Τ.Ε.Ι. ΗΠΕΙΡΟΥ
ΣΧΟΛΗ: ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ: ΤΗΛΕΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ : ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ

Επιβλέπων καθηγητής :Γλαβάς Ευριπίδης

Εισηγητές: Ρόκα Αφροδίτη
Σωτηράκου Σταματία

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Μια από τις περισσότερες ενδιαφέρουσες πτυχές της αρχιτεκτονικής υπολογιστών είναι το ποσοστό στο οποίο ο τομέας αλλάζει.Η καινοτομία εμφανίζεται σε σχεδόν-καθημερινή βάση, που προσφέρει τις ευκαιρίες για τα άτομα που συμβάλουν στον τομέα.Εντούτοις, αυτό το ποσοστό προόδου είναι μια από τις μέγιστες προκλήσεις της διδασκαλίας στην αρχιτεκτονική και την οργάνωση υπολογιστών.Αντίθετα σε πολλούς άλλους τομείς, οι σειρές μαθημάτων στην αρχιτεκτονική και την οργάνωση υπολογιστών πρέπει να αλλάξουν από όρο σε όρο βάσης για να ενσωματώνει τις νέες εξελίξεις στον τομέα χωρίς την υπερφορτώτωση των σπουδαστών με το υλικό.Το γράψιμο των εγχειριδίων για τον τομέα είναι ομοίως δύσκολο, δεδομένου ότι ο συντάκτης πρέπει να βρεί μια ισορροπία μεταξύ του συνδιασμού του υλικού και της ιστορικής πλευράς.

Αυτό το βιβλίο περιλαμβάνει μια επιλογή των θεμάτων που προορίζονται για να το καταστήσουν χρήσιμο στους αναγνώστες με ένα ευρύ φάσμα της προηγούμενης έκθεσης στον τομέα.Ένα μέρος των κεφαλαίων καλύπτουν πολλες από τις βασικές έννοιες στην οργάνωση των υπολογιστών, που περιλαμβάνει πώς η απόδοση μετριέται, πώς οι υπολογιστές αντιπροσωπεύουν τα αριθμητικά στοιχεία και τα προγράμματα, τα διαφορετικά πρότυπα προγραμματισμού για τους υπολογιστές, και τα βασικά του σχεδίου επεξεργαστών.Τα κεφαλαία 4 και 5 καλύπτουν την διοχέτευση και τις οδηγίες επιπέδου παραλιλισμού,δύο τεχνολογίες που είναι εξαιρετικά σημαντικές στην απόδοση των σύγχρονων επεξεργαστών.Τακεφάλαια 6,7 και 8 καλύπτουν τον σχεδιασμό των συστημάτων μνήμης,συμπεριλαμβανομένων των ιεραρχιών μνήμης, των κρυφών μνημών, και της εικονικής μνήμης.Το κεφάλαιο 9 περιγράφει τα I/O συστήματα, ενώ το κεφάλαιο 10 παρέχει μια εισαγωγή στα συστήματα πολυεπεξεργαστών - υπολογιστές που συνδυάζουν τους επεξεργαστές για να δώσουν βελτιωμένη απόδοση .

ΚΕΦΑΛΑΙΟ 1^{ο}

Η ΟΡΓΑΝΩΣΗ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

1.1 Στόχοι

Σε αυτό το κεφάλαιο αρχίζουμε την περιγραφή των βασικών μονάδων που αποτελούν τα συνήθη συστήματα υπολογιστών: Επεξεργαστές, μνήμη και σύστημα εισόδου/εξόδου (Input/Output). Επίσης θα περιγράψουμε σύντομα πώς τα προγράμματα λειτουργούν εσωτερικά στον υπολογιστή και πως τα λειτουργικά συστήματα μέσω των προγραμμάτων ελέγχουν τις επιμέρους φυσικές μονάδες που συνθέτουν τον υπολογιστή.

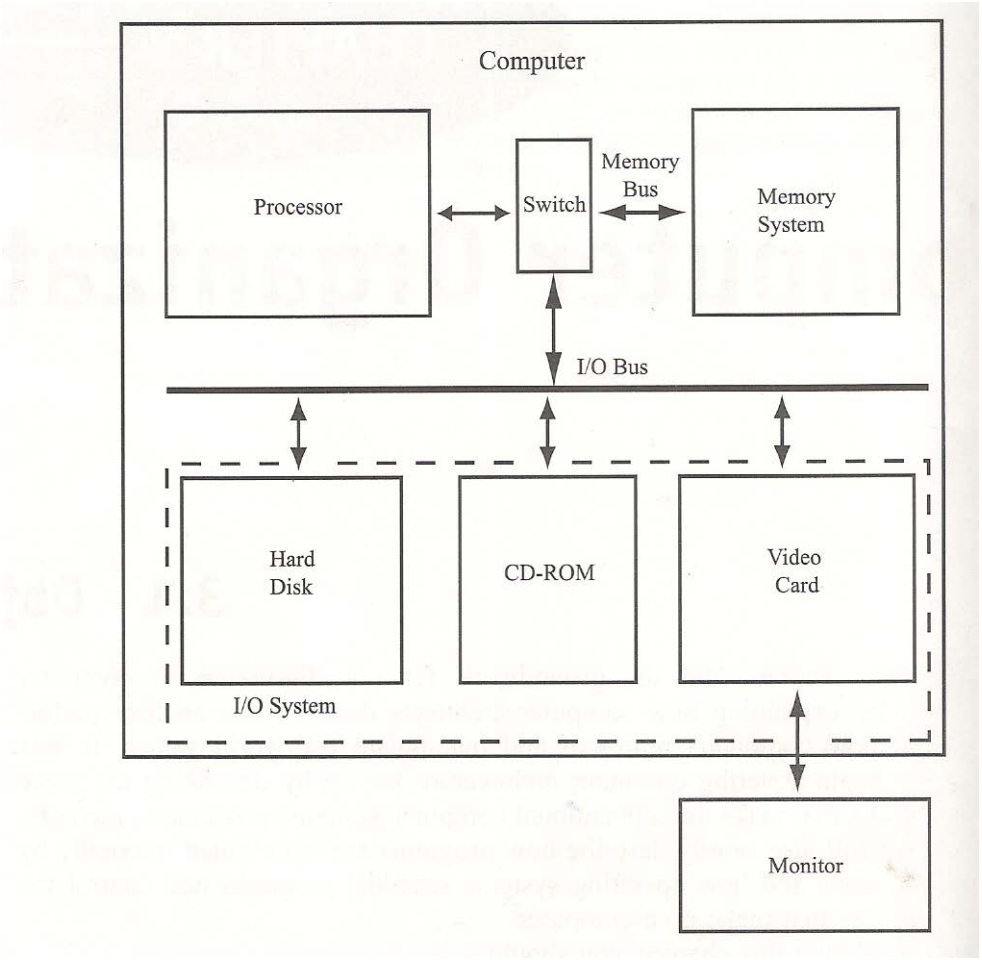
Ολοκληρώνοντας αυτό το κεφάλαιο θα έπρεπε:

1. Να καταλαβαίνετε τις βασικές έννοιες σχετικά με επεξεργαστές μνήμη και I/O συσκευές και να μπορείτε να περιγράψετε πώς λειτουργούν.
2. Να σας είναι οικεία η αρχιτεκτονική αποθηκευμένων προγραμμάτων.
3. Να καταλαβαίνετε τις βασικές λειτουργίες των λειτουργικών συστημάτων

1.2 Εισαγωγή

Όπως φαίνεται στο σχήμα 1-1τα περισσότερα συστήματα υπολογιστών μπορούν αν διαιρεθούν σε τρία υποσυστήματα. Τον επεξεργαστή(processor) τη μνήμη(memory system) και το σύστημα εισόδου εξόδου (I/O). Ο επεξεργαστής είναι υπεύθυνος για την εκτέλεση των προγραμμάτων, η μνήμη παρέχει αποθηκευτικό χώρο για τα προγράμματα κ τις πληροφορίες που διαχειρίζονται ενώ το I/O σύστημα επιτρέπει στον επεξεργαστή και τη μνήμη να ελέγχουν τις συσκευές που αλληλεπιδρούν με το έξω περιβάλλον ή με τις πληροφορίες που βρίσκονται σε CD ROM σκληρό δίσκο και κάρτα βίντεο και οθόνης, όπως φαίνεται και στο σχήμα.

Στα περισσότερα συστήματα ο επεξεργαστής έχει ένα δίαυλο που συνδέεται με ένα ηλεκτρικό διακόπτη, όπως η γέφυρα PCI που βρίσκεται σε πολλά συστήματα Η/Υ. Παρόλα αυτά, και για λόγους εξοικονόμησης χώρου, πολλοί επεξεργαστές έχουν τον ηλεκτρικό διακόπτη ολοκληρωμένο στο κύκλωμά τους. Με αυτό τον τρόπο γίνεται οικονομία και στο συνολικό κόστος του συστήματος.



ΣΧΗΜΑ1-1 computer organization

Ο διακόπτης επικοινωνεί με τη μνήμη διαμέσου ενός διαύλου μνήμης (memory bus) και ένα σετ καλωδίων που μεταφέρουν πληροφορίες ανάμεσα σε αυτά τα δύο συστήματα αποκλειστικά.

Ένας ξεχωριστός δίαυλος I/O συνδέει το διακόπτη με τις συσκευές εισόδου / εξόδου. Χρησιμοποιούνται διαφορετικοί δίαυλοι γιατί το σύστημα I/O είναι σχεδιασμένο έτσι ώστε να υποστηρίζει όσο το δυνατόν περισσότερες συσκευές ταυτόχρονα, ενώ η μνήμη είναι σχεδιασμένη έτσι ώστε να παρέχει το μεγαλύτερο δυνατό εύρος ζώνης ανάμεσα στα συστήματα μνήμης – επεξεργαστή.

1.3 Προγράμματα

Πρόγραμμα είναι μια ακολουθία οδηγιών/εντολών που υποδεικνύουν στον υπολογιστή τι να κάνει. Το πως βλέπει ο υπολογιστής τις οδηγίες διαφέρει κατά πολύ από το πώς τις βλέπει ο προγραμματιστής.

Για το μηχάνημα το πρόγραμμα είναι φτιαγμένο από ακολουθία αριθμών που αντιπροσωπεύουν ξεχωριστές λειτουργίες. Αυτές οι λειτουργίες είναι γνωστές ως οδηγίες, ενώ το σύνολο των λειτουργιών που κάθε επεξεργαστής καλείται να εκτελέσει είναι γνωστό ως σετ οδηγιών.

Σχεδόν όλοι οι υπολογιστές είναι υπολογιστές αποθηκευμένων προγραμμάτων. Δηλαδή αναπαριστούν τα προγράμματα ως αριθμούς που αποθηκεύουν στον ίδιο χώρο με τις πληροφορίες. Αυτού του είδους οι υπολογιστές ονομάζονται και Von Neumann, από το όνομα ενός από τους εμπνευστές της ιδέας αυτής, που ήταν και μια από τις πιο επαναστατικές στην ιστορία της αρχιτεκτονικής υπολογιστών. Νωρίτερα οι υπολογιστές προγραμματίζονταν με την τοποθέτηση διακοπτών ή πλακετών με κυκλώματα που καθόριζαν το νέο πρόγραμμα. Αυτή η διαδικασία ήταν χρονοβόρα και επιρρεπής σε λάθη.

Η αποθήκευση των προγραμμάτων όμως έχει δύο σημαντικά πλεονεκτήματα σε σχέση με τις προϋπάρχουσες αντιμετωπίσεις. Καταρχήν επιτρέπει στο πρόγραμμα να αποθηκευτεί εύκολα και να φορτωθεί στο μηχάνημα. Από τη στιγμή που ένα πρόγραμμα έχει αναπτυχθεί και διορθωθεί, οι αριθμοί που αναπαριστούν τις οδηγίες μπορούν να γραφούν σε μια συσκευή αποθήκευσης επιτρέποντας στο πρόγραμμα να φορτώνεται στη μνήμη κάποια στιγμή στο μέλλον. Στα πρώτα συστήματα οι πιο κοινές συσκευές αποθήκευσης ήταν οι διάτρητες κάρτες και οι χαρτοταινίες. Τα πιο σύγχρονα μέσα χρησιμοποιούν μαγνητικά μέσα όπως είναι οι σκληροί δίσκοι. Η δυνατότητα αποθήκευσης των προγραμμάτων ως πληροφορίες εξαλείφει τα λάθη στην επαναφόρτωση του προγράμματος , εφόσον ο χώρος που αποθηκεύει είναι απαλλαγμένος και αυτός από σφάλματα. Παλαιότερα που έπρεπε το πρόγραμμα να επανεισαχθεί κάθε φορά που θα χρησιμοποιούταν συχνά παρουσίαζε σφάλματα που αν δε διορθώνονταν το πρόγραμμα δεν έτρεχε σωστά.

Επιπρόσθετα η λογική των αποθηκευμένων προγραμμάτων επιτρέπει στα προγράμματα να αντιμετωπίζουν και τα ίδια αλλά και τα άλλα προγράμματα σαν πληροφορίες.Τα προγράμματα που μπορούν να αυτοεπεξεργάζονται λέγονται self modifying programs δηλαδή αυτοτροποιοποιησιμα προγράμματα.. Σε αυτού του είδους τα προγράμματα κάποιες από τις οδηγίες δημιουργούν άλλες μέσα στο πρόγραμμα. Στους πρώτους υπολογιστές τέτοια προγράμματα ήταν κοινά γιατί θεωρούνταν πιο γρήγορα από άλλα που δεν είχαν αυτή τη δυνατότητα αλλά και για ένα άλλο λόγο: Τα πρώτα προγράμματα χρησιμοποιούσαν ένα περιορισμένο αριθμό εντολών και έτσι ήταν πιο πρακτικό να μπορούν να επεξεργάζονται τα ίδια τον κώδικά τους. Στην πραγματικότητα οι προγραμματιστές με αυτό τον τρόπο αντικαθιστούσαν τις εντολές διακλαδώσεως όταν δε τους επέτρεπε το σετ οδηγιών να δημιουργήσουν μια διακλάδωση.

Ο αυτοτροποποιήσιμος κώδικας στις μέρες μας δεν είναι τόσο δημοφιλής, γιατί πλέον οποιαδήποτε αλλαγή στο πρόγραμμα κατά τη διάρκεια της εκτέλεσης κάνει πιο δύσκολη την εκσφαλμάτωση (debugging).Πλέον, αφού οι υπολογιστές έχουν γίνει ταχύτεροι, έχει μεγάλη σημασία οποιαδήποτε διευκόλυνση στην εκτέλεση και εκσφαλμάτωση του προγράμματος. Επιπλέον τα συστήματα μνήμης που χρησιμοποιούν λανθάνουσα (cach) μνήμη κάνουν τον αυτοτροποποιήσιμο κώδικα λιγότερο αποτελεσματικό και λειτουργικό.

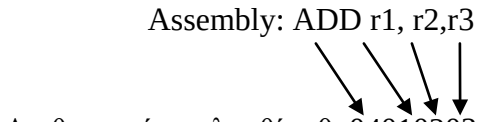
1.3.1. Εργαλεία ανάπτυξης προγραμμάτων

Τα προγράμματα που μπορούν να επεξεργαστούν άλλα προγράμματα σαν πληροφορίες είναι πιο κοινά, και πολλά από τα εργαλεία ανάπτυξης προγραμμάτων ανήκουν σε αυτή την κατηγορία. Συμπεριλαμβάνονται προγράμματα όπως οι μεταγλωττιστές (compilers), που μετατρέπουν γλώσσες όπως η C και η FORTRAN

σε γλώσσα μηχανής, και οι συμβολομεταφραστες (assemblers), που μετατρέπουν τη γλώσσα μηχανής σε αριθμητική ακολουθία που καταλαβαίνει ο επεξεργαστής, οι διασυνδετές (linkers) που συνδέουν σε ένα εκτελέσιμο αρχείο πολλά προγράμματα σε γλώσσα μηχανής. Ακόμα συμπεριλαμβάνονται οι εκσφαλματωτές (debuggers) προγράμματα που χρησιμοποιούνται για την εύρεση λαθών ενός άλλου προγράμματος και δίνουν στους προγραμματιστές στοιχεία για την τρέχουσα κατάστασή του και άλλα στοιχεία προόδου.

Οι πρώτοι υπολογιστές αποθηκευμένων προγραμμάτων προγραμματίζονταν απευθείας σε γλώσσα μηχανής, την αριθμητική ακολουθία που χρησιμοποιεί ο επεξεργαστής. Για να γράψει ένα πρόγραμμα ο προγραμματιστής έπρεπε να καθορίσει την σειρά με την οποία έπρεπε να δοθούν οι εντολές ώστε να εκτελεστούν σωστά και έπειτα εισήγαγε τους αριθμούς που αντιπροσώπευαν αυτές τις εντολές στον υπολογιστή. Ήταν μια χρονοβόρα διαδικασία που συχνά παρουσίαζε και πολλά λάθη.

Το πρώτο βήμα για την απλοποίηση του προγράμματος έγινε με την εμφάνιση των συμβολομεταφραστών (assemblers) που επέτρεπαν στους προγραμματιστές να γράφουν στη γλώσσα assembly.Σε αυτή τη γλώσσα κάθε εντολή μηχανής έχει και μια αναπαράσταση σε μορφή κειμένου (όπως ADD, SUB,LOAD) που δείχνει τι κάνει η εντολή.Το πρόγραμμα γράφεται σε μορφή κειμένου με τη χρήση των εντολών. Έπειτα ο προγραμματιστής μετατρέπει μέσω του συμβολομεταφραστή το πρόγραμμα σε γλώσσα μηχανής που μπορεί να εκτελέσει ο υπολογιστής. Το σχήμα 1-2 δείχνει ένα παράδειγμα εντολής γραμμένη σε γλώσσα assembly και αντίστοιχη γλώσσα μηχανής.

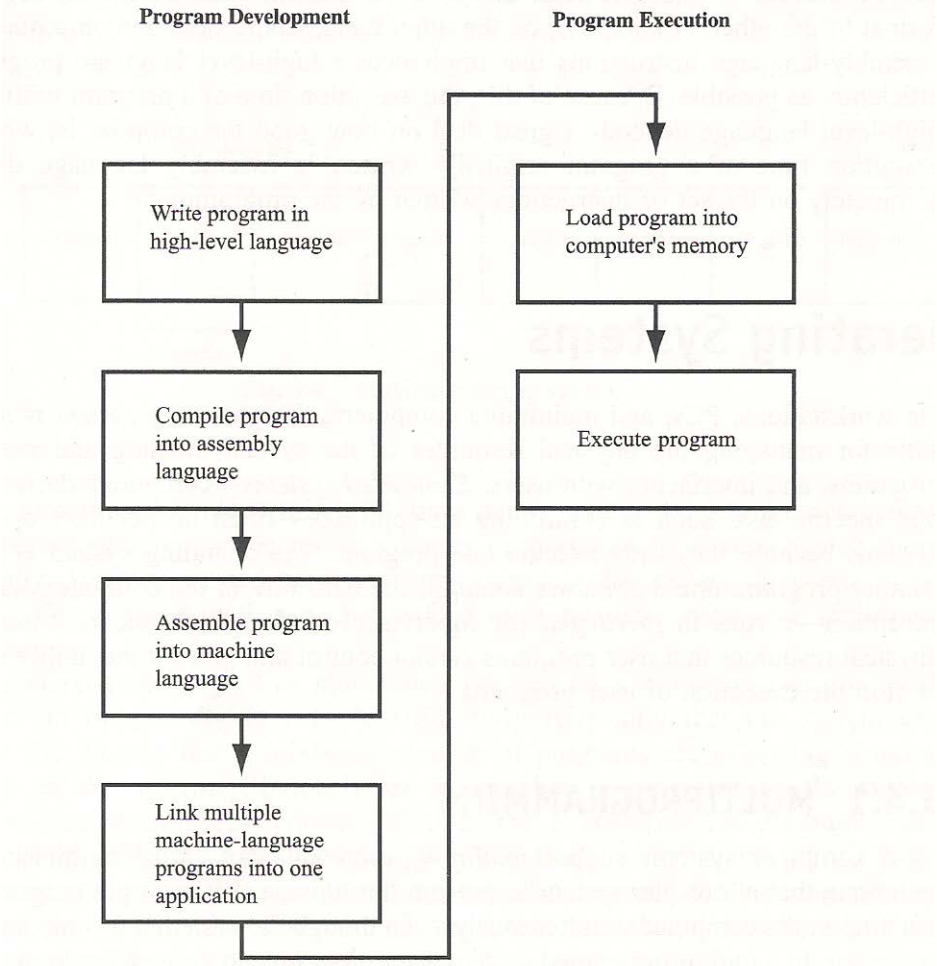


ΣΧΗΜΑ1-2 γλώσσα assembly

Με τη χρήση της assembly ο προγραμματισμός έγινε ευκολότερος αφού οι οδηγίες ήταν πλέον πιο κατανοητές από τους προγραμματιστές. Εξακολουθούσε όμως να είναι αρκετά μονότονο γιατί μια εντολή ουσιαστικά έκανε λίγη δουλειά και γιατί οι εντολές άλλαζαν από μηχάνημα σε μηχάνημα. Αν ο προγραμματιστής ήθελε να τρέξει το πρόγραμμα του σε άλλο μηχάνημα έπρεπε ουσιαστικά να το ξαναγράψει.

Γλώσσες υψηλού επίπέδου όπως οι FORTRAN, COBOL και C.δημιουργήθηκαν ακριβώς για την επίλυση αυτών των προβλημάτων μιας και μπορούν να καθορίσουν λεπτομερώς περισσότερες εντολές από ότι η assembly.Έρευνες έχουν δείξει ότι ο μέσος αριθμός εντολών που γράφει κάποιος προγραμματιστής είναι σχετικά ανεξάρτητος από τη γλώσσα που χρησιμοποιεί. Πράγμα που σημαίνει ότι με μια γλώσσα υψηλού επιπέδου ο προγραμματιστής γράφει λιγότερες εντολές και μπορεί να υλοποιήσει ένα πρόγραμμα σε λιγότερο χρόνο από ότι αν έγραφε σε assembly.

Ένα ακόμα πλεονέκτημα της χρήσης αυτών των γλωσσών είναι ότι τα προγράμματα που προκύπτουν μπορούν να χρησιμοποιηθούν και σε διαφορετικά μηχανήματα, αρκεί να γίνεται χρήση μεταφραστή από το εκάστοτε μηχάνημα. Σε αντίθεση, όπως είπαμε ένα πρόγραμμα σε assembly πρέπει να γραφεί εξ ολοκλήρου πράγμα που είναι πολύ χρονοβόρο.



ΣΧΗΜΑ1-3 program development

Το πρόβλημα με τις γλώσσες υψηλού επιπέδου είναι ότι το πρόγραμμα δεν μπορεί να εκτελεστεί αμέσως άλλα είναι απαραίτητη η χρήση μεταγλωττιστή (compiler) που θα μετατρέψει το πρόγραμμα σε γλώσσα assembly, που μετά θα μετατραπεί σε αριθμητική ακολουθία από ένα συμβολομεταφραστη (assembler).Το σχήμα 1-3 επεξηγεί τη διαδικασία της ανάπτυξης και εκτέλεσης ενός προγράμματος σε γλώσσα υψηλού επιπέδου. Εναλλακτικά αντί για τη χρήση μεταγλωττιστή μπορεί να γίνει χρήση του διερμηνέα (interpreter).Οι διερμηνείς είναι προγράμματα που δέχονται ως είσοδο το πρόγραμμα σε γλώσσα υψηλού επιπέδου και επιστρέφουν το ίδιο αποτέλεσμα που θα είχαμε αν πρώτα χρησιμοποιούσαμε μεταγλωττιστή και έπειτα τρέχαμε την μεταγλωττισμένη έκδοση του προγράμματος.. Τα διερμηνευμένα προγράμματα όμως είναι πιο αργά από τα μεταγλωττισμένα γιατί ο διερμηνέας πρέπει να εξετάσει κάθε οδηγία πριν την εκτελέσει. Σε πολλούς τομείς μοιάζει με τη λειτουργία του μεταγλωττιστή να καθορίσει τη σωστή assembly οδηγία που θα υλοποιήσει την οδηγία της γλώσσας υψηλού επιπέδου. Η διαφορά είναι ότι ο

διερμηνέας πρέπει κάθε φορά να κάνει αυτή τη διαδικασία. Αν ένα πρόγραμμα περιέχει ένα βρόγχο (loop) που πρέπει να εκτελεστεί 10000 φορές, ο διερμηνέας θα τον εξετάσει 10000 φορές, ενώ ο μεταγλωττιστής μόνο μία.

Δεδομένου του μειονεκτήματος της ταχύτητας οι διερμηνείς δεν είναι δημοφιλή προγράμματα χρησιμοποιούνται κυρίως σε περιπτώσεις που το ίδιο πρόγραμμα πρέπει να εκτελεστεί σε υπολογιστές διαφορετικού τύπου χωρίς να μεταγλώττιση Σε τέτοια περίπτωση ο διερμηνέας επιτρέπει την εκτέλεση του προγράμματος απευθείας.

Οι μεταγλωττιστές και οι συμβολομεταφραστές έχουν εντελώς διαφορετικές αποστολές. Γενικά υπάρχει αντιστοιχία ένα σε ένα ανάμεσα στη γλώσσα μηχανής και τη γλώσσα assembly, και έτσι ο συμβολομεταφραστής το μόνο που έχει να κάνει είναι να μετατρέψει τις εντολές από τη μια μορφή στην άλλη. Ο μεταγλωττιστής από την άλλη, πρέπει να αποφασίσει ποια είναι η σωστή σειρά από assembly εντολές, που θα υλοποιήσει τις οδηγίες που του δόθηκαν από την γλώσσα υψηλού επιπέδου.
Γιαυτο και ο χρόνος εκτέλεσης ενός προγράμματος γραμμένου σε γλώσσα υψηλού επιπέδου εξαρτάται από το πόσο καλός είναι ο μεταγλωττιστής, ενώ ο χρόνος εκτέλεσης ενός προγράμματος σε assembly εξαρτάται από το σετ οδηγιών που έγραψε ο προγραμματιστής.

1.4 Λειτουργικά προγράμματα

Σε σταθμούς εργασίας, Η/Υ και σε κεντρικούς υπολογιστές μεγάλης ισχύος το λειτουργικό σύστημα είναι υπεύθυνο για την διαχείριση των μέσων του συστήματος, για τη φόρτωση και εκτέλεση των προγραμμάτων και για τη αλληλεπίδραση με τους χρήστες.

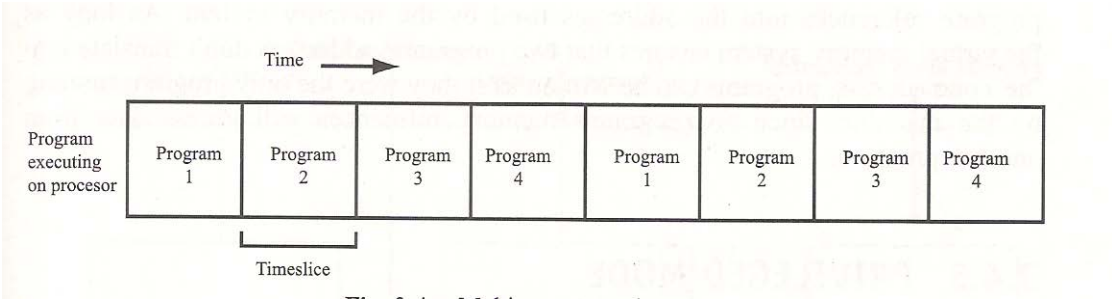
Τα Embedded systems δηλ συστήματα από υπολογιστές σχεδιασμένους για ένα συγκεκριμένο σκοπό, όπως είναι ο έλεγχος κάποιας συσκευής, συχνά δεν έχουν λειτουργικό σύστημα γιατί χρησιμοποιούν ένα μόνο πρόγραμμα. Το λειτουργικό σύστημα είναι απλώς ένα άλλο πρόγραμμα που γνωρίζει όλο το hardware που υπάρχει στον υπολογιστή και που έχει επιβλέποντα ρόλο. Με αυτό τον τρόπο έχει πρόσβαση σε όλα τα μέσα που άλλα προγράμματα δε μπορούν να ελέγξουν και μπορεί να επιτρέψει την εκτέλεση ή και να σταματήσει ένα άλλο πρόγραμμα.

