλιστα, έχει 6 λειτουργικές μονάδες (4 μονάδες αριθμητικής λογικής (ALU-Arithmetical Logical Unit) και 2 μονάδες πολλαπλασιασμού) και είναι αρχιτεκτονικής φόρτωσης-αποθήκευσης, όπως και ο ARM7TDMI, που σημαίνει ότι τα δεδομένα πρέπει να έρθουν σε κάποιον καταχωρητή προτού υποστούν επεξεργασία, και δε γίνεται απευθείας επεξεργασία στη θέση μνήμης. Ο επεξεργαστής αυτός έχει πάνω στο ίδιο ολοκληρωμένο κύκλωμα και έναν ελεγκτή DMA με τέσσερα κανάλια. Υποστηρίζει δεδομένα 8, 16 και 32 bit. Έχει 32 καταχωρητές γενικής χρήσης. Ο επεξεργαστής αυτός είναι αρχιτεκτονικής πολύ-μεγάλης λέξης εντολής (Very-Long Instruction Words–VLIW), που σημαίνει ότι δέχεται μια πολύ μεγάλη εντολή (μέχρι 256 bit), η οποία συνθέτεται από εντολές 32 bit (μέχρι και 8 εντολές), οι οποίες τροφοδοτούνται στις λειτουργικές μονάδες. Δεν απαιτείται κάθε φορά να υπάρχουν και οι 8 εντολές, αλλά ο επεξεργαστής δέχεται και λιγότερες εντολές ανά κύκλο. Όπως και πριν, ο επεξεργαστής αυτός διευθυνσιοδοτεί λέξεις 32 bit, άρα μπορεί να χειριστεί μνήμη έως 4 GB. Τέ-

λος, ο επεξεργαστής διαθέτει λογική μείωσης της κατανάλωσης ισχύος (power down logic), που σημαίνει ότι μπορεί κάποια μέρη του επεξεργαστή να τίθενται σε κατάσταση μικρής κατανάλωσης ισχύος, όταν αυτά δε χρησιμοποιούνται.

Όσον αφορά την αξιολόγηση του επεξεργαστή TI C6201, χρησιμοποιήσαμε το αναπτυξιακό πρόγραμμα Texas Instrument Code Composer Studio v2.2 [30] (Σχήμα 3.30). Το εργαλείο αυτό είναι παρόμοιο με το εργαλείο της εταιρείας ARM, αφού είναι ένα πλήρες εργαλείο μετάφρασης και προσομοίωσης εφαρμογών για επεξεργαστές της σειράς 6000 της εταιρείας TI. Ο χρήστης μπορεί να ορίσει τις διευθύνσεις μνήμης εντός και εκτός ολοκληρωμένου κυκλώματος, αλλά δεν μπορεί να καθορίσει ακριβώς τους χρόνους πρόσβασης, αφού είναι προκαθορισμένοι από την εταιρεία. Υποστηρίζονται μόνο δύο επίπεδα μνήμης (εντός και εκτός ολοκληρωμένου κυκλώματος). Το θετικό στοιχείο είναι ότι ο προσομοιωτής υποστηρίζει μεταφορές με τη χρήση του ελεγκτή DMA, αλλά ο χρήστης πρέπει να ορίσει με μια δύσκολη χειρωνακτική διαδικασία τις ακριβείς παραμέτρους. Η προσομοίωση γίνεται σε επίπεδο εντολών, συνεπώς είναι αρκετά ακριβής. Τέλος, ο χρόνος εκτέλεσης του προσομοιωτή για την ίδια εφαρμογή, συγκρινόμενος με τον προσομοιωτή του ARM7TDMI, είναι έως και 150 φορές μεγαλύτερος (5 λεπτά προσομοίωσης για μια εφαρμογή στον ARM7TDMI, αντιστοιχούν σε περίπου 12 ώρες προσομοίωση στον επεξεργαστή TI C6201).



Σχήμα 3.30: Προσομοιώνουμε τον επεξεργαστή TI C6201 με τη χρήση του προγράμματος Code Composer Studio v2.0.