1.4.4 Πολυπρογραμματισμός

Τα περισσότερα συστήματα υπολογιστών υποστηρίζουν τον πολυπρογραματισμό (multiprogramming), επίσης γνωστό ως πολυδιεργασία (multitasking).Πρόκειται για μια τεχνική η οποία επιτρέπει στο σύστημα να μας δώσει την ψευδαίσθηση ότι πολλά προγράμματα τρέχουν στον υπολογιστή ταυτόχρονα, παρόλο που το σύστημα διαθέτει μόνο ένα επεξεργαστή. Σε ένα τέτοιο σύστημα τα προγράμματα δεν χρειάζεται να γνωρίζουν ποια ούτε πόσα άλλα προγράμματα τρέχουν την ίδια στιγμή στο σύστημα. Το λειτουργικό σύστημα και το hardware προστατεύουν τα προγράμματα από το να εισβάλει το ένα στις πληροφορίες του άλλου εκτός και αν δύο προγράμματα είναι προγραμματισμένα να βλέπει το ένα τις πληροφορίες του άλλου. Πολλά πολυπρογραμματιστικά προγράμματα είναι και πολυχρηστικά (multiuser) , δηλαδή επιτρέπουν σε περισσότερους από ένα χρήστη να βρίσκονται συνδεδεμένοι στο σύστημα. Αυτά τα συστήματα απαιτούν από το λειτουργικό

σύστημα όχι μόνο να προστατεύει τις πληροφορίες του ενός προγράμματος από το άλλο αλλά και τις προσωπικές πληροφορίες κάθε χρήστη από αυτές του άλλου.

Ένα πολυπρογραματιστικό σύστημα δίνει την ψευδαίσθηση ότι τα πολλά προγράμματα τρέχουν ταυτόχρονα και το πετυχαίνει αλλάζοντας ταχύτατα το ένα πρόγραμμα με το επόμενο όπως φαίνεται και στο σχήμα 1-4.Κάθε πρόγραμμα επιτρέπεται να εκτελείται για ένα καθορισμένο χρόνο που είναι γνωστός ως timeslice (χρονομερίδιο).Όταν ο χρόνος αυτός παρέλθει το λειτουργικό σύστημα το σταματά το βγάζει από τον επεξεργαστή και φορτώνει στη θέση του το επόμενο. Η διαδικασία αυτή είναι γνωστή ως αλλαγή περιεχομένου (context switch). Για να πραγματοποιήσει αυτή την αλλαγή το λειτουργικό σύστημα αντιγράφει τα περιεχόμενα του φακέλου εγγραφής (register file) του τρέχοντος προγράμματος στη μνήμη και μετά μεταφέρει τα περιεχόμενα του φακέλου εγγραφής του επόμενου προγράμματος από τη μνήμη στο πίσω στο φάκελο. Τα προγράμματα δεν ξέρουν ότι λαμβάνει χώρα μια τέτοια αλλαγή. Το καθένα νομίζει ότι βρίσκεται διαρκώς στον επεξεργαστή.



ΣΧΗΜΑ1-4 multiprogrammed system

Οι περισσότεροι υπολογιστές αλλάζουν τα περιεχόμενα του επεξεργαστή 60 φορές το δευτερόλεπτο, δηλαδή το χρονομερίδιο (timeslice) είναι 1/60στο του δευτερολέπτου, αλλά υπάρχουν και συστήματα που κάνουν αυτή την αλλαγή συχνότερα. Αυτό προκάλεσε κάποια προβλήματα με τα προγράμματα που περιμένουν χρονικό περιθώριο 1/60στο και που χρησιμοποιούν αυτό το χρόνο για τη σωστή εκτέλεσή τους.

Αλλάζοντας τα περιεχόμενα 60 ή περισσότερες φορές το δευτερόλεπτο, ο υπολογιστής δίνει σε κάθε πρόγραμμα την ευκαιρία να εκτελεστή ικανοποιητικά και παράλληλα να δίνει την εντύπωση ότι εκτελούνται όλα ταυτόχρονα. Σε περιπτώσεις όμως που ένας μεγάλος αριθμός προγραμμάτων εκτελείται ταυτόχρονα η καθυστέρηση γίνεται ορατή. Για παράδειγμα, σε ένα σύστημα που εκτελούνται 120 προγράμματα, κάθε πρόγραμμα έχει μόλις 2 δευτερόλεπτα χρόνο στον επεξεργαστή, και η καθυστέρηση γίνεται αντιληπτή. Με τον πολυπρογραμματισμό μπορεί να αυξηθεί και ο χρόνος εκτέλεσης των εφαρμογών γιατί οι πόροι του συστήματος κατανέμονται σε όλα τα προγράμματα που τρέχουν.

1.4.2 Προστασία

Όπως είδαμε μία από τις κυριότερες απαιτήσεις ενός multiprogramming λειτουργικού συστήματος είναι να παρέχει προστασία στα προγράμματα που τρέχουν στον υπολογιστή. Αυτό σημαίνει ότι κάθε πρόγραμμα που τρέχει πρέπει να αντιμετωπίζεται από το λειτουργικό σύστημα σα να τρέχει μόνο αυτό. Δεν μπορεί αν έχει πρόσβαση σε περιεχόμενα άλλου προγράμματος και ούτε επιτρέπεται οι δικές του πληροφορίες να αλλάξουν εξαιτίας κάποιου άλλου προγράμματος. Παρομοίως τα προγράμματα δεν επιτρέπεται να επεμβαίνουν στη χρήση του I/O υποσυστήματος κάποιου άλλου προγράμματος.

Η παροχή προστασίας σε ένα πολυπρογραμματιστικό και πολυχρηστικό σύστημα απαιτεί ότι το λειτουργικό σύστημα θα ελέγχει όλους του πόρους του συστήματος, συμπεριλαμβανομένου του επεξεργαστή, της μνήμης και των συσκευών I/O. Αλλιώς κάθε πρόγραμμα θα έχει τη δυνατότητα να εισβάλει σε οποιοδήποτε από τα παραπάνω και να εκμεταλλευτεί πληροφορίες που ανήκουν σε άλλο πρόγραμμα ή χρήστη. Αυτός ο απόλυτος έλεγχος επιτρέπει στο λειτουργικό σύστημα να αποκλείει περισσότερα από ένα προγράμματα από η χρήση μιας I/O συσκευής όπως ο εκτυπωτής, ένα κάθε φορά.

Μια από τις τεχνικές που χρησιμοποιούνται για την προστασία των προγραμμάτων είναι και η εικονική μνήμη (virtual memory).Η εικονική μνήμη επιτρέπει σε κάθε πρόγραμμα να λειτουργεί σαν ν είναι το μοναδικό που τρέχει στον υπολογιστή. Αυτό το πετυχαίνει μεταφράζοντας τις διευθύνσεις μνήμης στις οποίες αναφέρεται το πρόγραμμα, σε διευθύνσεις που χρησιμοποιεί το σύστημα μνήμης. Το σύστημα εικονικής μνήμης εξασφαλίζει ότι οι διευθύνσεις δύο προγραμμάτων δε θα μεταφραστούν στην ίδια διεύθυνση, άρα τα δεδομένα του ενός προγράμματος δε μπορούν να προσπελαστούν από το άλλο.

1.4.3. Προνομιακή κατάσταση

Για να εξασφαλιστεί ότι το λειτουργικό σύστημα είναι το μόνο που μπορεί να ελέγξει τους πόρους του συστήματος, εκτελείται σε προνομιακή κατάστασή (privileged mode) ενώ οποιοδήποτε άλλο πρόγραμμα εκτελείται σε user mode(κάποιες φορές καλείται μη-προνομιακή κατάσταση). Ορισμένες εργασίες, όπως η πρόσβαση σε συσκευή I/O, η αλλαγή περιεχομένου του επεξεργαστή, ή η κατανομή μνήμης απαιτούν privileged mode από το πρόγραμμα. Έαν ένα απλό πρόγραμμα επιχειρήσει να κάνει κάτι από αυτά που αναφέραμε το hardware το σταματά και κάνει σήμα ότι υπήρξε σφάλμα. Γιαυτό και όταν ένα πρόγραμμα θέλει να κάνει κάποια εργασία που απαιτεί privileged mode στέλνει μια αίτηση στο λειτουργικό (system call) και το λειτουργικό εκτελεί την εργασία που του ζητήθηκε από το πρόγραμμα, εφόσον την εγκρίνει. Αλλιώς σημαίνει σφάλμα.

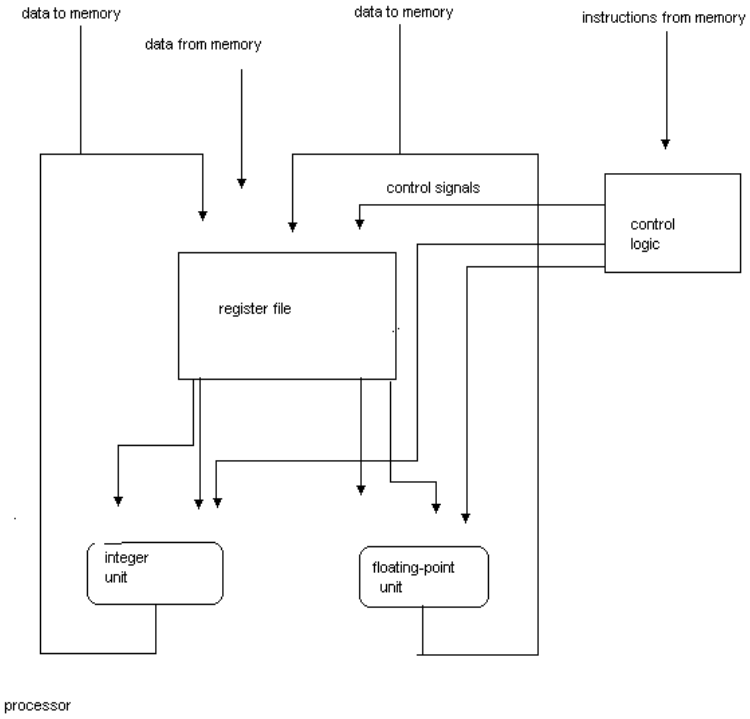
Επειδή ελέγχει τους πόρους του συστήματος το λειτουργικό σύστημα μπορεί να θεωρηθεί υπεύθυνο για χαμηλής ποιότητας διεπαφή με το χρήστη. Όταν ο χρήστης πιέζοντας ένα πλήκτρο ή με κάποιο άλλο τρόπο εισάγει δεδομένα στο σύστημα το λειτουργικό είναι υπεύθυνο να καθορίσει το κατάλληλο πρόγραμμα που θα παραλάβει τα δεδομένα που εισήχθησαν. Ακόμα όταν ένα πρόγραμμα θέλει να εκθέσει στην οθόνη κάποιο μήνυμα , πρέπει να εκτελέσει system call ώστε το λειτουργικό να παρουσιάσει το μήνυμα.

1.5 Η οργάνωση του υπολογιστή

Το σχήμα 1-1 παρουσίασε ένα διάγραμμα τυπικού υπολογιστικού συστήματος. Σε αυτό το τμήμα θα παρουσιάσουμε σύντομα μια εισαγωγή σε κάθε ένα από τα κύρια υποσυστήματα. Τον επεξεργαστή τη μνήμη και το I/O. Σκοπός αυτής της παρουσίασης είναι να καταλάβει ο αναγνώστης τα βασικά για τα υποσυστήματα αυτά που θα του είναι χρήσιμα στην αργότερη αναλυτική παρουσίαση τους.

1.5.1. Ο επεξεργαστής

Ο επεξεργαστής είναι ο κύριος υπεύθυνος για την εκτέλεση των εντολών που σχηματίζουν τα προγράμματα αλλά και τα λειτουργικά προγράμματα.. Όπως φαίνεται στο σχήμα 1-5, οι επεξεργαστές αποτελούνται από διάφορα κομμάτια: τις μονάδες εκτέλεσης (execution units) τα αρχεία εγγραφών (register files) και έλεγχο λογικής (control logic).Οι μονάδες εκτέλεσης περιλαμβάνουν το υλικό που εκτελεί τις οδηγίες. Πρόκειται για υλικό που φορτώνει και αποκωδικοποιεί εντολές, και αριθμητικές λογικές μονάδες (ALUs) που εκτελούν τον υπολογισμό. Πολλοί επεξεργαστές περιέχουν διαφορετικές μονάδες για ακέραιους ή δεκαδικούς υπολογισμούς γιατί είναι απαραίτητο το υλικό να μπορεί να διαχειριστεί αυτούς τους δύο τύπους πληροφοριών. Επιπρόσθετα όπως θα δούμε και στο κεφάλαιο 9, οι σύγχρονοι επεξεργαστές χρησιμοποιούν πολλαπλές εκτελεστικές μονάδες για την εκτέλεση των οδηγιών για να βελτιώσουν την απόδοση του συστήματος.



ΣΧΗΜΑ1-5 processor block diagram

Το αρχείο εγγραφής είναι μια μικρή αποθηκευτική περιοχή για δεδομένα που επεξεργάζεται ο επεξεργαστής. Διάφορες τιμές που αποθηκεύονται σε αρχεία εγγραφής μπορούν να προσπελαστούν γρηγορότερα από ότι αν αποθηκεύονταν στη μνήμη. Γενικά τα αρχεία αυτά υποστηρίζουν πολλές ταυτόχρονες προσπελάσεις. Έτσι μια εργασία, όπως μια πρόσθεση για παράδειγμα, μπορεί να διαβάσει όλες τις εισόδους της την ίδια στιγμή από το αρχείο εγγραφής, αντί να τις διαβάζει μία μία. Και όπως θα δούμε στο Κεφάλαιο 4 διαφορετικοί επεξεργαστές χρησιμοποιούν με διαφορετικό τρόπο τα αρχεία εγγραφής, αλλά εικονικά όλα τα αρχεία είναι παρόμοια.

Ο έλεγχος λογικής ελέγχει τα υπόλοιπα τμήματα του επεξεργαστή, καθορίζοντας ποιες εντολές θα εκτελεστούν και τι λειτουργίες πρέπει να γίνουν για την εκτέλεση τους. Στους πρώτους επεξεργαστές το κομμάτι αυτό ήταν μικρό τμήμα του επεξεργαστή, ειδικά σε σύγκριση με τις λογικές μονάδες και τα αρχεία εγγραφής. Στις μέρες μας όμως έχει αυξηθεί δραστικά αφού οι επεξεργαστές έγιναν πιο πολύπλοκοι. Έτσι σήμερα το control logic είναι ένα από τα πιο δύσκολα κομμάτια του επεξεργαστή.

1.5.2. Το σύστημα μνήμης

Το σύστημα μνήμης λειτουργεί ως αποθηκευτικός χώρος για τα δεδομένα και τις πληροφορίες των προγραμμάτων που χρησιμοποιεί ο υπολογιστής. Οι περισσότερο υπολογιστές έχουν δύο τύπου μνήμης. Τη ROM(read only memory) και τη RAM (random access memory).Όπως φαίνεται και από το όνομά της η ROM δεν μπορεί να τροποποιηθεί, αλλά μόνο να διαβαστεί. Η ROM χρησιμοποιείται για την αποθήκευση των εκτελέσιμων προγραμμάτων που χρησιμοποιεί ο υπολογιστής όταν ανοίγει ή επανεκκινείται. Τέτοιο πρόγραμμα είναι ο boot loader , που φορτώνει και αποφορτώνει το λειτουργικό πρόγραμμα από το δίσκο ή κάποια συσκευή I/O. Το boot προέρχεται από τη λέξη bootstrap, που σημαίνει αυτοδύναμος, με την έννοια ότι ο υπολογιστής μόνος του εκτελεί το πρόγραμμα που θα του φορτώσει το λειτουργικό σύστημα.

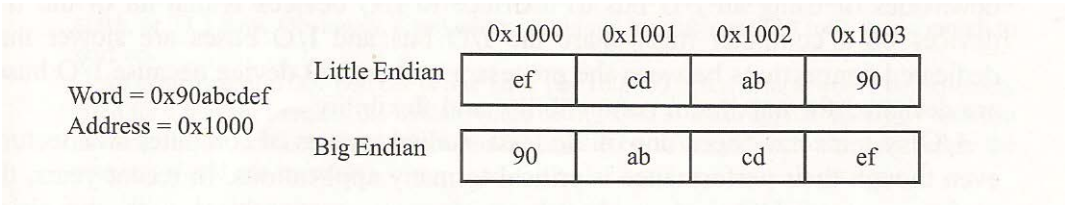
Η RAM δηλ η μνήμη τυχαίας προσπέλασης μπορεί να διαβαστεί αλλά και να τροποποιηθεί, και χρησιμοποιείται για να αποθηκεύσει τα προγράμματα, και το λειτουργικό σύστημα αλλά και οποιαδήποτε άλλα δεδομένα χρειάζεται τα σύστημα. Γενικά είναι ασταθής μνήμη γιατί χάνει τις πληροφορίες της μόλις κλείσει ο υπολογιστής. Οποιαδήποτε πληροφορία θέλουμε να σωθεί, πρέπει να γραφτεί σε ένα μόνιμο αποθηκευτικό μέσο, όπως ο σκληρός δίσκος.

Η μνήμη (RAM ή ROM) χωρίζεται σε ένα σετ αποθηκευτικών χώρων, το καθένα από τα οποία μπορεί να χωρέσει 1 byte (8 bits) δεδομένων. Αυτοί οι χώροι είναι αριθμημένοι και ο αριθμός(η διεύθυνσή του) χρησιμοποιείται για να υποδείξει στον επεξεργαστή πού να ψάξει για τα δεδομένα. Ένα πολύ σημαντικό χαρακτηριστικό του υπολογιστικού συστήματος είναι το εύρος των διευθύνσεων που χρησιμοποιεί, γιατί περιορίζει την ποσότητα της μνήμης στην οποία μπορεί να απευθυνθεί ο υπολογιστής. Οι περισσότεροι υπολογιστές χρησιμοποιούν διευθύνσεις είτε 32 bit είτε 64bit , δηλαδή μπορούν αν προσπελάσουν 2^{32} ή 2^{64} bytes μνήμης.

Μέχρι και το κεφάλαιο 9 θα χρησιμοποιούμε απλή μνήμη ταχείας προσπέλασης στην οποία όλες οι διεργασίες χρειάζονται τον ίδιο χρόνο. Το δικό μας σύστημα μνήμης θα υποστηρίζει δύο λειτουργίες : την φόρτωση και την αποθήκευση. Οι διαδικασίες χρειάζονται 2 τελεστές. Μια τιμή που θα αποθηκευτεί και άλλη μία με την τιμή της διεύθυνσης στην οποία θα αποθηκευτεί η τιμή. Λειτουργίες φόρτωσης παίρνουν τον τελεστή που καθορίζει τη διεύθυνση και βρίσκουν την τιμή που περιέχει.

Χρησιμοποιώντας αυτό το μοντέλο η μνήμη φαίνεται να λειτουργεί όπως ένα μεγάλο διαγραμμισμένο χαρτί, στο οποίο κάθε σειρά αντιστοιχεί με 1 byte αποθηκευτικό χώρο. Για να γράψεις (αποθηκεύσεις) μια τιμή στη μνήμη, μετράς από την αρχή της σελίδας μέχρι να βρεις τη σειρά με τη σωστή διεύθυνση. Τότε σβήνεις την τιμή που περιέχει και αποθηκεύεις την καινούρια. Για να τη διαβάσεις (φορτώσεις) ακολουθείς την ίδια διαδικασία από την αρχή έως ότου βρεις τη σωστή γραμμή και διαβάσεις την τιμή που είναι γραμμένη.
Οι περισσότεροι υπολογιστές επιτρέπουν στη μνήμη να φορτώσει ή να αποθηκεύσει περισσότερο από 1 byte.Γενικότερα η λειτουργία φόρτωσης ή αποθήκευσης χειρίζεται ποσότητα μνήμης ίση με το εύρος των bits του συστήματος. Οι διευθύνσεις που αποστέλλονται στη μνήμη είναι εκείνες με το μικρότερο byte.Για παράδειγμα ένα σύστημα 32 bit φορτώνει ή αποθηκεύει 32 bits(4 bytes)δεδομένων στα 4 bytes που ξεκινούν με τη διεύθυνση μνήμης. Έτσι μία φόρτωση της 424 μπορεί να επιστρέψει μια 32bit ποσότητα από τις τοποθεσίες 424, 425, 426, και 427.Για να απλοποιήσουν το σχεδιασμό του συστήματος μνήμης μερικοί υπολογιστές απαιτούν οι φορτώσεις και οι αποθηκεύσεις να είναι ευθυγραμμισμένες, δηλαδή ή διεύθυνση μνήμης να είναι πολλαπλάσιο του μεγέθους των δεδομένων που φορτώνονται ή αποθηκεύονται. Έτσι μια πληροφορία 4 bytes πρέπει να έχει ως διεύθυνση αριθμό πολλαπλάσιο του 4, μια 8 bytes να έχει διεύθυνση πολλαπλάσιο του 8 και ούτω καθεξής. Τα συστήματα που επιτρέπουν μη ευθυγραμμισμένες φορτώσεις και αποθηκεύσεις χρειάζονται περισσότερο χρόνο να ολοκληρώσουν τις λειτουργίες τους.

Ένα άλλο θέμα σχετικά με τις φορτώσεις και αποθηκεύσεις πολλών bytes είναι και η σειρά με την οποία γράφονται τα bytes στη μνήμη. Στους σύγχρονους υπολογιστές έχουν δύο τρόπους ταξινόμησης το little endian και το big endian Το Σχήμα 1-6 μας δίνει ένα παράδειγμα των δύο συστημάτων για το πώς γράφουν μια 32bit πληροφορία στη διεύθυνση 0x1000.



ΣΧΗΜΑ1-6 little endian versus big endian

Οι προγραμματιστές δεν χρειάζεται να γνωρίζουν ποιο από τα δύο endians χρησιμοποιεί το μηχάνημα με το οποίο δουλεύουν εκτός από την περίπτωση όπου η ίδια θέση μνήμης χρησιμοποιείται από φορτώσεις και αποθηκεύσεις διαφορετικού μήκους Παραδείγματος χάρη αν είχαμε ένα 1 byte του 0(if a 1-byte store of 0 into location 0x1000) στη διεύθυνση 0x1000στα συστήματα που παρουσιάζονται στο σχήμα 1-6, θα παίρναμε 0x90abcd00 από τον little endian και 0x00abcdef από τον big endian. Η διαφορετικότητα των endians μας απασχολεί κυρίως όταν έχουμε μετάδοση

από το ένα σύστημα στο άλλο. Γιατί αλλιώς ερμηνεύει ο big endian την ακολουθία των bytes και αλλιώς ο little. Έτσι οι πληροφορίες μετασχηματίζονται ανάλογα με το πιο σύστημα χρησιμοποιεί ο υπολογιστής που το διαβάζει..

Η σχεδίαση συστημάτων μνήμης επέφερε δραστικές αλλαγές στην επίδοση του υπολογιστή και συχνά είναι σημαντικός παράγων για την ταχύτητά του. Το εύρος ζώνης(δηλαδή το πόσες πληροφορίες μπορούν να φορτωθούν ή να αποθηκευτούν σε ένα συγκεκριμένο χρονικό όριο) και η λανθάνουσα κατάστασή (δηλαδή το πόσο χρόνο χρειάζεται μια διεργασία μνήμης για να ολοκληρωθεί) είναι καθοριστικής σημασίας παράγοντες για την εκτέλεση κάποιας διεργασίας. Η σχεδίαση συστημάτων μνήμης περιλαμβάνει και τη προστασία των δεδομένων του ενός προγράμματος από το άλλο) και κατά πόσο το σύστημα μνήμης αλληλεπιδρά με το σύστημα εισόδου/ εξόδου (I./O).

1.5.3. Το I/O υποσύστημα

Το I/O υποσύστημα περιλαμβάνει τις συσκευές που χρησιμοποιεί ο υπολογιστής για να επικοινωνήσει με το εξωτερικό περιβάλλον και τα αποθηκευμένα δεδομένα και πληροφορίες. Τέτοιες συσκευές είναι οι σκληροί δίσκοι, η οθόνη, οι εκτυπωτές, και οι μαγνητικοί οδηγοί. Όπως είδαμε και στο σχήμα 1-1 οι I/O συσκευές επικοινωνούν με τον επεξεργαστή δια μέσου ενός I/O διαύλου, που είναι ξεχωριστός από το δίαυλο μνήμης που χρησιμοποιεί ο επεξεργαστής για την επικοινωνία με το σύστημα μνήμης. Χρησιμοποιώντας ένα ξεχωριστό δίαυλο I/O ο υπολογιστής μπορεί αν επικοινωνήσει με ένα ευρύ πεδίο συσκευών I/O χωρίς να εγκαθιστά διεπαφή για την κάθε μία συσκευή. Επίσης ο δίαυλος υποστηρίζει ποικίλες συσκευές, επιτρέποντας στο χρήστη να εγκαταστήσει και άλλες καινούριες στην πορεία. Οι συσκευές είναι σχεδιασμένες έτσι ώστε να είναι συμβατές μ με κάθε τύπο υπολογιστή αρκεί αν έχει τον ίδιο τύπο διαύλου. Σχεδόν όλοι οι υπολογιστές έχουν τον δίαυλο PCI που χρησιμοποιείται ως πρότυπο και εξασφαλίζει αυτή τη συμβατότητα.. Το μόνο που χρειάζεται είναι ένα οδηγός συσκευής (device driver) για κάθε λειτουργικό σύστημα, δηλαδή ένα πρόγραμμα που επιτρέπει στο λειτουργικό να διαχειρίζεται τις συσκευές I/O. Με λίγα λόγια όλες οι συσκευές συνδέονται στον ίδιο δίαυλο γιαυτό και συχνά προκαλούνται καθυστερήσεις αν και οι δίαυλοι σχεδιάζονται έτσι ώστε να αποδίδουν το μέγιστο.

Τα συστήματα I/O δεν μελετούνται λιγότερο από τα υπόλοιπα στην αρχιτεκτονική υπολογιστών παρόλο που από την επίδοσή του εξαρτώνται πολλές εφαρμογές. Τα τελευταία χρόνια η σημασία τους έγινα ακόμα σπουδαιότερη με την ανάπτυξη των βάσεων δεδομένων και και των συστημάτων συναλλαγών αφού και τα δύο εξαρτώνται σε μεγάλο βαθμό από τα I/O συστήματα. Αυτό προκάλεσε μια δραστηριοποίηση στον τομέα βελτίωσής τους αφού οι εταιρείες(βάσεων δεδομένων και συστημάτων συναλλαγών) επενδύουν σοβαρά χρηματικά ποσά.

ΚΕΦΑΛΑΙΟ 2^{ο}

ΠΡΟΤΥΠΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

2.1 Στόχοι

Αυτό το κεφάλαιο περιγράφει τα πρότυπα προγραμματισμού που χρησιμοποιούνται σε δύο τύπους επεξεργαστών:στην αρχιτεκτονική stack based και στην αρχιτεκτονική καταχωρητή γενικής χρήσης .Αρχίζουμε με μια συζήτηση σχετικά με τους τύπους διαδικασιών που παρέχονται από τους περισσότερους επεξεργαστές.
Ακολουθεί μια περιγραφή της αρχιτεκτονικής βασισμένης στο σωρό και της αρχιτεκτονικής καταχωρητή γενικής χρησης .Η περιγραφή κάθε αρχιτεκτονικής περιλαμβάνει ένα δειγμά απο ένα σύνολο οδηγιών που τίθεται για εκείνο τον τύπο επεξεργαστή που θα αποτελέσει τη βάση για τα παραδείγματα και τις ασκήσεις σε όλο το υπόλοιπο αυτού του βιβλίου.Το κεφάλαιο ολοκληρώνεται με μια σύγκριση των δύο προτύπων προγραμματισμού και με μια συζήτηση για το πώς οι σωροί χρησιμοποιούνται για να εφαρμόσουν τις κλήσεις διαδικασίας, ακόμη και στην αρχιτεκτονική καταχωρητή γενικής χρησής .

Μετά την ολοκλήρωση αυτού του κεφαλαίου, πρέπει
1. Να είστε εξοικειωμένοι με τους διαφορετικούς τύπους διαδικασιών που παρέχονται στους περισσότερους επεξεργαστές.
2.Να καταλαβετε την αρχιτεκτονική stack based και να είστε σε θέση να γράψετε ένα μικρό προγραμμα συμβολικής γλώσσας σύμφωνα με τις οδηγίες που περιγράφονται σε αυτό το κεφάλαιο.
3.Να καταλάβετε την αρχιτεκτονική καταχωρητή γενικής χρησης και να είστε σε θέση
να γράψετε ένα μικρό προγραμμα συμβολικής γλώσσας για αυτή την αρχιτεκτονική.
4. Να είστε σε θέση να συγκρίνετε την αρχιτεκτονική stack based και την αρχιτεκτονική καταχωρητή γενικής χρήσης και να περιγράψετε τις καταστάσεις στις οποίες κάθε ύφος θα ήταν πιό κατάλληλο.
5.Να καταλάβετε τις διαδικασίες κλήσεις και πώς εφαρμόζονται.

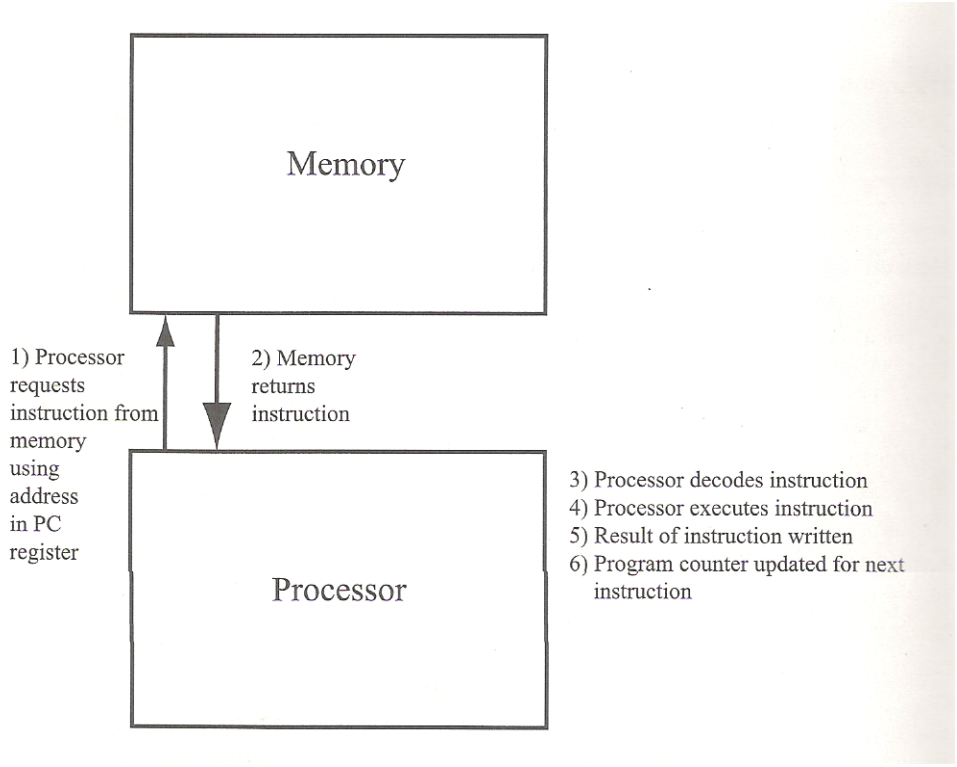
2.2 Εισαγωγή

Στο τελευταίο κεφάλαιο, περιγράψαμε τα προγράμματα ως σύνολο οδηγιών μηχανής, χωρίς να δώσουμε πολλές λεπτομέρειες για το ποιες οδηγίες είναι ή πώς εφαρμόζονται. Σε αυτό το κεφάλαιο, θα καλύψουμε τα δύο πρότυπα προγραμματισμού για τους επεξεργαστές:τη αρχιτεκτονική stack based και την αρχιτεκτονική καταχωρητή γενικής χρησης (GPR). Το πρότυπο προγραμματισμού ενός επεξεργαστή καθορίζει πώς οι οδηγίες έχουν πρόσβαση στους τελεστές τους και πώς οι οδηγίες περιγράφονται στη Συμβολική Γλώσσα (Assembly) του επεξεργαστή, αλλά όχι το σύνολο των διαδικασιών που παρέχονται από τον επεξεργαστή.Όπως θα δούμε, οι επεξεργαστές με διαφορετικά πρότυπα προγραμματισμού μπορούν να παρέχουν παρόμοια σύνολα διαδικασιών αλλά μπορούν να απαιτήσουν διαφορετικές προσεγγίσεις στον προγραμματισμό.

Το σχήμα 2-1 δίνει μια άποψη υψηλού επιπέδου για το πώς εκτελούνται οι οδηγίες που θα χρησιμοποιήσουμε σε αυτό το κεφάλαιο.Τα επόμενα κεφάλαια θα δώσουν πιό λεπτομερείς εξηγήσεις για την εκτέλεση μιας οδηγίας.Κατ' αρχάς, ο επεξεργαστής προσκομίζει (διαβάζει) την οδηγία από τη μνήμη. Η διεύθυνση της επόμενης οδηγίας που εκτελείται αποθηκεύεται σε έναν ειδικό κατάλογο γνωστό ως μετρητή προγράμματος (**P**rogram **C**ounter), που καλείται μερικές φορές δείκτης οδηγίας (**I**nstruction **P**ointer), έτσι ο επεξεργαστής μπορεί εύκολα να καθορίσει που πρέπει να ψάξει την επόμενη οδηγία στη μνήμη.

Μόλις το σύστημα μνήμης παραδώσει την οδηγία στον επεξεργαστή, ο επεξεργαστής εξετάζει την οδηγία για να δει τι πρέπει να κάνει για να συμπληρώσει την οδηγία, εκτελεί τη λειτουργία που διευκρινίζεται από την οδηγία, και γράφει το αποτέλεσμα της οδηγίας είτε σε έναν κατάλογο είτε στο σύστημα μνήμης. Ο επεξεργαστής έπειτα ενημερώνει το μετρητή πρόγραμματος για να περιέχει τη διεύθυνση της επόμενης οδηγίας που εκτελείται και επαναλαμβάνει τη διαδικασία.

Το υπόλοιπο αυτού του κεφαλαίου αρχίζει με μια συζήτηση των διαφορετικών τύπων οδηγιών που παρέχονται από τους περισσότερους επεξεργαστές. Δίνουμε έπειτα μια εισαγωγή στην αρχιτεκτονική stack based και παρουσιάζουμε ένα σύνολο οδηγιών για την αρχιτεκτονική stack based .Ακολουθεί μια συζήτηση για την αρχιτεκτονική καταχωρητή γενικής χρησης,συμπεριλαμβανομένου ενός δείγματος συνόλου οδηγιών αυτών των αρχιτεκτονικών.Το κεφάλαιο ολοκληρώνεται με μια συζήτηση για το πώς οι σωροί χρησιμοποιούνται για να εφαρμόσουν τη διαδικασία κλήσης και στις δύο αρχιτεκτονικες την αρχιτεκτονική stack based και την αρχιτεκτονική καταχωρητή γενικής χρησης .



ΣΧΗΜΑ2-1 basic instruction execution

2.3 Τύποι οδηγιών

Ένας από τους παράγοντες που διαφοροποιεί τους επεξεργαστές είναι τα σύνολα οδηγιών τους - τα σύνολα βασικών διαδικασιών που παρέχονται από κάθε επεξεργαστή. Οι πρόωροι υπολογιστές είχαν πολύ διαφορετικά σύνολα οδηγιών, και η σχεδίαση του σύνολο οδηγιών ήταν ένας από τους κύριους στόχους των αρχιτεκτόνων υπολογιστών. Δεδομένου ότι ο τομέας έχει προχωρήσει, το σύνολο διαδικασιών που παρέχεται από τους επεξεργαστές έχει συγκλίνει, και τώρα σχεδόν όλοι οι επεξεργαστές παρέχουν παρόμοια σύνολα οδηγιών, ανεξάρτητα από το εάν χρησιμοποιούν την αρχιτεκτονική stack based ή την αρχιτεκτονική καταχωρητή γενικής χρήσης. Αυτές οι βασικές διαδικασίες μπορούν να διαιρεθούν σε τέσσερις κατηγορίες: αριθμητικές διαδικασίες, διαδικασίες μνήμης, συγκρίσεις, και διαδικασίες ελέγχου (κλάδοι).

2.3.1 Αριθμητικές διαδικασίες

Οι αριθμητικές διαδικασίες εκτελούν βασικούς υπολογισμούς, όπως προσθέσεις, πολλαπλασιασμοί, λογικές διαδικασίες (AND-OR), και αντιγραφή στοιχείων.Γενικά παίρνουν μια ή δύο εισόδους, και παράγουν μια έξοδο.Οι αριθμητικές διαδικασίες διαβάζουν τις εισόδους τους και γράφουν τα αποτελέσματά τους στο αρχείο καταλόγων,αν και μερικές αρχιτεκτονικές CISC (complex instruction set computer) επιτρέπουν αριθμητικές διαδικασίες με αναφορά στη μνήμη
. Οι αρχιτεκτονικές CISC καλύπτονται λεπτομερέστερα στο επόμενο κεφάλαιο.

Operation	Function
ADD	Adds its two integer operands
FADD	Adds its two floating-point operands
SUB	Subtracts its second integer operand from its first
FSUB	Subtracts its second floating-point operand from its first
MUL	Multiplies its two integer operands
FMUL	Multiplies its two floating-point operands
DIV	Divides its first integer input by its second
FDIV	Divides its first floating-point input by its second
MOV	Copies its input (of any type) to its output, both of which may be of any type
OR	Performs a logical OR on its two inputs, which can be of any type
AND	Performs a logical AND on its two inputs, which can be of any type
NOT	Performs logical negation on its input, which can be of any type
ASH	Shifts (arithmetic) its first input by the number of positions specified in its second input
LSH	Shifts (logical) its first input by the number of positions specified in its second input

ΣΧΗΜΑ2-2 arithmetic operation

Το σχήμα 2-2 παρουσιάζει το σύνολο των αριθμητικών διαδικασιών που θα χρησιμοποιήσουμε σε αυτό το βιβλίο, το οποίο είναι αντιπροσωπευτικό των αριθμητικών διαδικασιών που παρέχονται από τους περισσότερους σύγχρονους επεξεργαστές.Οι περισσότεροι επεξεργαστές παρέχουν ένα σύνολο αυτών των διαδικασιών, έχοντας συχνά τις πολλαπλάσιες οδηγίες που παρέχουν τις παραλλαγές μιας ενιαίας λειτουργίας. Οι διαδικασίες που παρουσιάστηκαν εδώ επιλέχτηκαν ως συμβιβασμός μεταξύ της πληρότητας και της πολυπλοκότητας, με στόχο την εξασφάλιση ενός αρκετά πλούσιου σύνολου οδηγιών που τέθηκαν ως στόχος για να εκτελέσουν τα περισσότερα προγράμματα χωρίς να δυσκολέυουν τον αναγνώστη με την πολυπλοκότητα.Σημειώστε ότι πολλές από τις οδηγίες έχουν ακέραια μορφή(*integer*) και μορφή κινητής υποδιαστολής(*float*).Αυτό επιτρέπει στο υλικό να καθορίσει εάν πρέπει να μεταχειριστεί τις εισόδους των οδηγιών ως ακέραιους αριθμούς ή αριθμούς κινητής υποδιαστολής και καθορίζει ποιο αρχείο καταλόγων πρέπει να χρησιμοποιηθεί στις αρχιτεκτονικές που έχουν χωριστά αρχεία ακέραιων αριθμών και αριθμών κινητής υποδιαστολής.

2.3.2 Διαδικασίες μνήμης

Οι διαδικασιες μνήμης μεταφέρουν δεδομένα μεταξύ του επεξεργαστή και του συστήματος μνήμης. Όπως θα συζητηθεί στο κεφάλαιο 3, μια από τις μεγάλες διαφορές μεταξύ της αρχιτεκτονικης RISC (μειωμένος καθορισμένος υπολογιστής οδηγίας) και της αρχιτεκτονικής CISC είναι εάν η μνήμη μπορεί να προσπελάσει μόνο μέσω των διαδικασιών μνήμης ή εάν άλλες διαδικασίες μπορούν να έχουν πρόσβαση το σύστημα μνήμης.Για αυτό το βιβλίο, θα υποθέσουμε ότι οι επεξεργαστές έχουν δύο διαδικασίες μνήμης : load(LD) και store (ST).

Κάθε μια από αυτές τις διαδικασίες λειτουργεί σε ένα ποσό στοιχείων ίσο με το μέγεθος της λέξης της μηχανής, έτσι η LD διαβάζει μια λέξη των δεδομένων από τις διευθύνσεις ξεκινώντας απο την διεύθυνση που διευκρινίζεται με την εισοδό τους, και η ST γράφει μια λέξη απο τα στοιχεία στη μνήμη ξεκινώντας απο την διεύθυνση που διευκρινίζεται με την είσοδο της διευθύνσής της.Πολλοί επεξεργαστές παρέχουν ένα μεγαλύτερο σύνολο διαδικασιών load και store που λειτουργούν σε διαφορετικά ποσά στοιχείων.Δείτε το σχήμα 2-3.

Operation	Function
LD	Loads the contents of the address specified by its input operand into its destination
ST	Stores the value contained in its second input into the address specified by its first input.

ΣΧΗΜΑ 2-3 memory operations

2.3.3 Συγκρίσεις

Όπως υποδηλώνει και το ονομά τους , οι διαδικασίες σύγκρισης συγκρίνουν δύο ή περισσότερες τιμές έτσι ώστε το πρόγραμμα να μπορεί να λάβει αποφάσεις. Υπάρχει μεγάλη παραλλαγή μεταξύ των διαφορετικών επεξεργαστών στο πώς αυτοι χειρίζονται την έξοδο των διαδικασιών σύγκρισης. Μερικοί επεξεργαστές γράφουν τα αποτελέσματα της σύγκρισης σε έναν κατάλογο στο αρχείο καταλόγων.Άλλοι παρέχουν έναν ειδικό κατάλογο που κρατά το αποτέλεσμα της πιό πρόσφατης

διαδικασια σύγκρισης.Αρχιτεκτονικές που χρησιμοποιούν έναν ειδικό κατάλογο αποτελέσματος σύγκρισης έχουν χρησιμοποιηθεί λιγότερο στο παρελθόν ,όπως έχούν μόνο μια θέση για να βάλουν τα αποτελέσματα σύγκρισης καθιστά αδύνατο να εκτελέσουν τις πολλαπλάσιες συγκρίσεις παράλληλα.Το σχήμα 2-4 παρουσιάζει ένα κοινό σύνολο διαδικασιών σύγκρισης. Οι περισσότεροι επεξεργαστές παρέχουν επίσης ένα ισοδύναμο σύνολο με διαδικασίες σύγκρισης κινητής υποδιαστολής.

Operation	Function
EQ	Tests its two integer operands to see if they are equal
NEQ	Tests its two integer operands to see if they are not equal.
GT	Determines if its first integer operand is greater than its second
LT	Determines if its first integer operand is less than its second
GEQ	Determines if its first integer operand is greater than or equal to its second
LEQ	Determines if its first integer operand is less than or equal to its second

ΣΧΗΜΑ 2-4 comparison operations

Οι διαδικασίες σύγκρισης γενικά συνυπάρχουν με τις διαδικασίες ελέγχου για να δημιουργήσουν έναν άλμα υπο συνθήκη που εκτελεί ένα τμήμα του προγράμματος ή κάποιου άλλου ανάλογα με το αποτέλεσμα της σύγκρισης. Αυτή η χρήση είναι τόσο κοινή που πολλοί επεξεργαστές παρέχουν την διαδικασία άλματος υπο συνθήκη που συνδυάζει μια σύγκριση και ένα άλμα σε μια οδηγία.Για την απλότητα, τα σύνολα οδηγιών που παρουσιάζονται αργότερα σε αυτό το κεφάλαιο για την αρχιτεκτονική stack based και την αρχιτεκτονική καταχωρητή γενικής χρησης θα παραλείψουν τις οδηγίες σύγκρισης και θα παράσχουν μόνο τις υπό όρους οδηγίες άλματος, δεδομένου ότι τα υπό όρους άλματα είναι η πιό κοινή χρήση των συγκρίσεων .

2.3.4 Διαδικασίες ελέγχου

Οι διαδικασίες ελέγχου έχουν επιπτώσεις στη ροή προγράμματος του επεξεργαστη του PC .Όταν μια λειτουργία εκτός από μια λειτουργία ελέγχου εκτελεί,το υλικό αυξάνει το μετρητή προγράμματος από το μέγεθος της οδηγίας έτσι ώστε αυτό τώρα να δείχνει στην επόμενη οδηγία στο πρόγραμμα. Για παραδείγμα, σε μια αρχιτεκτονική στην οποία οι οδηγίες είναι 32 bit μακροχρόνιες, 4 θα προστεθούν στο PC αφότου εκτελέσθεί κάθε οδηγία επειδή 32 bit είναι 4 bytes.

Όταν μια λειτουργία ελέγχου εκτελεί, ο μετρητής προγράμματος τίθεται στην είσοδο της οδηγίας ελέγχου, αναγκάζοντας την εκτέλεση να μεταβεί σε ένα διαφορετικό σημείο του πρόγραμματος.Οι διαδικασίες ελέγχου, που καλούνται συχνά διακλαδώσεις , μπορούν να διαιρεθούν σε δύο κατηγορίες:απεριόριστες και υπό όρους διακλαδώσεις .Οι απεριόριστες διακλαδώσεις,αποκαλούνται επίσης και άλματα, πάντα θέτουν το PC ίσο με την είσοδο τους όταν εκτελούν. Οι υπό όρους διακλαδώσεις θέτουν το PC ίσο με την εισοδο τους εάν κάποιος όρος, όπως το αποτέλεσμα μιας σύγκρισης, είναι αληθινός. Ανάλογα με την αρχιτεκτονική, η σύγκριση μπορεί να εκτελεσθεί ως τμήμα της διαδικασίας διακλάδωσης, ή μπορεί να έχει εκτελεσθεί από μια χωριστή οδηγία νωρίτερα στο πρόγραμμα .Το σχήμα 2-5

παρουσιάζει ένα κοινό σύνολο διαδικασιών ελέγχου που θα υποθέσουμε ότι οι επεξεργαστές μας παρέχουν.

Όταν οι προγραμματιστές γράφουν συμβολική- γλωσσα(assembly)ή οι μεταγλωττιστές τα παράγουν από υψηλού -επιπέδου προγράμματα ,γενικά δεν διευκρινίζουν τις διευθύνσεις προορισμού των διακλάδώσεων ως διεύθυνση της οδηγίας προορισμού στη μνήμη. Αντ' αυτού, οι οδηγίες χαρακτηρίζονται με τιμές κειμένων, και οι εντολές διακλάδωσης αναφέρονται σε αυτές τις τιμές.Για παραδείγμα,στο (άπειρο) βρόχο που παρουσιάζεται στο σχήμα 2-6, η ετικέτα "loop_start" συνδέεται με την οδηγία που την ακολουθεί αμέσως. Η εντολη διακλαδωσης στο τέλος του βρόχου χρησιμοποιεί την ετικέτα "loop_start" ως στόχο της,δείχνοντας ότι το πρόγραμμα πρέπει να διακλαδιστεί πίσω στην οδηγία ακολουθώντας την ετικέτα μετά απο την εκτελέση της διακλάδωσης.Ένα από τα καθήκοντα του συμβολομετεφραστη είναι να υπολογιστεί η διεύθυνση που αντιστοιχεί σε κάθε ετικέτα και να εισάγει αυτη τη διεύθυνση σε οποιαδήποτε εντολη διακλάδωσης που αναφέρει η ετικέτα.Δεδομένου ότι θα συζητήσουμε λεπτομερέστερα στο επόμενο κεφάλαιο, αυτές οι διευθύνσεις εκφράζονται συχνά ώς μετατοπιση από τον κλάδο στον προορισμό τους απο την διεύθυνση του προορισμού στη μνήμη.

Operation	Function
BR (or JMP)	Sets the PC equal to the value of its input operand, causing the instruction at that address to be executed next.
BEQ	Sets the PC equal to the value of its first operand if its other two inputs are equal.
BNE	Sets the PC equal to the value of its first operand if its other two inputs are not equal.
BLT	Sets the PC equal to the value of its first operand if its second operand is less than its third operand.
BGT	Sets the PC equal to the value of its first operand if its second operand is greater than its third operand.
BLE	Sets the PC equal to the value of its first operand if its second operand is less than or equal to its third operand.
BGE	Sets the PC equal to the value of its first operand if its second operand is greater than or equal to its third operand.

ΣΧΗΜΑ 2-5 control operations

Η χρησιμοποίηση των ετικετών αντί των αριθμητικών διευθύνσεων έχει δύο πλεονεκτήματα.Κατ' αρχάς, είναι πολύ ευκολότερο για τον προγραμματιστή να καταλάβει. Εξετάζοντας το παράδειγμα με τον κώδικα στο σχήμα 2-6, είναι σαφές που είναι ο στόχος της εντολής διακλάδωσης, ακόμα κι αν δεν ξέρετε τίποτα για την αρχιτεκτονική του επεξεργαστή.Εάν ο στόχος διευκρινιστεί από μια αριθμητική διεύθυνση, θα έπρεπε να ξέρετε πόσο διαστημα καθε οδηγία παίρνει μέχρι να είναι σε θέση να υπολογίσει τον προορισμό κάθε διακλάδωσης.Το δεύτερο πλεονέκτημα της χρησιμοποίησης ετικέτων είναι ότι η διεύθυνση που αντιστοιχεί σε μια ετικέτα μπορεί να αλλάξει κάθε φορά που άλλαζει ο αριθμός οδηγιών πριν από την διακλάδωση.

Αντ' αυτού, ο προγραμματιστής ή ο μεταγλωττιστής χρησιμοποιεί ετικέτες για να διευκρινίσει τον προορισμό κάθε διακλάδωσης, και ο συμβολομεταφραστής υπολογίζει τη διεύθυνση κάθε ετικέτας όταν συγκεντρώνεται το πρόγραμμα.

```
loop_start:
        (instruction)
        (instruction)
        (instruction)
        BR loop_start;
```

ΣΧΗΜΑ 2-6 label example

2.4 Stack Based αρχιτεκτονική

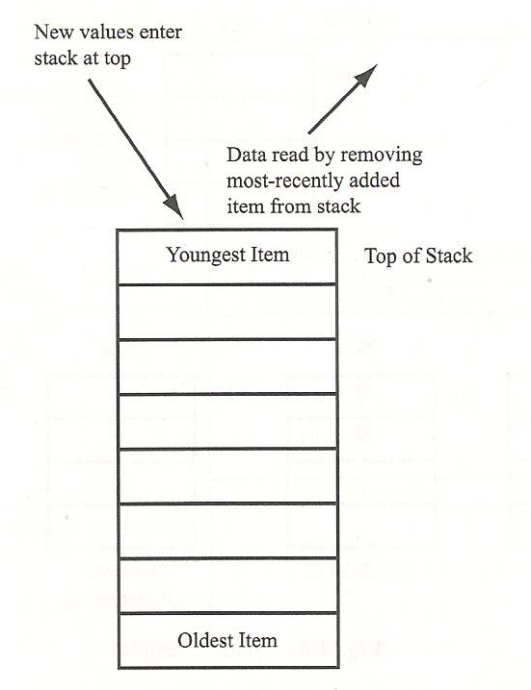
Σε μια αρχιτεκτονική stack based , το αρχείο καταλόγων είναι αόρατο στο πρόγραμμα.Αντ'αυτου, οι οδηγίες διαβάζουν τους τελεστέους τους , και γράφουν τα αποτελέσματά τους, ένας σωρός, μια last-in-first-out (LIFO) δομή δεδομένων.

2.4.1Ο σωρός

Όπως διευκρινίζεται στο σχήμα 2-7, ο σωρός είναι μια last-in-first-out δομή δεδομένων (LIFO).Το όνομα σωρός προέρχεται από το γεγονός ότι η δομή δεδομένων ενεργεί όπως ένα σωρός πίατων ή άλλων στοιχείων -- όταν τίθεται ένα νέο πιάτο στο σωρό των πιάτων, πηγαίνει στην κορυφή, και είναι το πρώτο πιάτο που αφαιρείται όταν παίρνει κάποιος ένα πιάτο μακριά του σωρού.Οι σωροί αποτελούνται από ένα σύνολο θέσεων, κάθε μία από τις οποίες μπορεί να κρατήσει μια λέξη των δεδομένων.Όταν μια τιμή προστίθεται στο σωρό, τοποθετείται στην κορυφαία θέση του σωρού, και όλα τα δεδομένα συγχρονως στο σωρό κινούνται προς τα κάτω κατα μία θέση. Μονο τα δεδομένα που βρίσκονται στην κορυφή του σωρού μπορούν να αφαιρεθούν.Όταν αυτό γίνεται, όλα τα άλλα δεδομενα στο σωρό κινούνται επάνω σε μια θέση. Γενικά,τα δεδομένα δεν μπορούν να διαβαστούν από έναν σωρό χωρίς διατάρραξη του σωρού, αν και μερικοί επεξεργαστές μπορούν να παρέχουν τις ειδικές διαδικασίες για να επιτρέψουν αυτό .

Οι σωροί υποστηρίζουν δυο βασικές διαδικασίες:την PUSH και την POP .Η διαδικασία PUSH παίρνουν ένα όρισμα και τοποθετεί την τιμή του ορισματός στην κορυφή του σωρού, που ωθεί όλα τα προηγούμενα στοιχεία προς τα κάτω κατα μια θέση. Η διαδικασια POP αφαιρεί την κορυφαία τιμή από το σωρό και την επιστρέφει, επιτρέποντας στην τιμή να χρησιμοποιηθεί σαν είσοδος μιας οδηγίας.Το σχήμα 2-8 δείχνει πώς ένα σύνολο απο PUSH και POP διαδικασίες έχουν επιπτώσεις σε έναν σωρό.

Αρχικά, ο σωρός είναι κενός.Η πρώτη λειτουργία PUSH τοποθετεί την τιμή 4 στην κορυφαία θέση του σωρού.Η δεύτερη λειτουργία PUSH τοποθετεί την τιμή 5 στην κορυφή του σωρού, που ωθεί το 4 κάτω στην επόμενη θέση. Η λειτουργία POP εκτελείται έπειτα, η οποία αφαιρεί το 5 από την κορυφή του σωρού και το επιστρέφει.Το 4 έπειτα κινείται στην κορυφή του σωρού.Τέλος, μια PUSH 7 λειτουργία εκτελείται, αφήνοντας το 7 στην κορυφή του σωρού και το 4 στην επόμενη θέση κάτω.