Στα δομοστοιχεία εντός του ολοκληρωμένου κυκλώματος βρίσκονται η σκιώδης μνήμη εντολών και ο επιταχυντής υλικού IP-block accelerator. Η σκιώδης μνήμη εντολών είναι μια μνήμη τεχνολογίας SRAM, η οποία βοηθάει στη μείωση της καθυστέρησης πρόσβασης στην εκτός ολοκληρωμένου κυκλώματος μνήμη εντολών του επεξεργαστή. Η μνήμη αυτή μειώνει την κατανάλωση ισχύος και αυξάνει την επίδοση του συστήματος. Ανάλογα με την εφαρμογή (και τον επεξεργαστή), η μνήμη αυτή έχει διαφορετικά χαρακτηριστικά, τα οποία πρέπει να προσδιοριστούν επακριβώς.

Τέλος, το IP-block είναι ένα εξειδικευμένο κύκλωμα για συγκεκριμένες λειτουργίες. Έχουμε αναπτύξει εργαλεία που βοηθούν το σχεδιαστή στο σχεδιασμό των μονάδων IP-block σε συνθέσιμη γλώσσα περιγραφής υλικού ολοκληρωμένων κυκλωμάτων πολύ υψηλής ταχύτητας [VHSIC (Very High Speed Integrated Circuits) Hardware Description Language (VHDL)]. Στην περίπτωση που δεν καλύπτονται οι απαιτήσεις της επίδοσης του συστήματος, ο σχεδιαστής μπορεί να υλοποιήσει κάποιες λειτουργίες (συνήθως συναρτήσεις) του μικροεπεξεργαστή, με τις οποίες ο επεξεργαστής ασχολείται σε μεγάλο βαθμό. Για κάθε συνάρτηση, μπορούμε να εξάγουμε τις πληροφορίες κατατομής (profiling) χρησιμοποιώντας τα εργαλεία προσομοίωσης για κάποιον επεξεργαστή. Εκτός του γεγονότος ότι κάποιες λειτουργίες λογισμικού γίνονται πολύ πιο γρήγορα σε υλικό, η χρήση των IP-block απελευθερώνει και τον επεξεργαστή, ο οποίος έχει τη δυνατότητα να λειτουργήσει παράλληλα με το εξειδικευμένο κύκλωμα, που επιπρόσθετα αυξάνει τις επιδόσεις. Εκτός από τις βελτιωμένες επιδόσεις, η χρήση των IP-block μειώνει και την κατανάλωση ισχύος του κυκλώματος, αφού η εφαρμογή μας ολοκληρώνει τη λειτουργία πολύ πιο γρήγορα και αφού, αντί να ενεργοποιείται ένα πλήθος κυκλωμάτων στον επεξεργαστή, όπως είναι οι καταχωρητές, η μνήμη εντολών κτλ, ενεργοποιείται ένα πολύ μικρότερο κύκλωμα που αποτελείται από πλήρεις αθροιστές και πύλες.

3.6 Ανασκόπηση του Κεφαλαίου

Σ' αυτό το κεφάλαιο αναλύσαμε τις αρχιτεκτονικές που συναντούμε στα τυπικά ΕΣ. Αναλύθηκαν οι αρχιτεκτονικές των επεξεργαστών και οι μνήμες και έγινε μια εισαγωγή στη διαδικασία εισόδου εξόδου. Στη συνέχεια, περιγράψαμε διεξοδικά την αρχιτεκτονική που επιλέξαμε για να αναπτύξουμε και να αξιολογήσουμε τη μεθοδολογία σχεδιασμού ΕΣ, καθώς και τους λόγους που μας κατεύθυναν προς την επιλογή των δομοστοιχείων. Η αρχιτεκτονική-στόχος που παρουσιάσαμε, είναι μια γενική αρχιτεκτονική που από τη μια μεριά συναντάται σε μια πληθώρα ενσωματωμένων συστημάτων και από την άλλη μεριά παρέχει ευελιξία στο σχεδιαστή να την τροποποιήσει, ώστε να καλύπτονται καλύτερα οι σχεδιαστικές του απαιτήσεις.