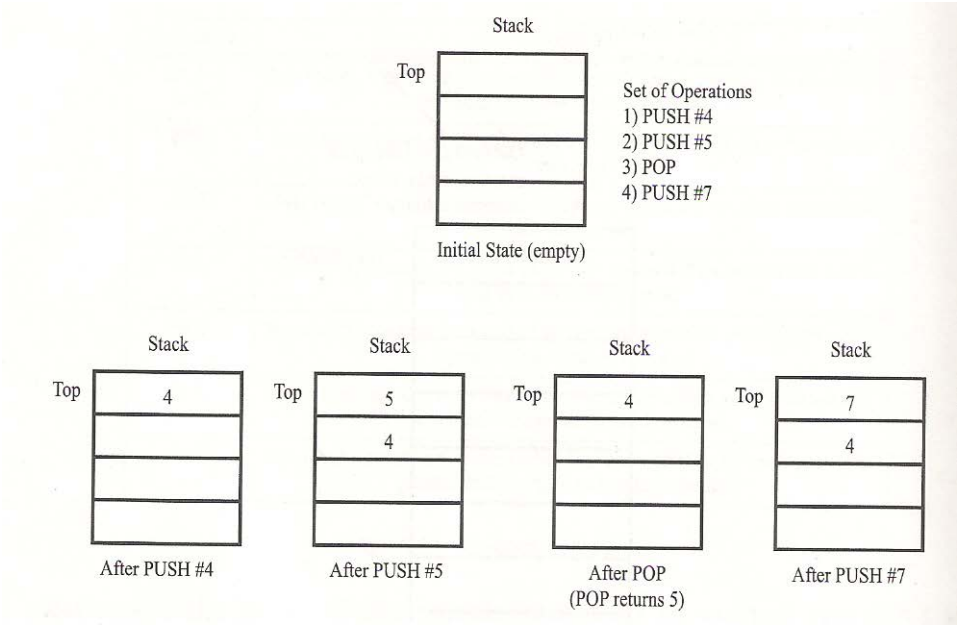


ΣΧΗΜΑ2-7 A stack

2.4.2 Εφαρμογή των σωρών

Σαν αφηρημένη δομή δεδομένων, οι σωροί υποτίθεται ότι είναι απείρως βαθιοί, σημαίνοντας ότι ένα αυθαίρετο ποσό δεδομενών μπορεί να τοποθετηθεί στο σωρό από το πρόγραμμα.Στην πράξη, οι σωροί εφαρμόζονται χρησιμοποιώντας την ενδίαμεση μνήμη, οι οποίοι είναι πεπερασμένοι στο μέγεθος.Εάν το ποσό των δεδομένων στο σωρό ξεπερνά το ποσό του διαστήματος που διατίθεται στο σωρό, εμφανίζεται ένα λάθος υπερχείλισης σωρών.

Το σχήμα 2-9 επιδεικνύει πώς ένας σωρός εφαρμόζεται στο σύστημα μνήμης ενός υπολογιστή.Μια σταθερή θέση καθορίζει το κατώτατο σημείο του σωρού, και ένας δείκτης δίνει τη θέση της κορυφής του σωρού (η θέση της τελευταίας τιμής που ωθείται επάνω στο σωρό).Όταν μια τιμή ωθείται επάνω στο σωρό,ο κορυφαίος δείκτης του σωρού αυξάνεται από το μέγεθος της λέξης της μηχανής και η τιμή που ωθείται αποθηκεύεται στη μνήμη στη διεύθυνση που δείχνεται από τη νέα τιμή του κορυφαίου δείκτη του σωρού.Για να απομακρυνουμε μια τιμή μακριά απο το σωρό, η τιμή στη θέση που δείχνεται διαβάζεται από τον κορυφαίο δείκτη του σωρού , και ο κορυφαίος δείκτης του σωρού μειώνεται από το μέγεθος λέξης της μηχανής.Όταν η κορυφή του σωρού και ο κατώτατος δείκτης είναι ιδιοι , ο σωρός είναι κενός, και μια προσπάθεια να απομακρυνθουν τα στοιχεία μακριά των αποτελεσμάτων του σωρού είναι ένα λάθος.Διάφορες παραλλαγές σε αυτήν την προσέγγιση είναι δυνατές, συμπεριλαμβανομένων αυτών όπου τα σημεία κατώτατων δεικτών δείχνουν στην υψηλότερη διεύθυνση στο σωρό και ο σωρός αυξάνεται προς τις χαμηλότερες διευθύνσεις.



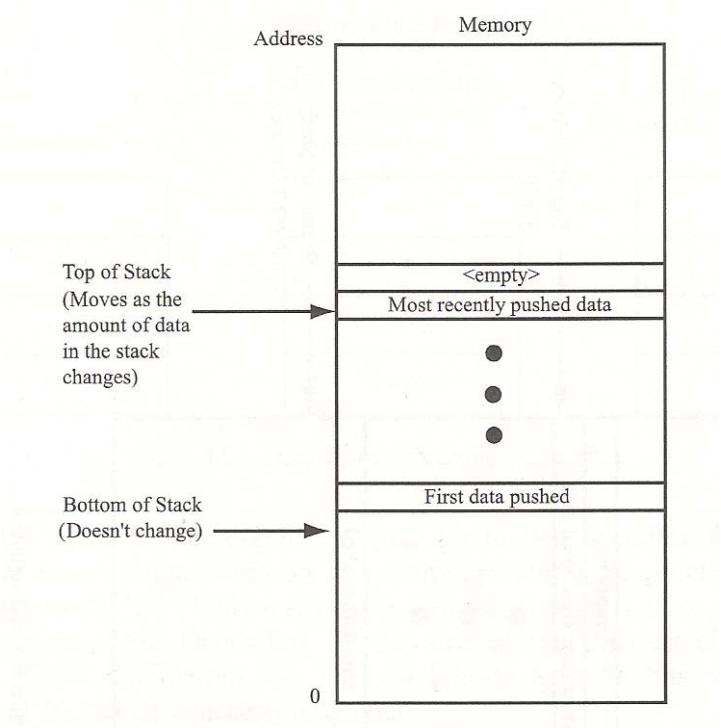
ΣΧΗΜΑ2-8 stack example

Αυτή η προσέγγιση οδηγεί σε έναν απολύτως λειτουργικό σωρό, αλλά η πρόσβαση του σωρού τείνει να είναι σχετικά αργή, λόγω της λανθάνουσας κατάστασης του συστήματος μνήμης.Εάν ο σωρός κρατήθηκε εντελώς στη μνήμη, η εκτέλεση μιας χαρακτηριστικής αριθμητικής οδηγίας, όπως ADD, θα απαιτούσε τέσσερις διαδικασίες μνήμης:μια για να προσκομίσει την οδηγία,δύο για να προσκομίσει τους τελεστέους από το σωρό,και μία για να γράψει το αποτελέσμα επάνω στο σωρό.Για να επιταχύνει τις προσβάσεις στους σωρούς,οι βασισμένοι στο σωρό επεξεργαστές μπορούν να ενσωματώσουν ένα αρχείο καταλόγων στον επεξεργαστή και να κρατήσουν τις κορυφαίες τιμές N (όπου το N είναι το μέγεθος του αρχείου καταλόγων) στο σωρό στο αρχείο καταλόγων.

Το σχήμα 2-10 επεξηγεί πώς αυτό δουλευει .Ουσιαστικά, το αρχείο καταλόγων αντιμετωπίζεται ως ενδίαμεση μνήμη από την ενδίαμεση μνημη στην κύρια μνήμη, με το δικό του κορυφαίο δείκτη του σωρού.Δεδομένου ότι το αρχείο καταλόγων περιέχει έναν σταθερό αριθμό θέσεων αποθήκευσης, δεν απαιτείτα ένα κατώτατος δείκτης σωρού .Ο κορυφαίος δείκτης σωρού παρακολουθεί απλά πόσοι από τους καταλόγους περιέχουν δεδομένα. Για να ωθήσει μια τιμή επάνω στο σωρό, ο κορυφαίος δείκτης σωρού αρχείων καταλόγων αυξάνεται, και η αξία αντιγράφεται στον κατάλογο τον οποίο δείχνει.Όταν το αρχείο καταλόγων γίνεται πλήρες, το περιεχόμενό του αντιγράφεται στον ενδίαμεσο σωρό στη μνήμη, και οι κορυφαίοι δείκτες σωρού και στη μνήμη και στο αρχείο καταλόγων ρυθμίζονται για να απεικονίσουν ότι το αρχείο καταλόγων είναι τώρα κενό και ότι η ενδίαμεση μνήμη περιέχει περισσότερα δεδομένα.Οι διαδικασίες POP χρησιμοποιούν μια παρόμοια προσέγγιση, με το αρχείο καταλόγων που ξαναγεμίζεται από τον απομονωτή στη μνήμη όταν γίνεται κενό.

Αυτή η προσέγγιση μπορεί να οδηγήσει σε πολλές προσβάσεις μνήμης όταν το αρχείο καταλόγων είναι σχεδόν πλήρες ή σχεδόν κενό και εναλλαγή PUSHes και POPs, επειδή το αρχείο καταλόγων αντιγράφεται συνεχώς σε και από τη μνήμη.Τα συστήματα μπορούν να επιλέξουν μόνο την μισό-αδειο ή το μισο-γεματό αρχείο

καταλόγων όταν αυτό υπερχειλισει πανω ή κατω για να μειώσουν αυτήν την επίδραση.

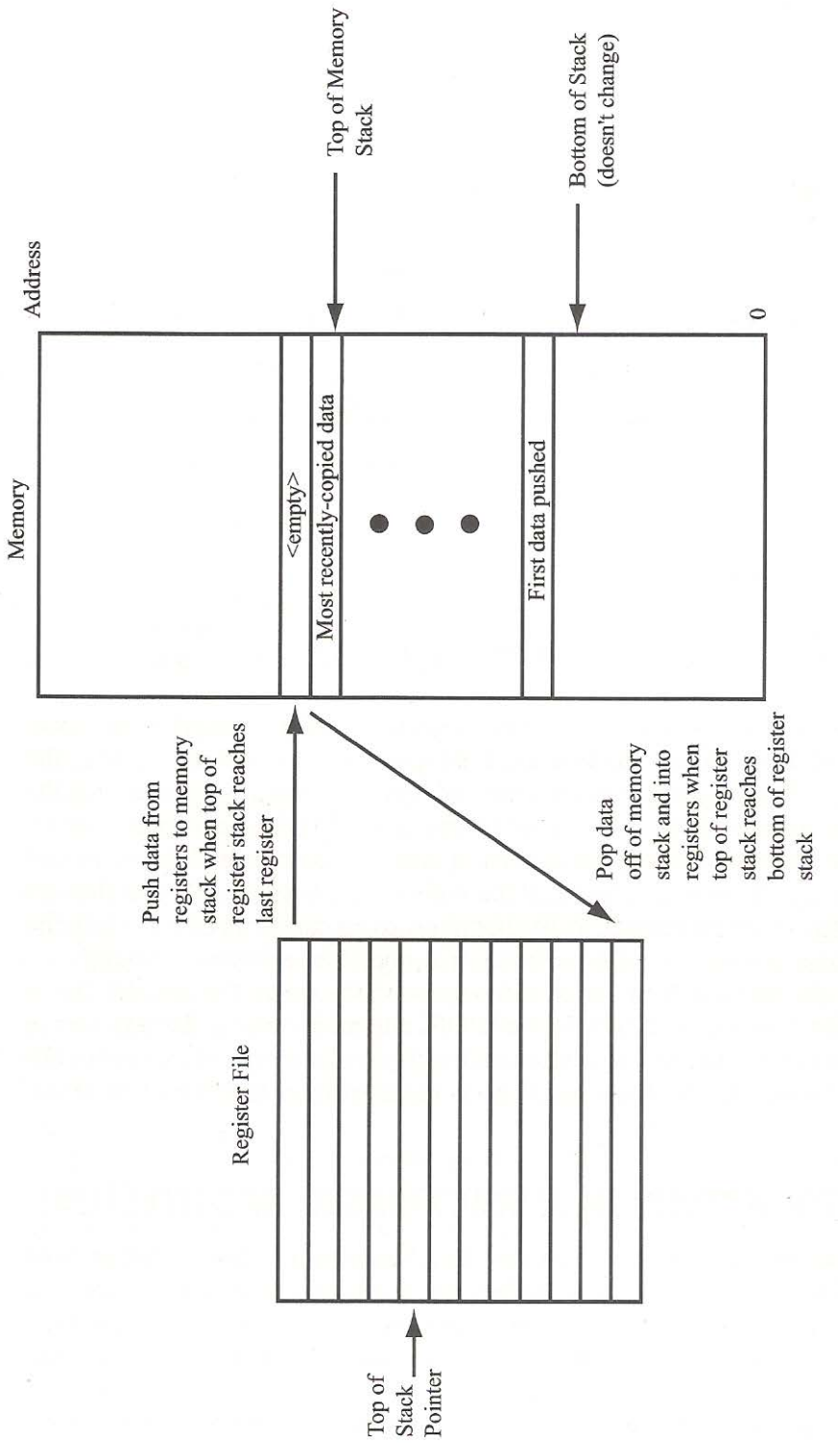


ΣΧΗΜΑ2-9 stack implementation in memory

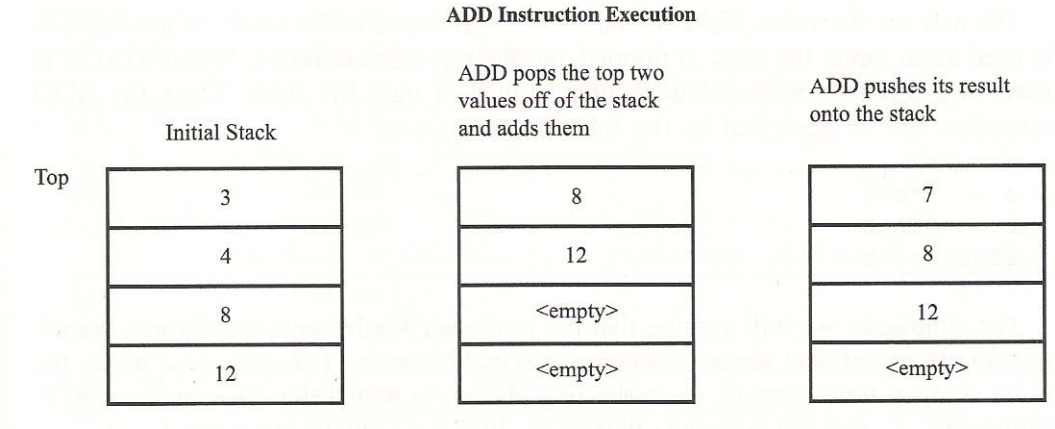
2.4.3 Οδηγίες σε μια stack based αρχιτεκτονική

Όπως αναφέρθηκε προηγουμένως,οι οδηγίες σε μια stack based αρχιτεκτονική παίρνουν τους τελεστέους τους και γράφουν τα αποτελέσματά τους στο σωρό.Όταν μια οδηγία εκτελεί,αυτη ανακτει απο το σωρο τους τελεστέους της, εκτελεί τον απαραίτητο υπολογισμό, και ωθεί το αποτέλεσμα πάνω στην κορυφή του σωρού.Το σχήμα 2-11 παρουσιάζει ένα παράδειγμα εκτελεσης της οδηγίας ADD σε μια stack based αρχιτεκτονική.

Ένα από τα σημαντικά πλεονεκτήματα των αρχιτεκτονικών βασισμένων στο σωρό είναι ότι τα προγράμματα λαμβάνουν πολύ λίγη μνήμη, επειδη αυτό δεν είναι απαραίτητο για να διευκρινίσει που βρίσκονται η πηγή και ο προορισμός της λειτουργίας .Κατά συνέπεια, μια οδηγία για έναν επεξεργαστή βασισμένο στο σωρό μπορεί συχνά να αντιπροσωπευθεί σε 1 byte που διευκρινίζει τη λειτουργία, αν και αυτό μπορεί να ποικίλει ανάλογα με το πόσες διαδικασίες ο επεξεργαστής υποστηρίζει.Η εξαίρεση σε αυτό είναι ότι οι οδηγίες PUSH μπορουν να πάρουν 24 bit ή περισσότερα (8 bit για να διευκρινίσει τη λειτουργία και 16 bit της σταθεράς) για να κωδικοποιησούν.



ΣΧΗΜΑ2-10 Stack implementation using memory and registers

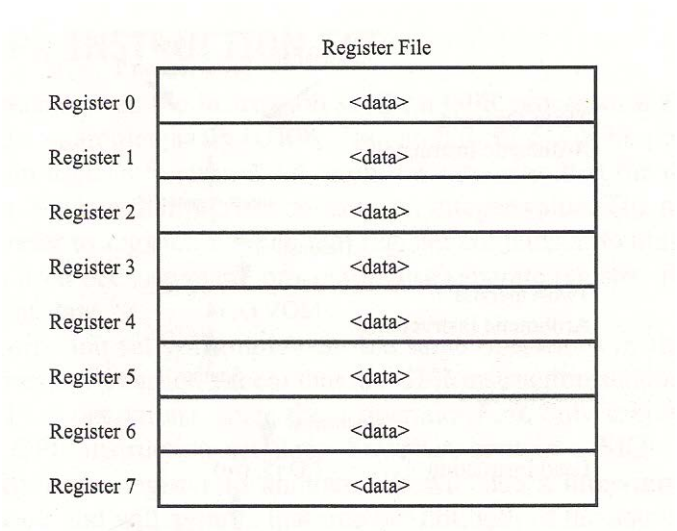


ΣΧΗΜΑ2-11 stack-based instruction execution

2.5 Αρχιτεκτονικές καταχωρητή γενικής χρήσης

Σε μια αρχιτεκτονική καταχωρητη γενικης χρήσης(GPR), οι οδηγίες διαβάζουν τους τελεστέους τους και γράφουν τα αποτελέσματά τους σε ένα τυχαίο αρχείο καταλόγων πρόσβασης, παρόμοιο με αυτό που διευκρινίζεται στο σχήμα 2-12. Το αρχείο καταλόγων καταχωρητη γενικης χρήσης επιτρέπει σε μια οδηγία να έχει πρόσβαση στους καταλόγους σε οποιαδήποτε διαταγη με τη διευκρίνιση του αριθμού (επίσης αποκαλειται ID καταλόγων) του καταλόγου που προσεγγίζεται, σαν τη μνήμη συστηματος που επιτρέπει στις διευθύνσεις στη μνήμη να προσεγγισουν οποιαδήποτε διαταγή.Μια σημαντική διαφορά ανάμεσα σε ένα αρχείο καταχωρητή γενικής χρήσης και σε ένα σωρό είναι ότι διαβάζοντας τα περιεχόμενα του καταχωρητή γενικής χρήσης δεν αλλάζουν , αντίθετα απόμακρυνουν τις τιμές από το σωρό.Διαδοχικά διαβάσματα ένος αρχείου καταχωρητη γενικης χρησης χωρίς να επεμβει με γράψιμο θα επιστρέψει το ίδιο αποτέλεσμα , ενώ ο διαδοχικός σωρός θα επιστρέψει το περιεχόμενο του σωρού στη διαταγή LIFO.

Πολλές αρχιτεκτονικές GPR ορίζουν ειδικές έννοιες σε μερικους από τους καταλόγους στο αρχείο καταλόγων για να καταστήσουν τον προγραμματισμό ευκολότερο.Για παραδείγμα, κάποιοι καλωδιομενοι επεξεργαστες καταγράφουν το 0(r0) με την τιμή 0 για να καταστήσει αυτο ευκολότερο για να παραγάγει αυτήν την κοινή σταθερά, και άλλα καθιστούν το μετρητή προγράμματος ορατό ως έναν από τους καταλόγους.Οι έννοιες προσδιορίζονται από το υλικό και δεν μπορούν αλλαχθουν από τα προγράμματα.



ΣΧΗΜΑ2-12 general-purpose register file

2.6 Συγκρίνοντας τις αρχιτεκτονικές καταχωρητή γενικής χρήσης και τις stack based αρχιτεκτονικες

Οι αρχιτεκτονικές stack based και οι αρχιτεκτονικές καταχωρητη γενικης χρησης διαφέρουν κυρίως στις διεπαφές τους των αρχείων καταλόγων τους.Στις αρχιτεκτονικές stack based, τα δεδομένα αποθηκεύονται σε έναν σωρό στη μνήμη.Το αρχείο του επεξεργαστή μπορεί να χρησιμοποιηθεί για να εφαρμόσει την κορυφαία μερίδα του σωρού για να επιτρέψει τη γρηγορότερη πρόσβαση στο σωρό.

Στις αρχιτεκτονικές GPR, το αρχείο καταλόγων είναι μια άμμεσα προσπελασιμη συσκευη όπου κάθε κατάλογος μπορεί να διαβαστεί ανεξάρτητα και να γραφτεί από τον επεξεργαστή.Σε αυτές τις αρχιτεκτονικές, το αρχείο καταλόγων και η μνήμη είναι απολύτως ανεξάρτητα, και τα προγράμματα είναι αρμόδια για να κινήσουν τα στοιχεία μεταξύ αυτών των δύο τύπων αποθηκεύσεων ανάλογα με τις ανάγκες.

Οι αρχιτεκτονικές stack based χρησιμοποιήθηκαν σε μερικά πρόωρα συγκροτήματα ηλεκτρονικών υπολογιστών για δύο λόγους.Κατ' αρχάς, επειδή οι τελεστέοι και ο προορισμός μιας οδηγίας σε μια αρχιτεκτονική βασισμένη σωρο είναι υπονοούμενοι, οι οδηγίες παίρνουν λιγότερα bits για να κωδικοποιησουν από ότι αυτες κάνουν στις αρχιτεκτονικές καταχωρητη γενικής χρησης .Αυτό μείωσε το ποσό μνήμης που απορροφήθηκε από τα προγράμματα, το οποίο ήταν ένα σημαντικό ζήτημα σε πρόωρες μηχανές.Δεύτερον, οι αρχιτεκτονικές stack based διαχειρίζονται τους καταλόγους αυτόματα, ελευθερώνοντας τους προγραμματιστές από την ανάγκη να αποφασιστεί ποια στοιχεία πρέπει να κρατηθούν στο αρχείο καταλόγων.

Ένα άλλο πλεονέκτημα είναι ότι το σύνολο οδηγίων δεν αλλάζει εάν το μέγεθος του αρχείου καταλόγων αλλάζει.Αυτό σημαίνει ότι τα προγράμματα που γράφονται για έναν stack based επεξεργαστή μπορούν να οργανωθούν στις μελλοντικές εκδόσεις του επεξεργαστή που έχουν περισσότερους καταλόγους, και θα δουν βελτιώσεις στην απόδοση επειδή είναι επίσης πολύ εύκολο να συνταχτούν --- τόσο εύκολος που μερικοί μεταγλωττιστές επεξεργάζονται ακόμα και συντάσσοντας για έναν επεξεργαστή GPR.

Οι αρχιτεκτονικές καταχωρητη γενικης χρησης έχουν γίνει κυρίαρχα στα πιό πρόσφατα έτη λόγω των βελτιώσεων στην τεχνολογία και στην αλλαγη στις γλώσσες υψηλου επιπέδου.Δεδομένου ότι οι ικανότητες μνήμης έχουν αυξηθεί και οι δαπάνες μνήμης έχουν μειωθεί, το ποσό του διαστήματος που απορροφείται από ένα πρόγραμμα έχει γίνει λιγότερο σημαντικό.Σημαντικότερα, οι μεταγλωττιστές για τις GPR αρχιτεκτονικες είναι γενικά ικανοί να επιτύχουν την καλύτερη απόδοση με έναν δεδομένο αριθμό καταχωρητη γενικης χρησης από ότι οι αρχιτεκτονικές βασισμένες στο σωρό με τον ίδιο αριθμό καταλόγων επειδή ο μεταγλωττιστής επιλέγει ποιες τιμές να κρατήσουν στον κατάλογο σε οποιοδήποτε χρόνο και μπορούν να χρησιμοποιήσουν οποιαδήποτε αξία στο αρχείο καταλόγων ως εισοδο σε οποιαδήποτε οδηγία.

Λόγω του πλεονεκτήματος απόδοσής τους και της μειωμένης σημασίας του μεγέθους κώδικα, ουσιαστικά όλοι οι πρόσφατοι επεξεργαστές τερματικών σταθμών είναι αρχιτεκτονικές GPR.Οι αρχιτεκτονικές βασισμένες στον σωρό είναι κάπως ελκυστικότερες για τα ενσωματωμένα συστήματα, στα οποία το χαμηλότερο κόστος και οι χαμηλές απαιτήσεις κατανάλωσης ισχύος περιορίζουν συχνά το ποσό μνήμης που μπορεί να περιληφθεί σε ένα σύστημα,φτιαχνοντας το μέγεθος του κώδικα .

2.7 Χρησιμοποιήση των σωρών για εφαρμογή των κλήσεων διαδικασίας

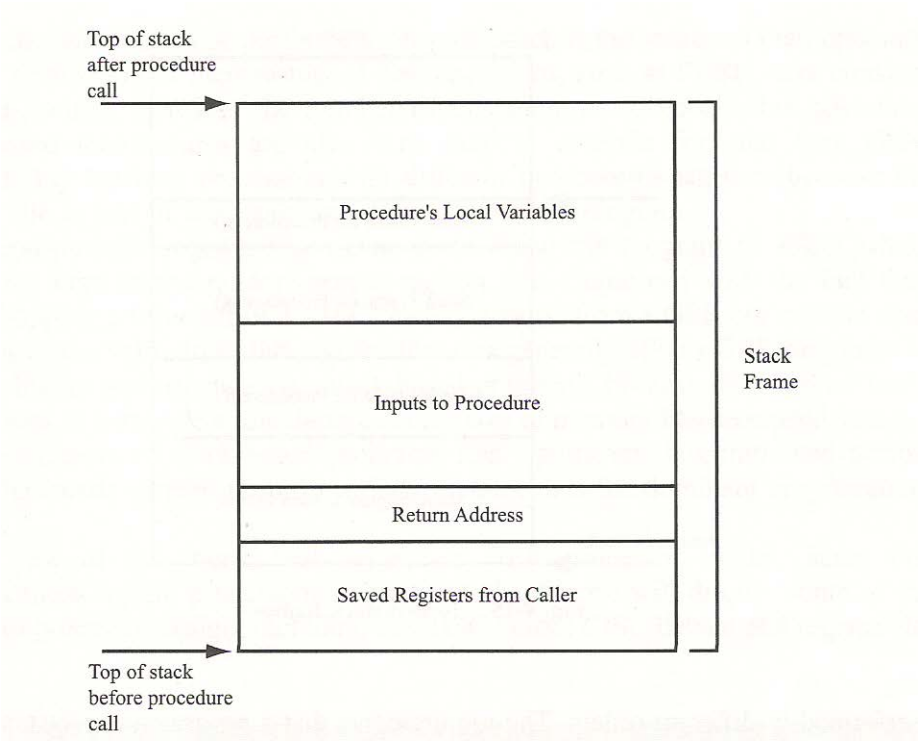
Οι κλήσεις διαδικασίας είναι σημαντικό μέρος ουσιαστικά όλων των γλωσσών υπολογιστών.Επιτρέπουν συνήθως στις χρησιμοποιημένες λειτουργίες να γραφτούν μιά φορά και να χρησιμοποιηθούν όποτε απαιτούνται, και παρέχουν την αφαίρεση, που καθιστά ευκολότερο για τους πολλαπλους ανθρώπους να συνεργαστούν σε ένα πρόγραμμα.Εντούτοις, διάφορες δυσκολίες περιλαμβάνονται στην εφαρμογή των κλήσεων διαδικασίας :

1.Τα προγράμματα χρειάζονται έναν τρόπο για να περασούν τις εισοδους στη διαδικασία που καλούν και να παραληφθούν τα αποτελέσματα πίσω από αυτα.

2.Οι διαδικασίες πρέπει να είναι σε θέση να διαθέσουν το διάστημα στη μνήμη για τις τοπικές μεταβλητές χωρίς οποιαδήποτε στοιχεία που χρησιμοποιούνται από το πρόγραμμά που τους καλεί.

3.Δεδομένου ότι οι διαδικασίες μπορούν να κληθούν από πολλές διαφορετικές θέσεις στα πλαίσια ενός προγράμματος και συντάσσονται συχνά χωριστά από το πρόγραμμα που τις καλεί, είναι γενικά αδύνατο να καθοριστεί ποιοι κατάλογοι μπορούν να χρησιμοποιηθούν ακίνδυνα από μια διαδικασία και ποιοι περιέχουν τα στοιχεία που θα απαιτηθούν αφότου ολοκληρώθει η διαδικασία.

4.Οι διαδικασίες χρειάζονται έναν τρόπο για να ανακαλύψουν από που κλήθηκαν έτσι ώστε η εκτέλεση να μπορεί να επιστρέψει στο καλουμενο πρόγραμμα όταν ολοκληρώθει η διαδικασία.



ΣΧΗΜΑ2-13 stack frame

Τα περισσότερα συστήματα προγραμματισμού χρησιμοποιούν μια δομή δεδομένων σωρών για να λύσουν αυτά τα προβλήματα.Στις αρχιτεκτονικές GPR, ένας σωρός εφαρμοζεται στη μνήμη, όπως αυτή που διευκρινίζεται στο σχήμα 2-9, ενώ οι αρχιτεκτονικές που βασιζονται στο σωρό μπορουν να χρησιμοποιήσουν τον κύριο σωρό του επεξεργαστή.Όταν μια διαδικασια καλείται, ένας φραγμός της μνήμης αποκαλούμενος πλαίσιο σωρών διατίθεται στο πλαίσιο σωρών.Όπως διευκρινίζεται στο σχήμα 2-13, το πλαίσιο σωρών μιας διαδικασίας περιέχει το διάστημα του περιεχόμενου του καλουμενου προγραμματος αρχείου καταλόγων, ένας δείκτης στη θέση οπου η διαδικασία πρέπει να διακλαδιστεί όταν ολοκληρώνει (η επιστροφή της διεύθυνσή του), τα επιχειρήματα εισοδου στη διαδικασία, και οι τοπικές μεταβλητές της διαδικασίας.

Όταν μια διαδικασία καλείται, το περιεχόμενο του καλουμενου προγραμματος αρχείου καταλόγων αντιγράφεται στο πλαίσιο του σωρού, μαζί με την επιστροφή θέσης του και τις εισοδους στη διαδικασία.Η διαδικασία χρησιμοποιεί έπειτα το υπόλοιπο του πλαισίου του σωρού για να κρατήσει τις τοπικές μεταβλητές της.Δεδομένου ότι ο αριθμός επιχειρημάτων εισόδου και τοπικών μεταβλητών ποικίλλει από διαδικασία σε διαδικασια , οι διαφορετικές διαδικασίες θα έχουν τα πλαίσια του σωρού των διαφορετικών μεγεθών.Η ρύθμιση των στοιχείων μέσα στο πλαίσιο του σωρού ποικίλει απο σύστημα προγραμματισμού σε σύστημα προγραμματισμού.

Όταν μια διαδικασία τελειώσει,μεταπηδά στην επιστροφή της διεύθυνσης που περιλαμβάνεται στο πλαίσιο του σωρού, και η εκτέλεση του καλούντος προγράμματος επαναλαμβάνεται.Το καλουμενο πρόγραμμα διάβαζει τα σωσμένα περιεχόμενα των αρχείων καταλόγων έξω από το πλαίσιο του σωρού και χειρίζεται το αποτέλεσμα της διαδικασίας ,που μπορεί να περάσουν είτε σε έναν κατάλογο είτε

στο σωρό.Τέλος, η κορυφή του δείκτη σωρών αποκαθίσταται στη θέση της προτού να κληθεί η διαδικασία, τοποθεντωντας το πλαίσιο του σωρού μακριά του σωρού.

Όταν ένα πρόγραμμα κάνει τις τοποθετημένες κλήσεις διαδικασίας (διαδικασίες που καλούν άλλες διαδικασίες), κάθε τοποθετημένη διαδικασία διαθέτει το πλαίσιο του σωρού της πάνω από εκείνους αυτά που είναι ήδη στο σωρό.Παραδείγματος χάριν, το σχήμα 2-14 παρουσιάζει το περιεχόμενο του σωρού κατά τη διάρκεια της εκτέλεσης της διαδικασίας h(), η οποία κλήθηκε από μέσα από τη διαδικασία g() . Η διαδικασία g() κλήθηκε μέσα από την f(), η οποία κλήθηκε από το κύριο πρόγραμμα.Εφ' όσον δεν ξεχειλίζει ο σωρός, οι κλήσεις διαδικασίας μπορούν να τοποθετηθούν τόσο βαθειά ανάλογα με τις ανάγκες και κάθε πλαίσιο του σωρού θα τοποθετηθει μακρια του σωρού όταν η εκτέλεση επιστρέψει στο καλουμενο πρόγραμμά της.

ΚΕΦΑΛΑΙΟ 3^{ο}

ΣΧΕΔΙΑΣΗ ΕΠΕΞΕΡΓΑΣΤΗ

3.1 Στόχοι

Αυτό το κεφάλαιο παρέχει μια εισαγωγή στο θέμα σχεδίασης του επεξεργαστή αναλύοντας ορισμένες έννοιες που έχουμε ήδη παρουσιάσει γενικά στα προηγούμενα κεφάλαια και προετοιμάζοντας το έδαφος για τα επόμενα δύο κεφάλαια, τα οποία θα παρουσιάσουν τεχνικές γα βελτίωση των επιδόσεων του επεξεργαστή.

3.2 Εισαγωγή

Το κεφάλαιο αυτό είναι το πρώτο από τρία κεφάλαια που ασχολούνται με τη σχεδίαση επεξεργαστή μέσω της τεχνικής μετασχηματισμού της αρχιτεκτονικής των σετ οδηγιών, όπως επίσης και με τις βασικές αρχές της μικροαρχιτεκτονικής επαξεργαστών. Το επόμενο κεφάλαιο εξηγεί μια τεχνική που αυξάνει την επίδοση του επεξεργαστή προχωρώντας σε εκτέλεση πολλαπλών οδηγιών (pipelining).Το κεφάλαιο 7 ολοκληρώνει την συζήτηση αυτή για τον επεξεργαστή επεξηγώντας τον (instruction level parallelism) Παραλληλισμό σε επίπεδο εντολών

Η σχεδίαση επεξεργαστή έχει διαιρεθεί σε δύο υποκατηγορίες: την αρχιτεκτονική των σετ οδηγιών και τη μικροαρχιτεκτονική του επεξεργαστή. Η πρώτη αναφέρεται στο σχεδιασμό σετ εργασιών που εκτελεί ο επεξεργαστής και περιλαμβάνει την επιλογή του μοντέλου προγραμματισμού, τον αριθμό εγγραφών και αποφάσεις για τον τρόπο προσπέλασης των πληροφοριών. Η μικροαρχιτεκτονική του επεξεργαστή περιγράφει πως οι οδηγίες θέτονται σε εφαρμογή και περιλαμβάνει παράγοντες όπως πόσος χρόνος χρειάζεται για την εκτέλεση των οδηγιών, πόσες οδηγίες μπορούν να εκτελούνται ταυτόχρονα και πως σχεδιάζονται υπομονάδες του επεξεργαστή όπως ο register file.Αυτοί οι ορισμοί μπορεί να είναι λίγο ασαφείς καθώς υπάρχει επικάλυψη των δύο αυτών μεθόδων σε τέτοιο σημείο που είναι δύσκολο να καθοριστεί ποια κατηγορία αρχιτεκτονικής χρησιμοποιείται κάθε φορά. Ένας ικανοποιητικός ορισμός είναι ότι ο,τιδήποτε προγραμματίζεται σε assembly είναι μέρος της αρχιτεκτονικής με σετ οδηγιών, ενώ ο,τιδήποτε αφορά την επίδοση είναι μέρος της μικροαρχιτεκτονικής του επεξεργαστή.

3.3 Αρχιτεκτονική των σετ οδηγιών

Όταν ο προγραμματισμός του υπολογιστή γινόταν κυρίως σε γλώσσα assembly, το σετ οδηγιών θεωρούταν το πιο σημαντικό κομμάτι της αρχιτεκτονικής υπολογιστών, αφού καθόριζε το πόσο εύκολη ή δύσκολη ήταν επίτευξη καλών. επιδόσεων του συστήματος. Με το πέρασμα των χρόνων όμως έχασε την σημαντικότητά της για διάφορους λόγους. Καταρχήν ο προγραμματισμός άρχισε να γίνεται σε γλώσσες υψηλότερου επιπέδου. Έτσι ο προγραμματιστής δεν ασχολείται πλέον με το σετ οδηγιών. Κατά δεύτερον και κυριότερο οι καταναλωτές απαιτούν συμβατικότητα ανάμεσα σε υπολογιστές διαφορετικής γενιάς, πράγμα που σημαίνει ότι περιμένουν

από τα προγράμματα που έτρεχαν στον παλιό τους υπολογιστή, να τρέχουν με τον ίδιο τρόπο και στον καινούριο. Αυτό έχει ως αποτέλεσμα το σετ οδηγιών των επεξεργαστών των δύο αυτών συστημάτων να παραμένει το ίδιο , ίσως με κάποιες βελτιώσεις. Γιαυτό και πλέον για την αύξηση των επιδόσεων δίνεται μεγαλύτερη βάση στη βελτίωση της αρχιτεκτονικής του επεξεργαστή.

Στο προηγούμενο κεφάλαιο καλύφθηκε ένα πολύ σημαντικό θέμα του σχεδιασμού σετ οδηγιών (ISA), η επιλογή μοντέλου προγραμματισμού. Όπως είδαμε το μοντέλο GPR είναι το επικρατέστερο και με αυτό θα ασχολούμαστε στο υπόλοιπο του βιβλίου. Στο κεφάλαιο αυτό θα καλύψουμε 4 από τα σημαντικότερα θέματα που αφορούν τα σετ οδηγιών. Τη διαμάχη μεταξύ RISC και CISC, την επιλογή μεθόδου διευθυνσιοδότησης (addressing mode), τη χρήση κωδικοποιήσεων των οδηγιών προκαθορισμένου η μεταβλητού μεγέθους και τις διανυσματικές οδηγίες πολυμέσων.

3.3.1. RISC ENANTION CISC

Πριν τη δεκαετία του 1980 γινόταν μια προσπάθεια να μειωθεί το «εννοιολογικό κενό» ανάμεσα στις γλώσσες που χρησιμοποιούνταν για τον προγραμματισμό και τις γλώσσες μηχανής. Υπήρχε η αντίληψη ότι αν οι γλώσσες μηχανής γίνονταν σαν τις γλώσσες προγραμματισμού θα επιτυγχανόταν καλύτερη επίδοση του συστήματος. Δηλ αν μειωνόταν ο αριθμός των οδηγιών για τη εκτέλεση ενός προγράμματος θα ήταν ευκολότερο να μεταγλωττιστεί η κάθε γλώσσα προγραμματισμού σε γλώσσα μηχανής. Το αποτέλεσμα αυτού του σκεπτικού ήταν η σχεδίαση τέτοιων σετ που περιείχαν περίπλοκες οδηγίες.

Καθώς όμως η τεχνολογία των μεταγλωττιστών βελτιώθηκε οι ερευνητές άρχισαν να σκέφτονται σοβαρά την πιθανότητα βελτίωσης του συστήματος εάν οι ως τότε περίπλοκες οδηγίες (complex instruction set - CISC) μετατρέπονταν σε άλλες απλούστερες.(redused instruction set - RISK).

Το πρωταρχικό επιχείρημα υπέρ των CISC υπολογιστών ήταν ότι χρησιμοποιούν εν γένει λιγότερες οδηγίες για να κάνουν υπολογισμούς άρα θα ήταν και γρηγορότεροι. Επιπλέον τα προγράμματα που ήταν γραμμένα για CISC υπολογιστές καταλάμβανα λιγότερο χώρο στη μνήμη από τα αντίστοιχα προγράμματα για RISC. Από την άλλη πλευρά όμως οι RISC υπολογιστές λόγω των απλούστερων εντολών μπορούσαν να εκτελέσουν περισσότερες οδηγίες στον ίδιο χρόνο που οι CISC θα εκτελούσαν μία. Επίσης αν στον επεξεργαστή RISC αυξανόταν το clock rate και μπορούσε να εκτελέσει τα προγράμματα σε λιγότερο χρόνο από το τον αντίστοιχο που θα χρειαζόταν ο CISC επεξεργαστής(παρόλο που θα απαιτούσε λιγότερες εντολές).Έτσι ο RISC επεξεργαστής είναι πιο αποδοτικός.

Τη δεκαετία του 1980 και στις αρχές του 90 υπήρχε αμφισβήτηση για το ποια από τις δύο μεθόδους είναι καλύτερη γιατί αναλόγως την οπτική γωνία κάθε μέθοδος είχε τα δικά της πλεονεκτήματα. Παρόλα αυτά η πλειοψηφία των σχεδιασμών σετ οδηγιών (ISA) γινόταν σε RISC επεξεργαστές αν και η Intel x86(IA-32) χρησιμοποιούσε CISC επεξεργαστές πιστεύοντας ότι είναι καλύτεροι.

Στα τελευταία 20 χρόνια η αρχιτεκτονική των επεξεργαστών αυτών συνέκλινε σε τέτοιο βαθμό που ήταν δύσκολο να διακρίνεις εάν επρόκειτο για CISC ή RISC.
Οι RISC επεξεργαστές ενσωμάτωσαν τα πιο χρήσιμα χαρακτηριστικά των CISC, κυρίως αυτά που δεν επηρέαζαν τον κύκλο του ρολογιού ενώ αντίστοιχα οι CISC επεξεργαστές απέβαλλαν τις περίπλοκες οδηγίες που δεν ήταν και τόσο λειτουργικές.
Μια πιο ξεκάθαρη απεικόνιση των μεθόδων αρχιτεκτονικής είναι ότι η μέθοδος RISC είναι αρχιτεκτονικής load-store που σημαίνει ότι μόνο οδηγίες αποθηκευμένες μπορούν να έχουν πρόσβαση στη μνήμη. Η αρχιτεκτονική GPR που περιγράφεται στο κεφάλαιο 4 είναι load-store αρχιτεκτονική.

Στις αρχιτεκτονικές CISC οι αριθμητικές αλλά και άλλες οδηγίες διαβάζουν τις οδηγίες ή και γράφουν τα αποτελέσματα των οδηγιών μέσω της μνήμης και δε χρησιμοποιούν register file.Για παράδειγμα η αρχιτεκτονική CISC επιτρέπει στην οδηγία ADD της μορφής ADD (r1),(r2),(r3) .Οι παρενθέσεις δηλώνουν ότι η εγγραφή αυτή περιέχει κάποια διεύθυνση στη μνήμη και εκεί θα βρει ο τελεστής το αποτέλεσμα, ή εκεί τα το τοποθετήσει.. Με τη χρήση αυτής της σημειογραφίας η εντολή ADD (r1),(r2),(r3) λέει στον επεξεργαστή να προσθέσει την τιμή που περιέχεται στη διεύθυνση r2 με την τιμή στην r3 και να αποθηκεύσει τα αποτέλεσμα στην διεύθυνση r1.

Η διαφορά μεταξύ των αρχιτεκτονικών load-store με τις αρχιτεκτονικές που συγχωνεύουν παραπομπές σε μνήμη με άλλες λειτουργίες, είναι ένα έξοχο παράδειγμα των ανταλλαγών που έγινα μεταξύ των RISC και CISC αρχιτεκτονικών. Επειδή οι RISC αρχιτεκτονική υλοποιείται με μοντέλα load-store, ένας RISC επεξεργαστής θα χρειάζονταν αρκετές εντολές για να εκτελέσει μια απλή ADD εντολή για την οποία ο CISC επεξεργαστής θα χρειαζόταν μόνο μια εντολή. Παρόλα αυτά το υλικό που απαιτεί ο CISC επεξεργαστής είναι πολυπλοκότερο αφού χρησιμοποιεί τελεστές από τη μνήμη. Ετσι ως αποτέλεσμα απαιτεί περισσότερο χρόνο από ένα RISC επεξεργαστή.

3.3.2. ADDRESSING MODES --Μέθοδοι διευθυνσιοδότησης

Όπως είδαμε μια από τις μεγαλύτερες διαφορές μεταξύ RISC και CISC αρχιτεκτονικής είναι το σετ οδηγιών που έχει τη δυνατότητα να προσπελάσει τη μνήμη. Ένα άλλο σχετικό θέμα που επηρεάζει και τους δύο τύπους αρχιτεκτονικής είναι ποια μέθοδο διευθυνσιοδότησης υποστηρίζει κάθε αρχιτεκτονική. Στην αρχιτεκτονική υπολογιστών με τον όρο μέθοδο διευθυνσιοδότησης περιγράφουμε το σύνολο τον συντακτικών και των μεθόδων που χρησιμοποιούνται από τις εντολές για να καθορίσουν μια διεύθυνση μνήμης, είτε αν πρόκειται για τελική διεύθυνση είτε για αναφορά σε διεύθυνση μνήμης ή ακόμα και για μια διεύθυνση στην οποία θα μεταβεί ολόκληρος κλάδος

Ανάλογα με τον τύπο αρχιτεκτονικής κάποιοι από τις μεθόδους διευθυνσιοδότησης είναι διαθέσιμοι σε όλες ή μερικές από τις οδηγίες που αναφέρονται στη μνήμη. Στην περίπτωση που είναι διαθέσιμες σε οποιαδήποτε οδηγία/εντολή η αρχιτεκτονική χαρακτηρίζεται *ορθογώνια.* Αυτό συμβαίνει γιατί η μέθοδος διευθυνσιοδότησης είναι ανεξάρτητο από τον τύπο οδηγιών που επιλέγουμε.

Ως τώρα έχουμε χρησιμοποιήσει δύο μεθόδους διευθυνσιοδότησης: την διευθυνσιοδότηση με καταχωρητή (register addressing) για τη φόρτωση εντολών την αποθήκευσή τους και τον τρόπο που αναφέρονται οι CISC οδηγίες στη μνήμη και τη διευθυνσιοδότηση ετικέτας (label addressing) που αφορά τις διακλαδισμένες οδηγίες. Στη διευθυνσιοδότηση με καταχωρητή η εντολή διαβάζει την τιμή που βρίσκεται σε ένα καταχωρητή (register) και έπειτα τη χρησιμοποιεί ως διεύθυνση μνήμης ή και σημείο διακλάδωσης. Για να δείξουμε ότι χρησιμοποιούμε αυτή τη μέθοδο χρησιμοποιούμε το σημείο (rx). Ένα σετ οδηγιών μπορεί να εκτελεστεί χρησιμοποιώντας μόνο αυτή τη μέθοδο γιατί κάθε διεύθυνση μπορεί να υπολογιστεί με αριθμητικούς υπολογισμούς και μετά να αποθηκευτεί σε ένα καταχωρητή. Όμως ο επεξεργαστής παρέχει και άλλες μεθόδους που μειώνουν τον αριθμό των οδηγιών και άρα βελτιώνουν την απόδοση του συστήματος.

Η δεύτερος μέθοδος που είδαμε ως τώρα είναι η διευθυνσιοδότηση ετικέτας.Με αυτή τη μέθοδο έχουμε ένα κλάδο εντολών που καθορίζουν τον προορισμό τους βάζοντας μια λέξη ετικέτα. Η «ετικέτα» αποθηκεύεται ως εντολή κάπου αλλού στ πρόγραμμα. Όπως είδαμε στο προηγούμενο κεφάλαιο αυτές οι «ετικέτες» δεν εμφανίζονται όταν το πρόγραμμα μεταφραστεί σε γλώσσα μηχανής. Στην πραγματικότητα οι περισσότερες διακλαδισμένες οδηγίες δεν περιέχουν καν τις διευθύνσεις προορισμού τους. Ο συμβολομεταφραστής ή ο διασυνδετής μεταφράζει την ετικέτα σε offset (αντιστάθμισμα-μετατοπιστής) από το σημείο της διακλάδωσης ως τη διεύθυνση στόχο. Το offset μπορεί να είναι είτε αρνητικό είτε θετικό. Πρακτικά ο κλάδος εντολών λέει στον επεξεργαστή πόσο μακριά βρίσκεται αποθηκευμένη η εντολή-ετικέτα, και όχι που ακριβώς βρίσκεται. Για να βρει τη διεύθυνση ο επεξεργαστής προσθέτει το offset με το PC του κλάδου εντολών.

Η χρήση offsets αντί για τη σαφή διεύθυνση της ετικέτας έχει δύο πλεονεκτήματα. Καταρχήν μειώνει τον αριθμό των bits που χρειάζονται για την κωδικοποίηση της εντολής / οδηγίας. Οι περισσότεροι κλάδοι τοποθετούν σχετικά κοντά τις διευθύνσεις στόχους και έτσι μειώνεται ο αριθμός των bits του offset.. Αν ο στόχος τους είναι μακριά η διεύθυνση μπορεί αν υπολογιστή με τη χρήση άλλων εντολών και κάποιου καταχωρητή. Κατά δεύτερον, το άλλο πλεονέκτημα είναι ότι ο φορτωτής (loader) μπορεί αν τοποθετήσει το πρόγραμμα σε διαφορετικές θέσεις στη μνήμη χωρίς να χρειαστεί να αλλάξει κάτι. Αν χρησιμοποιούσαμε της σαφή διεύθυνση, η διεύθυνση προορισμού κάθε κλάδου θα έπρεπε να ξανά υπολογίζεται κάθε φορά που φορτωνόταν το πρόγραμμα. Αυτό το χαρακτηριστικό είναι πολύ χρήσιμο στις βιβλιοθήκες των λειτουργικών προγραμμάτων που είναι δυναμικά συνδεδεμένες με το πρόγραμμα την ώρα εκτέλεσης, γιατί δεν έχουν τη δυνατότητα να προβλέψουν σε ποιες εντολές θα φορτωθούν.

3.3.3.Σταθερού μήκους vs μεταβλητού μήκους εντολές κωδικοποίησης.

Μόλις επιλεγεί το σε οδηγιών που θα υποστηρίζει ο επεξεργαστής ο αρχιτέκτονας υπολογιστών πρέπει να επιλέξει κάποια κωδικοποίηση για τι ISA, που είναι το σετ των bits που θα χρησιμοποιείται για να απεικονιστούν οι οδηγίες στη μνήμη. Γενικά οι αρχιτέκτονες θέλουν να βρουν μια κωδικοποίηση που είναι ταυτόχρονα πλήρης αλλά και αποκωδικοποιείται εύκολα, δηλαδή να είναι εύκολο για τον επεξεργαστή να καταλάβει ποια από τις εντολές αναπαριστάται, σύμφωνα βέβαια με ένα σχέδιο bits του προγράμματος. Δυστυχώς αυτοί τα δύο πράγματα είναι λίγο ασύμβατα.

Οι εντολές σταθερού μήκους χρησιμοποιούν τον ίδιο αριθμό bits για να κωδικοποιήσουν κάθε εντολή στο ISA.Έχουν το πλεονέκτημα ότι είναι εύκολο να αποκωδικοποιηθούν εύκολα και χωρίς καθυστερήσεις. Ακόμα ο επεξεργαστής που χρησιμοποιεί κωδικοποίηση ISA σταθερού μήκους μπορεί εύκολα να προβλέψει ποια οδηγία θα εκτελεστεί μετά, αν υποθέσουμε ότι η τρέχουσα εντολή δεν διακλαδίζεται. Έτσι είναι ευκολότερο για τον επεξεργαστή να χρησιμοποιήσει την τεχνική pipelining, το αντικείμενο του επόμενου κεφαλαίου, για να βελτιώσει τη απόδοση επικαλύπτοντας την εκτέλεση πολλών εντολών.

Οι τα σετ εντολών κωδικοποίησης μεταβλητού μήκους χρησιμοποιούν διαφορετικό αριθμό bits για να κωδικοποιήσου τις εντολές στον ISA, αναλόγως με τον αριθμό των εντολών τα addressing modes που χρησιμοποιούνται και άλλους παράγοντες. Με τη χρήση τους κάθε εντολή καταλαμβάνει όσο χρόνο πρέπει στη μνήμη, παρόλο που πολλά συστήματα απαιτούν όλες οι εντολές να έχουν μέγεθος κάποια bytes.Επίσης οι εντολές μεταβλητού μεγέθους μειώνουν και το χώρο που καταλαμβάνει ένα πρόγραμμα όμως είναι πιο δύσκολο να αποκωδικοποιηθούν, αφού κομμάτια των εντολών όπως οι τελεστές εισαγωγής μπορούν να αποθηκεύονται σε διαφορετική θέση και με διαφορετικό μέγεθος. Ακόμα το hardware δεν μπορεί να προβλέψει τη θέση της επόμενης οδηγίας με ωσότου η τρέχουσα η εντολή να έχει αποκωδικοποιηθεί αρκετά ώστε να καταλάβει το μέγεθος της.

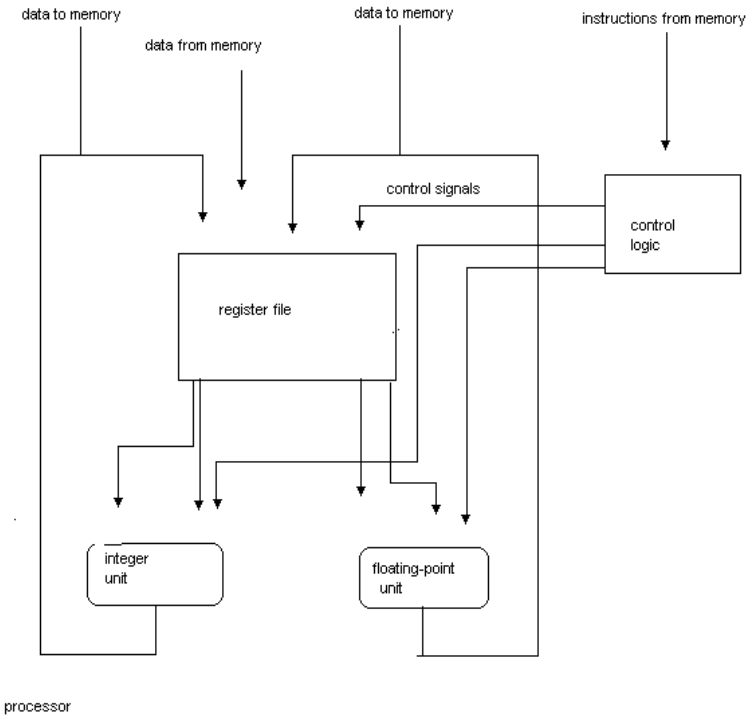
Δεδομένων των θετικών και των αρνητικών κάθε περίπτωσης, οι σταθερού μεγέθους εντολές είναι πιο κοινές στην αρχιτεκτονική σήμερα. Οι μεταβλητού μεγέθους χρησιμοποιούνται κυρίως σε αρχιτεκτονικές με μεγάλη διαφοροποίηση ανάμεσα στη μεγαλύτερη εντολή του ISA και το μέσο όρο. Τέτοια παραδείγματα περιλαμβάνουν stack based (βασισμένες στο σωρό;;;) αρχιτεκτονικές, στις οποίες πολλές λειτουργίες δε χρειάζεται να διευκρινίζουν ποιες είναι οι είσοδοί τους, και τις CISC αρχιτεκτονικές που συχνά περιέχουν κάποιες εντολές που ενέχεται να χρειαστούν μεγάλο αριθμό εισόδων.

Εδώ τελειώνει η συζήτησή μας για την αρχιτεκτονική των σετ οδηγιών. Καλύψαμε τη διαμάχη RISC εναντίον CISC, που είναι από τις δημοφιλέστερες στην αρχιτεκτονική υπολογιστών, τη σημασία των addressing modes στον ISA του επεξεργαστή, και τα θετικά και αρνητικά των μεταβλητών κωδικοποίησης σταθερού και μεταβλητού μήκους. Ακόμα κάναμε μια μικρή εισαγωγή στις διανυσματικές εντολές πολυμέσων, μια σύγχρονη επέκταση του ISA του επεξεργαστή που βελτιώνει τις επιδόσεις σε εφαρμογές παράλληλων δεδομένων.

Το υπόλοιπο του κεφαλαίου αυτού θα ασχοληθεί με τη συζήτηση για την μικροαρχιτεκτονική του επεξεργαστή, η οποία θα συνεχιστεί και στα επόμενα δύο κεφάλαια.Θα ξεκινήσουμε μια εκ βαθέων συζήτηση πάνω στον τρόπο με τον οποίο οι επεξεργαστές εκτελούν εντολές, περισσότερο από όσο έχουμε δει το θέμα ως τώρα και θα συνεχίσουμε με μια συζήτηση για το σχεδιασμό των αρχείων εγγραφής. Για λόγους ευκολίας , στο εξής, θα υποθέσουμε ότι έχουμε ένα RISC επεξεργαστή, παρόλο που οι έννοιες που θα καλύψουμε είναι εφαρμόσιμες και σε CISC αρχιτεκτονικές.

3.4.Μικροαρχιτεκτονική επεξεργαστή

Όπως περιγράψαμε και νωρίτερα η μικροαρχιτεκτονική επεξεργαστή συμπεριλαμβάνει όλες τις λεπτομέρειες σχετικά με το πώς υλοποιείται ο επεξεργαστής. Ο ISA καθορίζει πως ο επεξεργαστής προγραμματίζεται και η μικροαρχιτεκτονική πως κατασκευάζεται. Προφανώς ο ISA έχει μεγάλη επίδραση στην αρχιτεκτονική. Ένα ISA που περιέχει μόνο απλές εντολές υλοποιείται με τη χρήση ενός απλής μικροαρχιτεκτονικής ενώ αν περιέχει περίπλοκες λειτουργίες θα χρειαστεί και περίπλοκη μικροαρχιτεκτονική.
Το σχεδιάγραμμα χωρίζει τον επεξεργαστή σε τρία κύρια υποσυστήματα. Της μονάδες εκτέλεσης, τα αρχεία εγγραφής/καταχωρητές και την control logic. Τα δύο πρώτα μαζί συχνά περιγράφονται και ως μονοπάτι δεδομένων (datapath) του επεξεργαστή, αφού τα δεδομένα και οι εντολές ρέουν ανάμεσά τους τακτικά. Η control logic είναι πιο ακανόνιστη και καθορίζεται από τον επεξεργαστή.

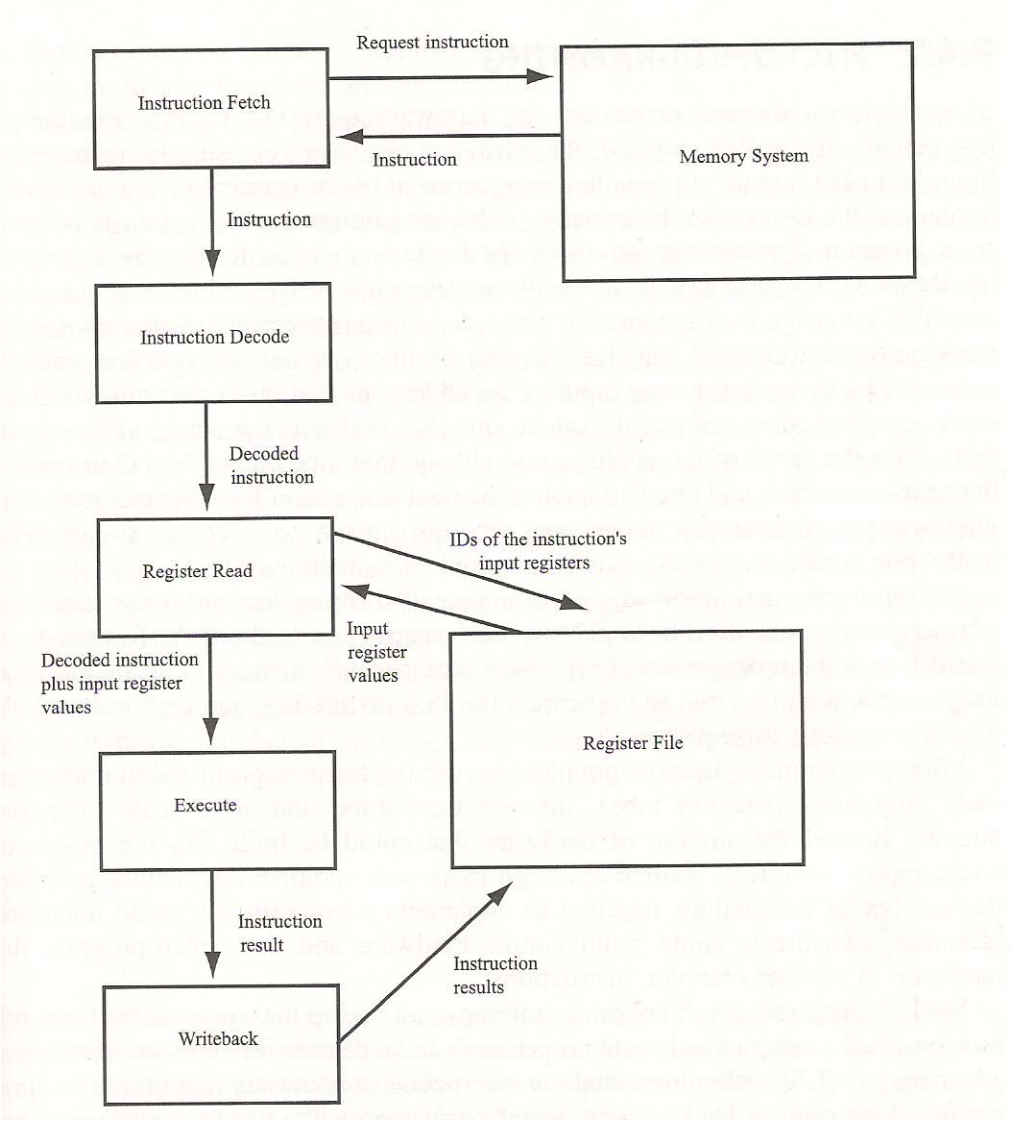


ΣΧΗΜΑ3-1 processor block diagram

3.4.1. Μονάδες εκτέλεσης

Το σχήμα 3-2 δείχνει τα βήματα που συνεπάγονται για την εκτέλεση μια εντολής και πως οι διαφορετικές μονάδες του επεξεργαστή αλληλεπιδρούν κατά την εκτέλεσή της.. Καταρχήν ο επεξεργαστής φέρνει την εντολή από τη μνήμη Η εντολή έπειτα αποκωδικοποιείται για να αποφασιστεί και τι καταχωρητές εισόδου και εξόδου έχει. Η αποκωδικοποιημένη εντολή αναπαριστάται από ένα σετ σχεδίων bits που

υποδεικνύουν στο hardware πώς θα εκτελέσει την εντολή. Αυτά τα σχέδια των bits αποστέλλονται στο επόμενο τμήμα της μονάδας εκτέλεσης η οποία διαβάζει τις εισόδους της εντολής από το αρχείο εγγραφής. Η αποκωδικοποιημένη εντολή και οι τιμές εισόδου προωθούνται στο hardware που υπολογίζει το αποτέλεσμα που τελικά γράφεται στα αρχείο εγγραφής.



ΣΧΗΜΑ3-2 instruction execution

Οι οδηγίες / εντολές που έχουν πρόσβαση στη μνήμη έχουν μια παρόμοια ροή εκτέλεσης με την διαφορά ότι το αποτέλεσμα της μονάδας εκτέλεσης αποστέλλεται στη μνήμη είτε ως διεύθυνση μιας εργασίας φόρτωσης, είτε ως διεύθυνση και δεδομένα μιας εργασίας αποθήκευσης,. Όταν το αποτέλεσμα της φόρτωσης επιστρέφει από τη μνήμη γράφεται σε ένα καταχωρητή και παρόμοια αποθηκεύεται στον καταχωρητή και το αποτέλεσμα του υπολογισμού.

Πολλές μονάδες εκτέλεσης υλοποιούνται χρησιμοποιώντας μια φυσική δομή όμοια με αυτή του σχήματος. Οι μονάδες που υλοποιούν κάθε βήμα τοποθετούνται η μία δίπλα στην άλλα, συνδεδεμένες με μικρά καλώδια. Καθώς εκτελείται η οδηγία τα

δεδομένα ρέουν από τη μία μονάδα στην επόμενη, πραγματοποιώντας διαδοχικά τις πράξεις που πρέπει..

3.4.2. Μικροπρογραμματισμός

Σε ένα μικροπρογραμματισμένο (microprogrammed) επεξεργαστή το hardware δεν εκτελεί αμέσως τις εντολές του ISA. Αντιθέτως εκτελεί πολύ απλές μικρο-διεργασίες των οποίων η σειρά καθορίζετε από τις εντολές. Ουσιαστικά κάθε εντολή στον ISA μεταφράζεται σε ένα μικρό πρόγραμμα μικροεντολών από το hardware, με τον ίδιο τρόπο που ένας μεταγλωττιστής μεταφράζει κάθε οδηγία από γλώσσα υψηλού επιπέδου σε ακολουθία εντολών assembly. Για παράδειγμα ένας μικροπρογραμματισμένος επεξεργαστής μπορεί να μεταφράσει την εντολή ADD r11,r2,r3 σε έξι μικρο-διεργασίες: μία που διαβάζει την τιμή της r2 και τη στέλνει στη μία είσοδο του αθροιστή, μία που διαβάζει την τιμή του r3 και τη στέλνει στην άλλη είσοδο του αθροιστή, μία που κάνει την πρόσθεση, μία που γράφει το αποτέλεσμα της πρόσθεσης στο r1 και μια που προσαυξάνει το PC έτσι ώστε να προχωρήσει στην επόμενη εντολή και τέλος άλλη μία που φέρνει την επόμενη εντολή από τη μνήμη. Έκαστη από τις μικροδιεργασίες χρειάζεται περίπου ένα κύκλο επεξεργασίας για να εκτελεστεί άρα συνολικά η ADD εντολή χρειάστηκε έξι κύκλους για να ολοκληρωθεί σε αυτό το σύστημα.

Οι μικροπρογραμματισμένοι επεξεργαστές περιέχουν μια μικρή μνήμη που συγκρατεί τη σειρά των μικροεντολών που χρησιμοποιούνται για την εκτέλεση μιας οδηγίας του ISA. Για να εκτελεστεί μια οδηγία/ εντολή ο μικροπρογραμματισμένος επεξεργαστής βρίσκει από τη μνήμη το σετ των μικροεντολών που πρέπει να εκτελέσει για να υλοποιηθεί η ετολή του ISA και μετά τις εκτελεί μία μία.
Ο μικροπρογραμματισμός έγινε δημοφιλής γιατί η τεχνολογία που χρησιμοποιούσαν οι πρώτοι υπολογιστές (κενοί αγωγοί, διακριτά τρανζίστορ και μικρής κλίμακας ολοκληρωμένα κυκλωματα) περιόριζε την ποσότητα του hardware που μπορούσε να ενσωματωθεί στον επεξεργαστή. Οι αρχιτέκτονες υπολογιστών ήθελαν να σχεδιάζουν ISA με περίπλοκες οδηγίες για να μειώσουν τον αριθμό των οδηγιών των προγραμμάτων. Με το μικροπρογραμματισμό μπόρεσαν να εκτελέσουν πολύπλοκες εντολές χρησιμοποιώντας απλό hardware.

Οι σύγχρονοι επεξεργαστές τείνουν να καταργήσουν το μικροπρογραμματισμό για δύο λόγους. Καταρχήν πλέον είναι πρακτικότερο να υλοποιήσεις το ISA των επεξεργαστών απευθείας στο hardware γιατί οι πρόοδοι στην VLSI τεχνολογία κάνουν τον μικροκώδικα άχρηστο. Δεύτερον οι μικροπρογραμματισμένοι επεξεργαστές είναι πιο αργοί από τους μη μικροπρογραμματισμένους γιατί το overhead που απαιτείται για την εξαγωγή κάθε μικροεντολής είναι μεγάλο και καθυστερεί τη μνήμη.

3.4.3.Σχεδίαση αρχείων εγγραφής

Ως τώρα αναφερόμαστε στα αρχεία εγγραφής σαν συσκευή που περιέχει δεδομένα ακέραια ή δεκαδικά. Οι περισσότεροι επεξεργαστές δεν υλοποιούν με αυτό τον τρόπο τους φακέλους εγγραφών. Αντίθετα δημιουργούν άλλους αρχεία για ακεραίους και άλλα για δεκαδικούς. Οι ακέραιοι συντάσσονται ως «rx» ενώ οι δεκαδικοί ως «fx». Χρησιμοποιώντας αυτή τη σύνταξη φαίνεται καθαρά ποιο αρχείο αναφέρεται στις εντολές που μπορεί να χρειάζονται και τους δύο αυτούς τύπους. Οι αριθμητικές εντολές είναι γενικά περιορισμένες στη χρήση του κατάλληλου αρχείου, ανάλογα με τον τύπο του υπολογισμού που εκτελούν. Παρόλα αυτά μερικές αριθμητικές εντολές μπορούν να μεταφέρουν πληροφορίες από το ένα τύπο καταχωρητή στον άλλο.
Οι επεξεργαστές υλοποιούν διαφορετικά αρχεία εγγραφής για δύο λόγους. Καταρχήν τους επιτρέπουν να τοποθετηθούν κοντά στις μονάδες εκτέλεσης που τους χρειάζονται. Ένας καταχωρητής σε ακέραιο θα τοποθετηθεί κοντά σε μια μονάδα που εκτελεί ακέραιους υπολογισμούς, ενώ ένας δεκαδικός κοντά σε μονάδα που εκτελεί δεκαδικού υπολογισμού. Έτσι μειώνεται το μέγεθος των καλωδίων που συνδέουν τους καταχωρητές στις μονάδες εκτέλεσης.

Ο δεύτερος λόγος για τον οποίο οι επεξεργαστές χωρίζουν τα ακέραιας από τα δεκαδικά αρχεία εγγραφής είναι ότι καταλαμβάνουν λιγότερο χώρο στους επεξεργαστές που εκτελούν περισσότερες από μία εντολές σε κάθε κύκλο. Οι λεπτομέρειες του θέματος ξεφεύγουν από τους σκοπούς του βιβλίου αυτού αλλά θα πούμε ότι το μέγεθος του αρχείου μεγαλώνει όσο είναι το τετράγωνο των ταυτόχρονων αναγνώσεων και έγγραφών που επιτρέπει το αρχείο εγγραφής. Κατά την εκτέλεση ενός κύκλου το αρχείο εγγραφής επιτρέπει συνήθως δύο αναγνώσεις και μία εγγραφή, αφού έτσι γίνεται με τις περισσότερες λειτουργίες. Κάθε επιπρόσθετη λειτουργία που θέλει να εκτελέσει ο επεξεργαστής ανεβάζει ως και τρεις φορές τον αριθμό αναγνώσεων και εγγραφών. Με το να χωρίσουμε τα αρχεία εγγραφής σε ακέραια και δεκαδικά μειώνεται αυτός ο αριθμός αναγνώσεων και εγγραφών, Και εφόσον ο χώρος που καταλαμβάνει το αρχείο εγγραφής μεγαλώνει γρηγορότερα και μη γραμμικά σύμφωνα με τον αριθμό αυτό δύο ξεχωριστά αρχεία εγγραφής καταλαμβάνουν λιγότερο χώρο από ότι ένα με τον ίδιο αριθμό αναγνώσεων και εγγραφών.

ΚΕΦΑΛΑΙΟ 4^{ο}

ΔΙΟΧΕΤΕΥΣΗ

4.1 Στόχοι

Αυτό το κεφάλαιο καλύπτει την διοχέτευση, μια τεχνική σχετικά με την απόδοση των επεξεργαστών.Η διοχέτευση επιτρέπει σε έναν επεξεργαστή να επικαλύψει την εκτέλεση διάφορων οδηγιών έτσι ώστε περισσότερες οδηγίες να μπορούν να εκτελεσθούν στην ίδια χρονική περίοδο.

4.2 Εισαγωγή

Οι πρόωροι υπολογιστές εκτέλεσαν τις οδηγίες σε μια πολύ απλή μόδα :Ο επεξεργαστής προσκόμισε μια οδηγία από τη μνήμη, την αποκωδικοποίησε για να καθορίσει τι οδηγία ήταν, διάβασε τις εισόδους της οδηγίας από το αρχείο καταλόγων, διενέργησε τον υπολογισμό που απαιτήθηκε από την οδηγία, και έγραψε το αποτέλεσμα στο αρχείο καταλόγων.Οι οδηγίες που προσπελαυνουν την μνήμη είναι ελαφρώς διαφορετικές, αλλά κάθε οδηγία ήταν εντελώς τελειωμένη προτού να αρχίσει η εκτέλεση της επόμενης.Το πρόβλημα με αυτήν την προσέγγιση είναι ότι το υλικό που απαιτείται για να εκτελέσει κάθε ένα από αυτά τα βήματα (η ευρύτητα οδηγίας, οδηγία αποκωδικοποιήσης, δίαβασμα καταλογού , εκτέλεση οδηγίας, και γραψιμο στο κατάλογος) είναι διαφορετικό, έτσι το μεγαλύτερο μέρος του υλικού είναι άπρακτο σε οποιαδήποτε δεδομένη στιγμή, περιμένει τα άλλα μέρη του επεξεργαστή να ολοκληρώσουν το μέρος της εκτέλεσης μιας οδηγίας.

Από πολλές απόψεις, αυτό είναι παρόμοιο με το ψήσιμο διάφορων φραντζολών του ψωμιού με την κατασκευή της ζύμης για μια φραντζόλα,αφήνοντας τις φραντζολες να φουσκώσουν, ψήνοντας τη φραντζόλα, και επαναλαμβάνοντας έπειτα ολόκληρη την διαδικασία.Ενώ κάθε ένα από τα βήματα στο ψήσιμο κάθε φραντζόλας του ψωμιού πρέπει να γίνει μέσα στη διαταγή και διαρκεί ένα καθορισμένο χρονικό διάστημα, ένα άτομο θα μπορούσε να ψήσει διάφορες φραντζόλες του ψωμιού πολύ γρηγορότερα με την κατασκευή της ζύμης για τη δεύτερη φραντζόλα ενώ η ζύμη για την πρώτη φραντζόλα φουσκώνει, κατασκευάζοντας τη ζύμη για την τρίτη φραντζόλα ενώ η δεύτερη φραντζόλα φούσκωνει και η πρώτη φραντζολα ψηνοταν, και η συνέχιση αυτής της διαδικασίας για κάθε φραντζόλα έτσι ώστε να υπήρχουν τρεις φραντζόλες ψωμιου υπό εξέλιξη οποιαδήποτε στιγμή.Κάθε φραντζόλα του ψωμιού θα χρειαζοταν το ίδιο χρονικό διάστημα για να φτιαχτεί, αλλά ο αριθμός φραντζολών που γίνεται σε ένα δεδομένο χρονικό διάστημα θα αυξανόταν.

Η διοχέτευση είναι μια τεχνική εκτέλεσης διάφορων οδηγιών για να μειώσει το χρόνο εκτέλεσης ενός συνόλου οδηγιών.Παρόμοια με την αναλογία ψησίματος, κάθε οδηγία χρειάζεται το ίδιο χρονικό διάστημα να εκτελέσει σε έναν διοχετευμένο επεξεργαστή όπως θα γινόταν σε ένα μη διοχετευμένο επεξεργαστή (περισσότερο, πραγματικά, επειδή η διοχέτευση προσθέτει το υλικό στον επεξεργαστή),

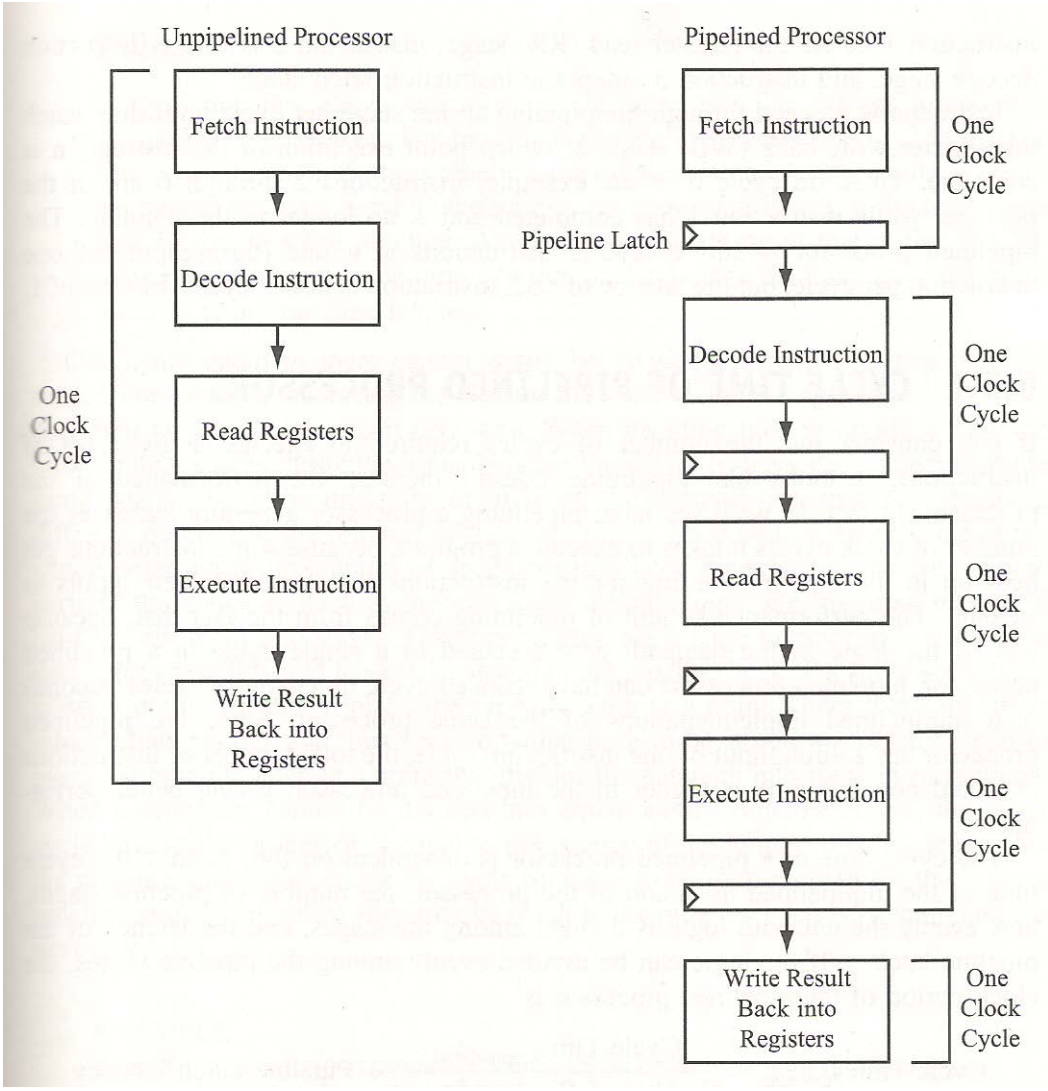
αλλά το ποσοστό στο οποίο οι οδηγίες μπορούν να εκτελεσθούν αυξάνεται με την εκτέλεση οδηγίας.

Όταν συζητάμε για την διοχετεύση, και η απόδοση του υπολογιστή είναι γενική , δύο όροι χρησιμοποιούνται συχνά: λανθάνουσα κατάσταση και ρυθμοαπόδοση.Η λανθάνουσα κατάσταση είναι το χρονικό διάστημα που μια ενιαία λειτουργία χρειάζεται για να εκτελέσει.Η ρυθμοαπόδοση είναι το ποσοστό στο οποίο οι διαδικασίες χρειάζονται για να εκτελεσθούν (γενικά εκφρασμένος ως διαδικασίες/δευτερόλεπτο ή διαδικασίες/κύκλο). Στο μη διοχετευμένο επεξεργαστή,ρυθμοαπόδοση = 1/λανθάνουσα κατάσταση, δεδομένου ότι η εκτέλεση της οδηγίας επικαλύπτεται.
Η λανθάνουσα κατάσταση ενός διοχετευμένου επεξεργαστή είναι επίσης σημαντική, εντούτοις, δεδομένου ότι καθορίζει πόσο συχνά οι εξαρτώμενες οδηγίες μπορούν να εκτελεσθούν.

4.3 Διοχέτευση

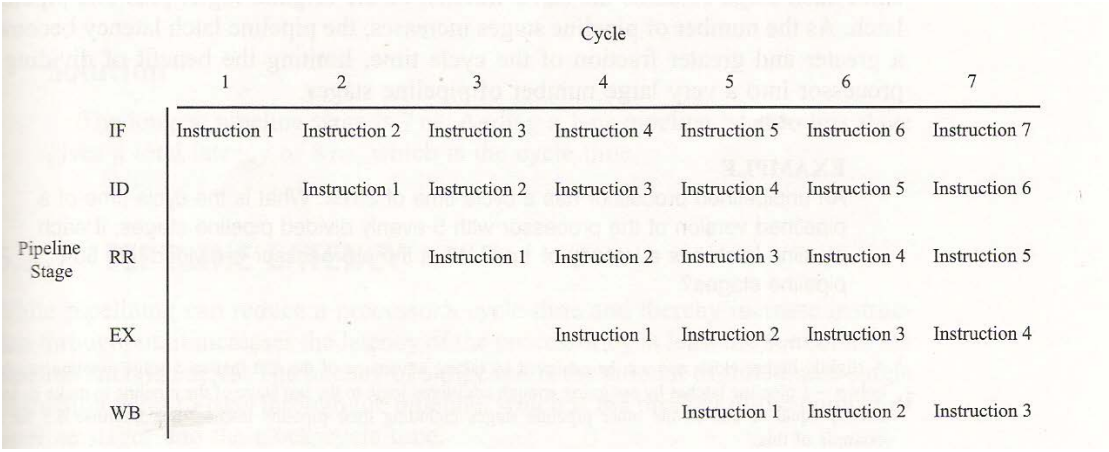
Για να εφαρμόσουν τη διοχέτευση, οι σχεδιαστές διαιρούν την δίοδο δεδομένων ενός επεξεργαστή σε τμήματα, και τοποθετούν τους σύρτες διοχετευσης μεταξύ κάθε τμήματος, όπως φαίνεται στο σχήμα 4-1.Στην έναρξη κάθε κύκλου, οι σύρτες διοχετευσης διαβάζουν τις εισόδους τους και τις αντιγράφουν στα αποτελέσματά τους, τα οποία παραμένουν σταθερά σε όλο το υπόλοιπο του κύκλου.Αυτό σπάζει τη δίοδο δεδομένων σε διάφορα τμήματα, κάθε ένα από τα οποία έχει μια λανθάνουσα κατάσταση ενός κύκλου ρολογιού, δεδομένου ότι μια οδηγία δεν μπορεί να περάσει μέσω ενός σύρτη διοχετευσεων μέχρι την έναρξη του επόμενου κύκλου.

Το ποσό της διόδου δεδομένων όπου ένα σήμα ταξιδεύει κατευθείαν σε έναν κύκλο καλείται στάδιο της διοχέτευσης, και οι σχεδιαστές περιγράφουν συχνά μια διοχέτευση που παίρνει n κύκλους ως n-στάδια διοχετευσης .Στο σχήμα 4-1, η διοχέτευση έχει πέντε στάδια.Το στάδιο 1 είναι ένα τμήμα οδηγίων ευρύτητας και ο σχετικός σύρτης διοχετευσεων,το στάδιο 2 είναι ένα τμήμα οδηγιων αποκωδικοποιησης και ο σύρτης διοχετευσεων του, και τα στάδια 3,4 και 5 είναι οι επόμενοι φραγμοί της διοχετευσης.Οι αρχιτέκτονες υπολογιστών διαφωνούν στο εάν ένας σύρτης διοχετευσεων είναι το τελευταίο μέρος ενός σταδίου ή το πρώτο μέρος του επόμενου σταδίου, έτσι ένα εναλλακτικό τμήμα της διοχετευσης στα στάδια επρόκειτο να μετρήσει το φραγμό οδηγίας ευρύτητας ως στάδιο 1, ο πρώτος σύρτης διοχετευσεων και το τμήμα οδηγιων αποκωδικοποιησης το ως στάδιο 2, και τα λοιπά.



ΣΧΗΜΑ 4-1 pipelined versus nonpipelined processor

Το σχήμα 4-2 δείχνει πώς οι οδηγίες διατρέχουν τη διοχετευση στο σχήμα 4-1. Στον κύκλο 1, η πρώτη οδηγία εισάγει το στάδιο οδηγίων ευρύτητας (IF) της διοχέτυσης και οι σταματάει στο σύρτη διοχετευσεων μεταξύ της οδηγίας ευρύτητας και στα στάδια της οδηγίας αποκωδικοποιησης (ID) στη διοχετευση .Στον κύκλο 2,η δεύτερη οδηγία εισάγει το στάδιο οδηγίων ευρύτητας , ενώ η οδηγία 1 απορεει στο στάδιο οδηγιων αποκωδικοποιησης.Στον τρίτο κύκλο, η οδηγία 1 εισάγει το στάδιο read register (RR), η οδηγία 2 είναι στο στάδιο οδηγιων αποκωδικοποιησης , και η οδηγία 3 εισάγει το στάδιο οδηγίων ευρύτητας .



ΣΧΗΜΑ4-2 instruction flow in a pipelined processor

Οι οδηγίες προχωρούν μέσω της διοχετευσης σε ένα στάδιο ανά κύκλο έως ότου φθασουν στο σταδιο κατάλογος write-back (WB), στο οποίο το σημείο εκτέλεσης της οδηγίας είναι πλήρης.Κατά συνέπεια, στον κύκλο 6 στο παράδειγμα, οι οδηγίες 2 μέχρι 6 είναι στη διοχετευση,ενώ η οδηγία 1 έχει ολοκληρώσει και δεν είναι πλέον καθ'οδόν .

Ο διοχετυμενος επεξεργαστής εκτελεί ακόμα τις οδηγίες σε ένα ποσοστό (ρυθμοαπόδοση) μιας οδηγίας ανά κύκλο, αλλά η λανθάνουσα κατάσταση κάθε οδηγίας είναι 5 κύκλοι αντι 1.

4.3.1 Ο κύκλος ζωής των διοχετευμένων επεξεργαστών

Εάν εξετάζετε ακριβώς τον αριθμό των κύκλων που απαιτούνται για να εκτελέσουν ένα δεδομένο σύνολο οδηγιών,αυτό μοιάζει με διοχετευση που δεν αυξάνει την απόδοση του επεξεργαστή.Στην πραγματικότητα, όπως θα δούμε αργότερα,η διοχέτευση ένος επεξεργαστή αυξάνει γενικά τον αριθμό των κύκλων ενός ρολογίου που χρειάζεται για να εκτελέσει ένα πρόγραμμα, επειδή μερικές οδηγίες διατηρούνται κρατημένες ψηλά στην διοχετευση περιμένοντας για τις οδηγίες που παράγουν τις εισόδους τους που εκτελούνται.Το όφελος απόδοσης της διοχετευσης προέρχεται από το γεγονός ότι, επειδή εκτελείται λιγότερο απο τη λογική στην δίοδο δεδομένων σε ένα απλό κύκλο σε έναν διοχετευμένο επεξεργαστή, οι διοχετευμένοι επεξεργαστές μπορούν να έχουν μειώσει τον χρόνο κύκλου (περισσοτεροι κυκλοι/δευτερόλεπτο) από ότι οι μη διοχετευμενες εφαρμογές του ίδιου επεξεργαστή.Δεδομένου ότι ο διοχετευμένος επεξεργαστής έχει μια ρυθμοαπόδοση της μίας οδηγίας/κύκλο, ο συνολικός αριθμός οδηγιών που εκτελούνται ανά χρόνο μονάδων είναι υψηλότερος στο διοχετευμένο επεξεργαστή, που δίνει την καλύτερη απόδοση.

Ο χρόνος κύκλου ενός διοχετευμένου επεξεργαστή εξαρτάται από τέσσερις παράγοντες:το κύκλος ζωής της μη διοχετευμενης εκτροπής του επεξεργαστή, τον

αριθμός σταδίων διοχετευσης,πόσο ομοιόμορφα η λογική διοδος δεδομενων διαιρείται μεταξύ των σταδίων, και τη λανθάνουσα κατάσταση των συρτών διοχετευσης.Εάν η λογική μπορεί να διαιρεθεί ομοιόμορφα μεταξύ των σταδίων διοχέτευσης, η περίοδος ενός ρολογιου του διοχετευμένου επεξεργαστή είναι :

Cycle TimePipelined =Cycle TimeUnpipelined + Pipeline Latch Latency
Number of Pipeline Stages

δεδομένου ότι κάθε στάδιο περιέχει το ίδιο κλάσμα της αρχικής λογικής, συν έναν σύρτη διοχετευσης.Δεδομένου ότι ο αριθμός σταδίων διοχετευσης αυξάνεται, η λανθάνουσα κατάσταση συρτών διοχετευσης γίνεται μεγαλύτερη και μεγαλύτερο το μέρος του χρόνου του κύκλου, που περιορίζει το όφελος ένος επεξεργαστή σε πολύ μεγάλο αριθμό σταδίων διοχετευσης .

4.3.2 Λανθάνουσα κατάσταση διοχετεύσεων

Ενω διοχετεύουμαι μπορούμε να μειώσουμε τον χρόνο κύκλου ενός επεξεργαστή και με αυτόν τον τρόπο να αυξήσουμε την ρυθμοαπόδοση της οδηγίας, αυτο αυξάνει τη λανθάνουσα κατάσταση του επεξεργαστή τουλάχιστον όσο το ποσό όλων των λανθανουσών καταστάσεων συρτών διοχετευσης.Η λανθάνουσα κατάσταση μιας διοχετευσης είναι το χρονικό διάστημα που μια ενιαία οδηγία χρειαζεται για να περάσει μέσω της διοχετευσης,η οποία είναι το προϊόν του αριθμού σταδίων διοχετευσης και του χρόνου κύκλου ενός ρολογιού.

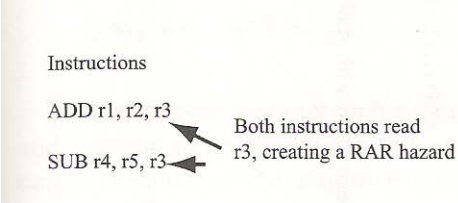
4.4 Κίνδυνοι οδηγίας και ο αντίκτυπος τους στη ρυθμοαπόδοση

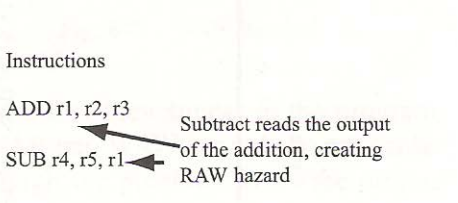
Όπως περιγράφεται παραπάνω,η διοχετεύση αυξάνει την απόδοση των επεξεργαστών από την αυξανόμενη ρυθμοαπόδοση οδηγίας.Επειδή διάφορες οδηγίες επικαλύπτονται κατά την διοχέτευση, ο χρόνος κύκλου μπορεί να μειωθεί, αυξάνοντας το ποσοστό στο οποίο οι οδηγίες εκτελούν.Στην ιδανική περίπτωση, η ρυθμοαπόδοση μιας διοχέτευσης είναι απλά 1/χρόνο κύκλου, έτσι ένα 5-σταδίο διοχέτευσης με χρόνο κύκλου 6ns και με μη διοχέτευμένο χρόνο κύκλου 25ns θα είχε μια ιδανική ρυθμοαπόδοση 1/6ns=1.67x108 οδηγίας/ων , περισσότερο από 4 x βελτίωση πέρα από τη ρυθμοαπόδοση του μη διοχετευμένου επεξεργαστή 4 x 107οδηγίας/ων.

Εντούτοις, υπάρχουν διάφοροι παράγοντες που περιορίζουν τη δυνατότητα μιας διοχέτευσης να εκτελεσεί τις οδηγίες στο μέγιστο ποσοστό τους, συμπεριλαμβανομένων των εξαρτήσεων μεταξύ των οδηγιών, των διακλάδωσεων, και του χρόνου που απαιτείται για να έχει πρόσβαση στη μνήμη.Σε αυτό το κεφάλαιο, θα συζητήσουμε πώς οι εξαρτήσεις και οι διακλαδώσεις οδηγίας έχουν επιπτώσεις στο χρόνο εκτέλεσης των προγραμμάτων για τους διοχετευμένους επεξεργαστές.

Οι κίνδυνοι οδηγίας (εξαρτήσεις) εμφανίζονται όταν οι οδηγίες διαβάζουν ή γράφουν τους καταλόγους που χρησιμοποιούνται από άλλες οδηγίες.Διαιρούνται σε τέσσερις κατηγορίες, ανάλογα με το εάν οι δύο οδηγίες περιλαμβάνουν read ή write άλλων κατάλογων.Read-After-Read (RAR) κινδύνοι, όπως φαίνεται στο σχήμα 4,3,

συμβαίνουν όταν δύο οδηγίες διαβάζουν από τον ίδιο κατάλογο.Οι κίνδυνοι RAR δεν προκαλούν πρόβλημα για τον επεξεργαστή επειδή η ανάγνωση ενός καταλόγου δεν αλλάζει την αξία του .Επομένως, δύο οδηγίες που έχουν έναν RAR κίνδυνο μπορούν να εκτελέσουν στους διαδοχικούς κύκλους (ή στον ίδιο κύκλο, στους επεξεργαστές που μπορούν να εκτελέσουν περισσότερες από μια οδηγίες/ κύκλο).





ΣΧΗΜΑ4-3 RAR hazard

ΣΧΗΜΑ4-4 RAW hazard

Οι Read-After-Write (RAW) κίνδυνοι εμφανίζονται όταν διαβάζει μια οδηγία έναν κατάλογο που γράφτηκε από μια προηγούμενη οδηγία, όπως φαίνεται στο σχήμα 4.4.Οι κίνδυνοι RAW είναι επίσης γνωστοι ως εξαρτήσεις στοιχείων ή αληθινές εξαρτήσεις, επειδή εμφανίζονται όταν πρέπει να χρησιμοποιήσει μια οδηγία το αποτέλεσμα μιας άλλης οδηγίας .

Όταν ένας RAW κίνδυνος εμφανίζεται, η οδηγία ανάγνωσης δεν μπορεί να προχωρήσει μετά από το στάδιο δίαβασματος του κατάλογου της διοχέτευσης έως ότου η οδηγία γραψίματος έχει περάσει στο στάδιο write-back, επειδή τα στοιχεία τα οποια χρειάζονται οι οδηγίες δεν είναι διαθέσιμα μέχρι τότε.Αυτό ονομάζεται pipeline stall ή φυσαλίδα.Σημειώστε οτι η οδηγία ανάγνωσης μπορεί να προχωρήσει μέσω της ευρύτητας οδηγίας και του στάδιου αποκωδικοποίησης οδηγίας της διοχέτευσης προτού να ολοκληρώσει η οδηγία γραψίματος, επειδή η οδηγία ανάγνωσης δεν χρειάζεται την τιμή που παράγεται από την οδηγία γραψίματος έως ότου φτάσει στο στάδιο διάβασμα καταλόγου .

4.4.1 Διακλαδώσεις

Οι οδηγίες διακλάδωσης μπορούν επίσης να προκαλέσουν καθυστερήσεις στους διοχετευμένους επεξεργαστές, επειδή ο επεξεργαστής δεν μπορεί να καθορίσει ποία οδηγία να προσκομίσει μέχρι η διακλάδωση να έχει εκτελέστει.Αποτελεσματικά, οι οδηγίες διακλάδωσης ,ιδιαίτερα οδηγίες άλματος υπό συνθήκη, δημιουργούν τις εξαρτήσεις στοιχείων μεταξύ της οδηγίας διακλάδωσης και του σταδίου ευρύτητας οδηγίας της διοχέτευσης ,ένω η οδηγία διακλάδωσης υπολογίζει τη διεύθυνση της επόμενης οδηγίας όταν το στάδιο ευρήτητας οδηγίας πρέπει να προσκομίσει .Το σχήμα 4-8 επιδεικνύει πώς μια οδηγία διακλάδωσης θα εκτελούσε στη διοχέτευση πέντε σταδίων .

Το PC ενημερώνεται στο τέλος του κύκλου ότι η οδηγία διακλάδωσης είναι στο στάδιο εκτάλεσης, που επιτρέπει στην επόμενη οδηγία να προσκομιστεί στον ακόλουθο κύκλο.

Η καθυστέρηση μεταξύ σε μια οδηγία διακλάδωσης εισάγει τη διοχέτευση και στο χρόνο στον οποίο η επόμενη οδηγία εισάγει τη διοχέτευση καλείται συχνά καθυστέρηση διακλάδωσης του επεξεργαστή.Οι καθυστερήσεις διακλάδωσης καλούνται μερικές φορές κινδύνοι ελέγχου, επειδή η καθυστέρηση οφείλεται στον έλεγχο ροής του προγράμματος.Η διοχέτευση που διευκρινίζεται στο σχήμα 4.8 έχει μια καθυστέρηση διακλάδωσης τεσσάρων κύκλων.

Οι καθυστερήσεις διακλάδωσης ασκούν σημαντική επίδραση στην απόδοση των σύγχρονων επεξεργαστών, και διάφορες τεχνικές έχουν αναπτυχθεί για να τις εξετάσουν.Μια τεχνική είναι να αυξηθεί το υλικό για να επιτρέψει στο αποτέλεσμα μιας οδηγίας διακλάδωσης να υπολογιστεί νωρίτερα στην διοχέτευση.Παραδείγματος χάριν, εάν η διοχέτευση υπολόγισει τη νέα αξία του PC στο στάδιο ανάγνωση κατάλογου αντί του στάδιου εκτελέσης, η καθυστέρηση διακλάδωσης θα μειωνόταν σε τρεις κύκλους.Μια άλλη τεχνική είναι να αυξηθεί το υλικό που προβλέπει τη διεύθυνση προορισμού κάθε διακλάδωσης προτού να ολοκληρώσει η διακλάδωση, επιτρέποντας στον επεξεργαστή να αρχίσει να προσκομίζει τις οδηγίες από εκείνη την διεύθυνση νωρίτερα στη διοχέτευση.Αυτη η τεχνική πρόβλεψη διακλάδωσης είναι πέρα από το πεδίο αυτού του βιβλίου, αλλά βελτιώνουν την απόδοση των σύγχρονων επεξεργαστών.

4.4.2 Δομικοί κίνδυνοι

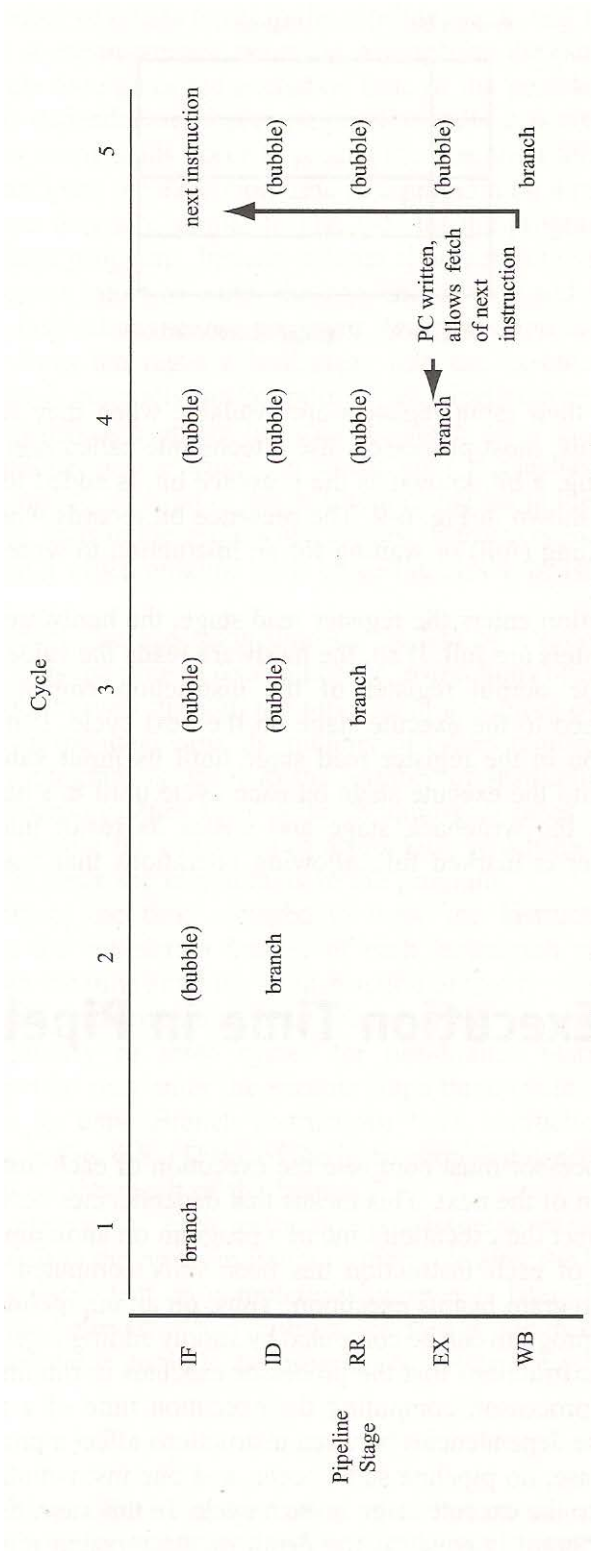
Μια τελική αιτία των καθυστερήσεων στους διοχετευμένους επεξεργαστές είναι οι δομικοί κίνδυνοι.Οι δομικοί κίνδυνοι εμφανίζονται όταν το υλικό του επεξεργαστή δεν είναι σε θέση εκτελέσει όλες τις οδηγίες στη διοχέτευση ταυτόχρονα Παραδείγματος χάριν, εάν το αρχείο καταλόγων δεν είχε αρκετές θέσεις για να επιτρέψει σε μια οδηγία στο στάδιο WB να γράψει το αποτέλεσμά του στο αρχείο καταλόγων στον ίδιο κύκλο όπου μια άλλη οδηγία στο στάδιο RR διαβάζεται από το αρχείο καταλόγων, αυτό θα ήταν απαραίτητο για να χρονοτριβήσει οποιαδήποτε οδηγία στο στάδιο RR που διαβάζεται εάν υπήρξε επίσης μια οδηγία στο στάδιο WB σε εκείνο τον κύκλο.(Η επιλογή να χρονοτριβησεί η οδηγία στο στάδιο WB για να επιτρέψει στην οδηγία στο στάδιο RR να προχωρήσει θα ήταν μια φτωχή επιλογή, δεδομένου ότι η καθυστέρηση στο στάδιο WB θα απέτρεπε τις οδηγίες στο EX στάδιο από την προώθηση.)

Οι δομικοί κίνδυνοι μέσα σε μια ενιαία διοχέτευση είναι σχετικά σπάνιοι στους σύγχρονους επεξεργαστές, επειδή τα σύνολά του υλικού και οδηγίας έχουν ως σκοπό να υποστηρίξουν την διοχέτευση.Εντούτοις, οι επεξεργαστές που εκτελούν περισσότερες από μια οδηγίες σε έναν κύκλο,οι οποίες καλύπτονται στο επόμενο κεφάλαιο,έχουν συχνά τους περιορισμούς στους τύπους οδηγιών που το υλικό μπορεί

να εκτελέσει ταυτόχρονα .Παραδείγματος χάριν, ένας επεξεργαστής πρέπει να είναι σε θέση να εκτελέσει δύο οδηγίες σε κάθε κύκλο, αλλά μόνο εάν μια από τις οδηγίες ήταν μια λειτουργία ακέραιων αριθμών και άλλη ένας υπολογισμός κινητής υποδιαστολής.

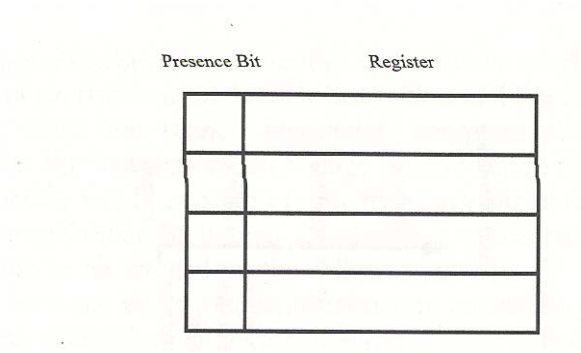
4.4.3 SCOREBOARDING

Οι διοχετευμένοι επεξεργαστές πρέπει να κρατήσουν τη διαδρομή στην οποία οι κατάλογοι θα γραφτούν από τις οδηγίες που είναι ήδη στη διοχέτευση , έτσι ώστε οι επόμενες οδηγίες να μπορούν να αποφασίσουν εάν οι κατάλογοι εισόδους τους είναι διαθέσιμοι όταν φθάνουν στο στάδιο ανάγνωσης κατάλογου. Για να κάνουν αυτό, οι περισσότεροι επεξεργαστές χρησιμοποιούν μια τεχνική αποκαλούμενη scoreboarding καταλόγων. Στο scoreboarding,ένα bit, προστίθεται σε κάθε κατάλογο στο αρχείο καταλόγων, όπως φαίνεται στο σχήμα 4.5.Η παρουσία των bit αντιγράφει εάν ο κατάλογος είναι διαθέσιμος για την ανάγνωση (πλήρη) ή περιμένει για μια οδηγία για να γράψει την τιμή εξόδου του (κενή).



ΣΧΗΜΑ4-5 branch instruction in pipeline

Όταν μια οδηγία εισάγει το στάδιο ανάγνωσης καταλόγου , το υλικό έλεγχει για να δεί εάν όλες οι εισοδοι των καταλόγων είναι γεμάτοι .Σε αυτή την περίπτωση, το υλικό διαβάζει τις αξίες απο όλες τις εισόδους των καταλόγων, χαρακτηρίζει τον κατάλογο εξόδου της οδηγίας κενό, και επιτρέπει στην οδηγία να προχωρήσει στο στάδιο εκτέλεσης στον επόμενο κύκλο.Αν όχι, το υλικό κρατά την οδηγία στο στάδιο αναγνωσής καταλόγου εώς ότου οι τιμές εισόδου της γίνονται πλήρεις, παρεμβάλλοντας τις φυσαλίδες εκτελέστε το στάδιο σε κάθε κύκλο έως ότου συμβει αυτό.Όταν μια οδηγία φθάνει στο στάδιο writeback και γράφει το αποτέλεσμά της στον κατάλογο προορισμού της, εκείνος ο κατάλογος είναι χαρακτηρισμένο σύνολο, που επιτρέπει τις διαδικασίες που διαβάζουν τον κατάλογο να προχωρήσουν.



ΣΧΗΜΑ4-6 register scoreboard

ΚΕΦΑΛΑΙΟ 5ο

ΠΑΡΑΛΛΗΛΙΣΜΟΣ ΣΕ ΕΠΙΠΕΔΟ ΕΝΤΟΛΩΝ INSTRUCTION LEVEL PARALLELISM.

5.1.Στόχοι

Αυτό το κεφάλαιο καλύπτει μια ποικιλία τεχνικών εκμετάλλευσης του instruction level parallelism (ILP) , εκτελώντας πολλές ανεξάρτητες εντολές την ίδια στιγμή.

5.2. Εισαγωγή

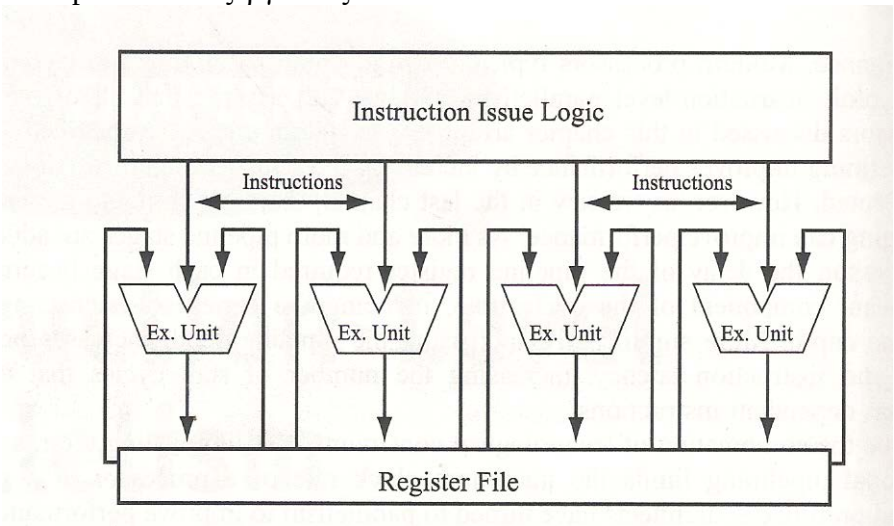
Στο προηγούμενο κεφάλαιο είδαμε τη συνεχή διοχέτευση (pipelining) , μια σημαντική τεχνική που αυξάνει την επίδοση του υπολογιστή, υπερκαλύπτοντας την εκτέλεση πολλαπλών εντολών. Με αυτό τον τρόπο οι εντολές μπορούν να εκτελεστούν με πιο γρήγορο ρυθμό από ότι αν κάθε εντολή έπρεπε να περιμένει την ολοκλήρωση της προηγούμενης για να εκτελεστεί. Σε αυτό το κεφάλαιο θα εξερευνήσουμε τεχνικές που εκμεταλλεύονται το instruction level parallelism με την εκτέλεση πολλαπλών οδηγιών ταυτόχρονα, αυξάνοντας ακόμα περισσότερο τις επιδόσεις. Οι σύγχρονοι επεξεργαστές υποστηρίζουν και τη διοχέτευση αλλά και τεχνικές που εκμεταλλεύονται τον instruction level parallelism και θα υποθέσουμε ότι όλοι οι ILP επεξεργαστές που θα δούμε σε αυτό το κεφάλαιο υποστηρίζουν την διοχέτευση, και αν όχι θα το αναφέρουμε.

Η διοχέτευση βελτιώνει τις επιδόσεις αυξάνοντας το ρυθμό εκτέλεσης των εντολών. Παρόλα αυτά, όπως είδαμε και στο προηγούμενο κεφάλαιο, υπάρχουν περιορισμοί στην ποσότητα της διοχέτευσης που βελτιώνει την απόδοση του συστήματος. Καθώς όλο και περισσότερα στάδια διοχέτευσης προστίθενται στον επεξεργαστή η καθυστέρηση του καταχωρητή διοχέτευσης που απαιτείται σε κάθε στάδιο γίνεται σημαντικό κομμάτι του χρόνου του κύκλου και συνεπώς αυξάνει το βάθος διοχέτευσης και μειώνει το κέρδος χρόνου. Επιπρόσθετα η αύξηση του βάθους διοχέτευσης αυξάνει τις καθυστερήσεις του κλάδου και των εντολών καθώς και τον αριθμό των καθυστερήσεων των κύκλων που συμβαίνουν ενδιάμεσα από τις εξαρτημένες εντολές.
Με το συνδυασμό των τεχνολογικών περιορισμών και τις μειώσεις λόγω των ορίων στη διοχέτευση περιορίζουν το ρυθμό ρολογιού (clock rate) του επεξεργαστή σε μια επινοημένη διεργασία, οι αρχιτέκτονες στράφηκαν στον παραλληλισμό για να βελτιώσουν την απόδοση με την εκτέλεση πολλαπλών εργασιών ταυτόχρονα. Τα παράλληλα συστήματα υπολογιστών συνήθως παίρνουν δύο μορφές: πολυεπεξεργαστές και instruction level parallelism επεξεργαστές., οι οποίοι ποικίλουν ανάλογα με τον αριθμό των εργασιών που θέλουμε να εκτελούνται παράλληλα. Στα πολυεπεξεργαστικά συστήματα σχετικά μεγάλες διεργασίες, όπως διαδικασίες (procedures) ή επαναλήψεις βρόγχων (loop iterations), εκτελούνται παράλληλα. Αντίθετα στα instruction level parallelism συστήματα οι επεξεργαστές εκτελούν παράλληλα ιδιαίτερες οδηγίες.

Οι επεξεργαστές που εκμεταλλεύονται τον instruction level parallelism είναι πιο επιτυχημένοι από τους πολυεπεξεργαστές στην αγορά προσωπικών υπολογιστών και σταθμός εργασίας, γιατί παρέχουν βελτιώσεις στην εκτέλεση συμβατικών προγραμμάτων πράγμα που δεν έγινε με τους πολυεπεξεργαστές. Συγκεκριμένα οι υπερβαθμωτοί (superscalar) επεξεργαστές μπορούν να επιταχύνουν κατά την εκτέλεση προγραμμάτων που μεταγλωττίστηκαν για εκτέλεση σε σειριακούς (non-ILP) επεξεργαστές, χωρίς να χρειαστούν ξανά μεταγλώττιση. Η άλλη αρχιτεκτονική που θα καλυφθεί σε αυτό το κεφάλαιο οι επεξεργαστές VLIW απαιτούν τα προγράμματα να ξαναμεταφραστούν στη νέα αρχιτεκτονική άλλα πετυχαίνει και καλές επιδόσεις σε προγράμματα γραμμένα σε σειριακές γλώσσες όπως η C και η FORTRAN όταν βέβαια ξαναμεταγλωττιστούν για να λειτουργούν σε VLIW επεξεργαστή.

Ένα διάγραμμα υψηλού επιπέδου ενός instruction level parallel επεξεργαστή φαίνεται στο σχήμα 5-1. Ο επεξεργαστής περιέχει πολλαπλές μονάδες εκτέλεσης για την εκτέλεση εντολών ή κάθε μία από τις οποίες διαβάζει τους τελεστές της από, και γράφει τα αποτελέσματα σε ένα κεντρικό αρχείο εγγραφής. Όταν μια διεργασία γράφει τα αποτελέσματα της πίσω στο αρχείο εγγραφής το αποτέλεσμα γίνεται ορατό σε όλες τις μονάδες εκτέλεσης του επόμενου κύκλου επιτρέποντας στις διεργασίες να εκτελούνται σε διαφορετικές μονάδες και όχι σε αυτές που δημιουργούν τις εισόδους τους. Οι instruction level parallel επεξεργαστές συχνά έχουν περίπλοκο παρακαμπτήριο (bypassing) hardware το οποίο προωθεί τα αποτελέσματα κάθε εντολής σε όλες τις εκτελεστικές μονάδες για να μειώσει τις καθυστερήσεις ανάμεσα σε εξαρτημένες εντολές.

Οι οδηγίες που φτιάχνουν ένα πρόγραμμα μπορούν να διαχειριστούν από την instruction issue logic που κατανέμει τις εντολές στις παράλληλες μονάδες Με αυτό τον τρόπο αλλαγές ροής όπως οι κλάδοι συμβαίνουν ταυτόχρονα σε όλες τις μονάδες κάνοντας πιο εύκολη τη γραφή και τη μεταγλώττιση προγραμμάτων σε instruction level parallel επεξεργαστές.



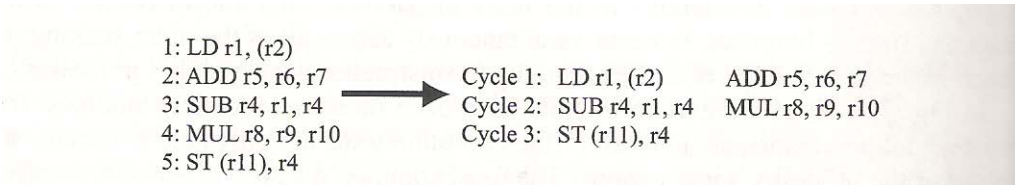
ΣΧΗΜΑ5-1 instruction-level parallel processor

Στο σχήμα 5-1 , όλες οι εκτελεστικές μονάδες έχουν σχεδιαστεί σαν πανομοιότυπες μονάδες. Στην πραγματικότητα, μερικές ή και όλες από τις μονάδες μπορούν να εκτελέσουν ένα υποσετ από τις εντολές του επεξεργαστή. Ο πιο κοινός διαχωρισμός

γίνεται μεταξύ διεργασιών ακεραίων και διεργασιών δεκαδικών γιατί αυτές οι λειτουργίες απαιτούν διαφορετικό hardware. Η υλοποίηση αυτών των δύο σετ hardware σε ξεχωριστές μονάδες αυξάνει τον αριθμό των εντολών που μπορούν να εκτελεστούν ταυτόχρονα χωρίς να αυξάνονται και οι απαιτήσεις σε hardware. Σε άλλους επεξεργαστές κάποιες από τις εκτελεστικές μονάδες ακεραίων μπορεί να σχεδιαστούν να εκτελούν μόνο τις πιο κοινές διεργασίες ακεραίων. Αυτό μειώνει το μέγεθος των μονάδων εκτέλεσης παρόλο που σημαίνει ότι κάποιοι συνδυασμοί ανεξάρτητων ακεραίων δεν μπορούν να εκτελεστούν παράλληλα.

5.3 Τι είναι INSTRUCTION LEVEL PARALLELISM

Οι instruction level parallel επεξεργαστές εκμεταλλεύονται το γεγονός ότι πολλές από τις εντολές / οδηγίες σε ένα σειριακό πρόγραμμα δεν εξαρτώνται από τις εντολές που προηγούνται αυτών μέσα στο πρόγραμμα. Για παράδειγμα δείτε το κομμάτι του προγράμματος στα δεξιά του σχήματος 5-2. Οι εντολές 1,3 και 5 εξαρτώνται οι μία από την άλλη γιατί η εντολή 1 δημιουργεί μια τιμή που χρησιμοποιεί η εντολή 3, η οποία με τη σειρά της δημιουργεί άλλη τιμή του χρησιμοποιεί η εντολή 5. Οι εντολές 2 και 4 δε χρησιμοποιούν αποτέλεσμα άλλης εντολής, ούτε δημιουργούν κάποιο αποτέλεσμα που χρησιμοποιείται από άλλη εντολή στο τμήμα που εξετάζουμε. Οι εξαρτήσεις αυτές απαιτούν οι εντολές 1,4 και 5 να εκτελεστούν στη σωστή σειρά για να έχουμε και το σωστό αποτέλεσμα, και οι εντολές 2 και 4 μπορούν να εκτελεστούν νωρίτερα, μετά η και παράλληλα χωρίς να αλλάζουν το αποτέλεσμα.



ΣΧΗΜΑ5-2 instruction-level parallelism example

Σε έναν επεξεργαστή που εκτελεί μία εντολή τη φορά, ο χρόνος εκτέλεσης αυτού του προγράμματος θα ήταν 5 κύκλοι, ακόμα και σε ένα επεξεργαστή που δεν υποστηρίζει διοχέτευση(pipelining) και που έχει καθυστέρηση εντολής 1 κύκλο. Αντίθετα, ένας επεξεργαστής που ναι μεν δεν υποστηρίζει διοχέτευση αλλά μπορεί να εκτελέσει δύο εντολές ταυτόχρονα μπορεί να εκτελέσει το πρόγραμμα σε τρεις κύκλους. Αν κάθε εντολή είχε μια καθυστέρηση ενός κύκλου όπως φαίνεται στο σχήμα. Επειδή οι εντολές 1,3 και 5 είναι εξαρτημένες δε μπορούμε να μειώσουμε το χρόνο εκτέλεσης του τμήματος του προγράμματος ακόμα και αν αυξήσουμε τον αριθμό των εντολών που μπορεί ο επεξεργαστής να εκτελέσει ταυτόχρονα.

Αυτό το παράδειγμα διευκρινίζει και τις δυνάμεις αλλά και τις αδυναμίες του instruction level parallelism. Οι ILP επεξεργαστές μπορούν να πετύχουν σημαντικές επιταχύνσεις σε μια ευρεία ποικιλία προγραμμάτων εκτελώντας παράλληλα εντολές, αλλά η μέγιστη απόδοσή του εξαρτάται από τις εξαρτήσεις των εντολών αυτών. Γενικότερα, όσο περισσότερες εκτελεστικές μονάδες προστίθενται στον επεξεργαστή τόσο μειώνεται η προσαύξηση στην βελτίωση της απόδοσης. Η μετάβαση από μια σε δύο μονάδες εκτέλεσης μειώνει τον πραγματικό χρόνο εκτέλεσης. Όταν όμως ο

αριθμός των μονάδων αυξάνεται σε τέσερερις, οχτώ και παραπάνω οι παραπάνω μονάδες τον περισσότερο χρόνο μένουν αδρανείς, ειδικά να το πρόγραμμα δεν έχει μεταγλωττιστεί έτσι ώστε να τις εκμεταλλεύεται.

5.4. Περιορισμοί του INSTRUCTION LEVEL PARALLELISM

Η απόδοση των ILP επεξεργαστών περιορίζεται από την ποσότητα των instruction level parallelism που ο μεταγλωττιστής και το hardware μπορούν να εντοπίσουν στο πρόγραμμα. Με τη σειρά το ο instruction level parallelism περιορίζεται από αρκετούς παράγοντες, όπως εξαρτήσεις δεδομένων (data dependencies), εξαρτήσεις ονομάτων (name dependencies) όπως WAR και WAW κίνδυνοι, και κλάδοι (branches). Ακόμα η δυνατότητα κάποιου επεξεργαστή να εκμεταλλεύεται τον instruction level parallelism μπορεί αν περιοριστεί από τον τύπο και τον αριθμό των εκτελεστικών μονάδων και από περιορισμούς σε σχέση με ο ποιες εντολές μπορούν να εκτελεστούν παράλληλα.

Ένας πολύ σημαντικός περιορισμός προέρχεται από τις RAW εξαρτήσεις, οι οποίες περιορίζουν την απόδοση απαιτώντας οι οδηγίες να εκτελεστούν σειριακά, έτσι ώστε να μην υπάρχουν λάθη στο αποτέλεσμα . Οι εντολές με WAW εξαρτήσεις πρέπει επίσης να κατανέμονται σειριακά για να εξασφαλιστεί ότι η σωστή εντολή θα γράψει τελευταία στον καταχωρητή. Οι εντολές με WAR εξαρτήσεις μπορούν να κατανεμηθούν στον ίδιο κύκλο αλλά όχι σε τυχαία σειρά (out-of-order) γιατί οι εντολές διαβάζουν τις εισόδους τους από το αρχείο εγγραφής πριν κατανεμηθούν. Έτσι μια εντολή που διαβάζει τον καταχωρητή μπορεί να κατανεμηθεί στον ίδιο κύκλο που κάποια άλλη γράφει σε αυτόν αλλά εμφανίζεται αργότερα στο πρόγραμμα. Η εντολή που διαβάζει τον καταχωρητή θα πάρει την τιμή της πριν αυτή αλλάξει από την εντολή αποθήκευσης. Πιο κάτω στο κεφάλαιο, θα μιλήσουμε για την μετονομασία καταχωρητών (register rename), μια τεχνική του hardware που επιτρέπει σε WAR και WAW εξαρτήσεις να εκτελούνται με τυχαία σειρά χωρίς να αλλάζει το αποτέλεσμα του προγράμματος.
Οι κλάδοι (branches) περιορίζουν το instruction level parallelism γιατί ο επεξεργαστής δεν ξέρει ποιες εντολές να εκτελέσει μετά έναν κλάδο , πριν ολοκληρωθεί η εκτέλεση του. Αυτό απαιτεί από τον επεξεργαστή να περιμένει να ολοκληρωθεί η εκτέλεση του κλάδου πριν προχωρήσει στην εκτέλεση οποιασδήποτε εντολής. Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο πολύ επεξεργαστές ενσωματώνουν hardware πρόβλεψης για να μειώσουν την επίδραση της εκτέλεσης του κλάδου στο χρόνο εκτέλεσης του προγράμματος. Το hardware αυτό προβλέπει την διεύθυνση προορισμού ενός κλάδου, πριν ολοκληρωθεί η εκτέλεσή του.

5.5 Υπερβαθμωτοί επεξεργαστές (superscalar processors)

Οι υπερβαθμωτοί επεξεργαστές βασίζονται στο hardware για την εξαγωγή instruction level parallelism από σειριακά προγράμματα. Κατά τη διάρκεια κάθε στυλ, η λογική κατανομής εντολών του υπερβαθμωτού επεξεργαστή εξετάζει τις εντολές του σειριακού προγράμματος για να καθορίσει ποιες εντολές θα κατανεμηθούν και σε πιο κύκλο. Αν υπάρχει ήδη αρκετός instruction level parallelism, ο υπερβαθμωτός επεξεργαστής μπορεί να εκτελέσει μία εντολή ανά μονάδα ανά κύκλο, ακόμα και αν το πρόγραμμα μεταγλωττίστηκε για εκτέλεση σε επεξεργαστή που μπορεί να εκτελεί μόνο μία εντολή ανά κύκλο.

Αυτή η ιδιότητα είναι ένα από τα κυριότερα πλεονεκτήματα των υπερβαθμωτών επεξεργαστών και ο λόγος γιατί εικονικά όλοι οι σταθμοί εργασίας και τα pc's έχουν τέτοιο τύπο επεξεργαστή. Οι υπερβαθμωτού επεξεργαστές μπορούν να τρέξουν προγράμματα μεταγλωττισμένα για καθαρά σειριακούς επεξεργαστές και μπορούν να πετύχουν καλύτερες επιδόσεις σε αυτά τα προγράμματα από επεξεργαστές που δε μπορούν να εκμεταλλευτούν τον instruction level parallelism. Έτσι χρήστες που αγοράζουν συστήματα με υπερβαθμωτούς επεξεργαστές μπορούν να εγκαταστήσουν τα παλιά τους προγράμματα και να τα δουν να εκτελούνται σε πολύ καλύτερους χρόνους, πράγμα αδύνατο με το παλιό τους σύστημα.

Η δυνατότητα των υπερβαθμωτών επεξεργαστών να εκμεταλλεύονται τον instruction level parallelism σε σειριακά προγράμματα δε σημαίνει ότι οι μεταγλωττιστές είναι άχρηστοι σε τέτοια συστήματα, Στην πραγματικότητα οι καλοί μεταγλωττιστές είναι ακόμα πιο χρήσιμοι στους υπερβαθμωτούς παρά στους σειριακούς επεξεργαστές. Οι υπερβαθμωτοί επεξεργαστές μπορούν να εξετάσουν μόνο ένα μικρό κομμάτι των εντολών ενός προγράμματος τη φορά , για να καθορίσουν ποιες εντολές μπορούν να εκτελεστούν παράλληλα.. Αν ο μεταγλωττιστής μπορεί να οργανώσει τις εντολές με τέτοιο τρόπο ώστε πολλές από τις ανεξάρτητες εντολές να εμφανιστούν στο μικρό κομμάτι (παράθυρο) που εξετάζεται, θα έχουμε και καλύτερες επιδόσεις στο πρόγραμμα. Αν οι περισσότερες από τις εντολές στο κομμάτι αυτό είναι εξαρτημένες ο υπερβαθμωτός επεξεργαστής δε θα μπορέσει να επιτύχει χρόνο εκτέλεσης ιδιαίτερα μεγαλύτερο από αυτόν που θα πετύχαινε ένας απλός σειριακός επεξεργαστής. Στο 5.9. θα μελετήσουμε τεχνικές με τις οποίες ο μεταγλωττιστής βελτιώνει τις επιδόσεις υπερβαθμωτών επεξεργαστών.

5.6.Εκτέλεση σε σειρά vs τυχαία εκτέλεση.

Ένα από τα πιο σημαντικά ζητήματα στη σχεδίαση υπερβαθμωτών επεξεργαστών είναι και ο τρόπος με τον οποίο θα εκτελούνται οι εντολές. Δηλαδή αν θα εκτελούνται με τη σειρά που εμφανίζονται στο πρόγραμμα (in order) ή εάν έχει τη δυνατότητα να εκτελεί τις εντολές με οποιαδήποτε σειρά που δεν αλλάζει το τελικό αποτέλεσμα του προγράμματος (out of order). Η τυχαία εκτέλεση παρέχει καλύτερη απόδοση από ότι η εκτέλεση σε σειρά αλλά απαιτεί πιο περίπλοκο hardware.

5.6.1.Πρόβλεψη χρόνου εκτέλεσης σε in order επεξεργαστές

Στο προηγούμενο κεφάλαιο, χωρίσαμε το χρόνο εκτέλεσης των προγραμμάτων σε επεξεργαστές διοχέτευσης (pipelined processors) στο χρόνο κατανομής όλων των προγραμμάτων και στην καθυστέρηση διοχέτευσης, και έτσι πήραμε αυτό τον τύπο :

Χρόνος Εκτέλεσης (σε κύκλους)= Καθυστέρηση διοχέτευσης + Χρόνο Κατανομής-1

Σε διοχετευμένους (pipelined) ILP επεξεργαστές μπορούμε ν χρησιμοποιήσουμε τον ίδιο τύπο για τον χρόνο εκτέλεσης αλλά το να υπολογίσουμε τι χρόνο κατανομής είναι πιο δύσκολο αφού ο επεξεργαστής μπορεί να κατανείμει περισσότερες από μια εντολές σε ένα κύκλο. Με δεδομένο ότι η καθυστέρηση διοχέτευσης δεν αλλάζει από πρόγραμμα σε πρόγραμμα, οι περισσότερες ασκήσεις σε αυτό το κεφάλαιο θα βασιστούν στον καθορισμό του χρόνου κατανομής σε ILP επεξεργαστές.

Στους in order υπερβαθμωτούς επεξεργαστές ο χρόνος κατανομής ενός προγράμματος μπορεί να καθοριστεί εξετάζοντας σειριακά τον κώδικα για να δούμε ποιες εντολές μπορούμε να κατανείμουμε. Αυτή τεχνική είναι ίδια με την τεχνική που χρησιμοποιείται για διοχετευμένους (pipelined) επεξεργαστές που εκτελούν μόνο μια εντολή σε κάθε κύκλο. Η βασική διαφορά μεταξύ ενός in order υπερβαθμωτού και ενός διοχετευμένου μη υπερβαθμωτού επεξεργαστή είναι ότι ο υπερβαθμωτός μπορεί να κατανείμει μια εντολή στον ίδιο κύκλο με την προηγούμενή της αν το επιτρέπουν οι εξαρτήσεις, αρκεί ο αριθμός των εντολών που κατανέμονται στον κύκλο να μην υπερβαίνει τον αριθμό των εντολών που μπορεί ο επεξεργαστής να εκτελέσει ταυτόχρονα. Σε επεξεργαστές όπου μερικές ή όλες οι μονάδες εκτέλεσης μπορούν να εκτελέσουν μόνο ορισμένες εντολές, το σετ των εντολών που κατανέμεται σε ένα κύκλο πρέπει να συναγωνίζεται τους περιορισμούς των μονάδων εκτέλεσης.

5.6.2. Πρόβλεψη χρόνου εκτέλεσης σε out of order επεξεργαστές.

Ο καθορισμός του χρόνου κατανομής σε έναν out of order επεξεργαστή είναι σαφώς δυσκολότερος από τον καθορισμό του χρόνου της ίδιας ακολουθίας σε in order επεξεργαστή, γιατί είναι πολλές οι πιθανές ακολουθίες με τις οποίες μπορούν να εκτελεστούν οι εντολές. Γενικά η καλύτερη προσέγγιση είναι να εξετάσουμε αν υπάρχουν εξαρτήσεις ανάμεσα στις εντολές. Μετά από αυτό οι εντολές κατανέμονται σε κύκλους για να υπολογιστεί η μικρότερη καθυστέρηση ανάμεσα στην εκτέλεση της πρώτης και της τελευταίας εντολής.

Η προσπάθεια εξεύρεσης της καλύτερης δυνατής ακολουθίας σετ οδηγιών, μεγαλώνει μα τον αριθμό των οδηγιών/ εντολών του σετ, αφού πρέπει να σκεφτούμε κάθε πιθανή τακτοποίηση. Γι αυτό το λόγο θα υποθέσουμε ότι η λογική εντολών (instruction logic)σε έναν υπερβαθμωτό επεξεργαστή περιορίζει με κάποιο τρόπο τη σειρά με την οποία μπορούν να ταξινομηθούν οι εντολές ώστε να απλοποιηθεί η διαδικασία κατανομής τους. Η υπόθεση που θα κάνουμε είναι ότι ο επεξεργαστής κατανέμει μια εντολή στον πρώτο κύκλο που του επιτρέπουν οι εξαρτήσεις ανάμεσα στις εντολές. (Αυτή η απλοποίηση μπορεί να μην ισχύει σε όλους τους επεξεργαστές, αφού ανάλογα με τον επεξεργαστή και την λογική κατανομής εντολών τους αλλάζει και το τρόπος τακτοποίησης.) Αν περισσότερες εντολές μπορούν να κατανεμηθούν σε αυτό τον κύκλο από τον αριθμό εκτελεστικών μονάδων, ο επεξεργαστής τις ταξινομεί με βάση τη σειρά τους στο πρόγραμμα, ακόμα και αν η ταξινόμησή του σε άλλη σειρά θα μείωνε το χρόνο κατανομής. Όταν ο μεταγλωττιστής μπορεί να ελέγξει την κατανομή των εντολών, όπως στους VLIW επεξεργαστές που είδαμε στο 7.8., υποθέτουμε ότι ο μεταγλωττιστής ψάχνει όλες τις πιθανές περιπτώσεις και επιλέγει τη σειρά εντολών με μικρότερο χρόνο εκτέλεσης. Αυτό γίνεται γιατί ο μεταγλωττιστής μπορεί να αφιερώσει περισσότερη προσπάθεια στην τακτοποίηση των εντολών από τη λογική κατανομής εντολών.

Με αυτή την απλοποίηση είναι πιο εύκολο να βρούμε το χρόνο κατανομής μιας ακολουθίας εντολών σε έναν out-of-order επεξεργαστή. Ξεκινώντας από την πρώτη εντολή της ακολουθίας, ο επεξεργαστής διαβάζει όλες τις εντολές εκχωρώντας τες στον πρώτο κύκλο κατά τον οποίο οι τελεστές εισόδου της εντολές είναι διαθέσιμοι. Ο αριθμός εντολών που εκχωρούνται σε κάθε κύκλο πρέπει να είναι μικρότερος από τον αριθμό των εντολών που μπορεί ο επεξεργαστής να κατανείμει ταυτόχρονα και το σετ τον εντολών προς κατανομή δεν πρέπει να παραβιάζει τους περιορισμούς των

μονάδων εκτέλεσης του επεξεργαστή, ακόμα και αν αυτό σημαίνει ότι μια εντολή κατανέμεται ενώ εμφανίζεται αργότερα στο πρόγραμμα. Το να επαναλάβουμε την παραπάνω διαδικασία για κάθε εντολή μας δίνει το χρόνο κατανομής.

5.7ΤΕΧΝΙΚΕΣ ΜΕΤΑΓΛΩΤΤΙΣΗΣ ΤΟΥ INSTRUCTION LEVEL PARALLELISM

Οι μεταγλωττιστές χρησιμοποιούν μια ευρεία ποικιλία τεχνικών για να βελτιώσουν την απόδοση των μεταγλωττισμένων προγραμμάτων, συμπεριλαμβανομένων των constant propagation (διάδοση σταθερών), dead code elimination, (απαλοιφή αδρανή κώδικα) και register allocation (κατανομή καταχωρητών). Μια γενική συζήτηση για τις βελτιώσεις των μεταγλωττιστών είναι πέρα από τους σκοπούς του βιβλίου αυτού, αλλά θα καλύψουμε λεπτομερώς σε αυτό το κομμάτι το loop unrolling (αύξηση βρόγχου), έναν τρόπο βελτίωσης που αυξάνει σημαντικά τον instruction level parallelism. Κατά δεύτερο λόγο θα αναφερθούμε σε μια άλλη τεχνική μεταγλώττιση που αυξάνει την ποιότητα, την software pipelining (διοχέτευση λογισμικού).

5.7.1. LOOP UNROLLING

Οι επαναλήψεις βρόγχων τείνουν να έχουν χαμηλό instruction level parallelism, γιατί συχνά περιέχουν αλυσίδες εξαρτημένων εντολών, αλλά και εξαιτίας του περιορισμένου αριθμού των διακλαδισμένων εντολών. Η αύξηση βρόγχου αναφέρεται σε αυτούς τους περιορισμούς, μετατρέποντας ένα βρόγχο με N επαναλήψεις, σε βρόγχο με N/M επαναλήψεις, όπου κάθε επανάληψη κάνει την δουλειά των M επαναλήψεων του παλιού βρόγχου. Αυτό αυξάνει τον αριθμό των εντολών στους κλάδους δίνοντας στον μεταγλωττιστή την ευκαιρία να βρουν περισσότερες εντολές που μπορούν να εκτελεστούν παράλληλα, Ακόμα αν οι επαναλήψεις του αρχικού βρόγχου είναι ανεξάρτητες ή περιέχουν μόνο λίγες εξαρτημένους υπολογισμούς, με την αύξηση των βρόγχων δημιουργούνται πολλαπλές αλυσίδες εξαρτημένων εντολών, στη θέση της μίας που προϋπήρχε, αυξάνοντας την δυνατότητα του συστήματος να εφαρμόσει τον instruction level parallelism.

5.7.2. ΔΙΟΧΕΤΕΥΣΗ ΛΟΓΙΣΜΙΚΟΥ- SOFTWARE PIPELINING

Η αύξηση βρόγχου βελτιώνει την απόδοση αυξάνοντας τον αριθμό των ανεξάρτητων λειτουργιών μέσα στις επαναλήψεις του βρόγχου. Υπάρχει άλλη μια μέθοδος βελτίωσης , η διοχέτευση λογισμικού (software pipelining), η οποία βελτιώνει την απόδοση διανέμοντας κάθε επανάληψη του αρχικού βρόγχου σε πολλαπλές επαναλήψεις του διοχετευμένου βρόγχου, έτσι ώστε κάθε επανάληψη του νέου βρόγχου να κάνει τη δουλειά πολλαπλών επαναλήψεων του αρχικού βρόγχου. Για παράδειγμα ένας βρόγχος που φέρνει το b[i] και το c[i] από τη μνήμη, τα προσθέτει έτσι ώστε να πάρει το a[i] και γράφει το αποτέλεσμα πάλι στη μνήμη, μπορεί να μετατραπεί έτσι ώστε κάθε επανάληψη να γράφει το a[i-1] ,να υπολογίζει το a[i] με βάση τις τιμές του b[i] και του c[i] που έφερε στον προηγούμενο υπολογισμό και τελικά να φέρνει τα b[i+1] και c[i+1] από τη μνήμη για να κάνει τον επόμενο υπολογισμό. Έτσι η διαδικασία υπολογισμού του στοιχείου του a[] πίνακα διανέμεται σε τρεις επαναλήψεις του νέου βρόγχου.
Η παρεμβολή των τμημάτων διαφορετικών επαναλήψεων του βρόγχου κατά αυτό τον τρόπο αυξάνει το istruction level parallelism με τον ίδιο τρόπο που το κάνει και η αύξηση βρόγχου. Ακόμα αυξάνει τον αριθμό των εντολών μεταξύ του υπολογισμού

μια τιμής και της χρήσης της, κάνοντας την τιμή είναι διαθέσιμη εάν χρειαστεί στη συνέχεις. Πολλοί μεταφραστές συνδυάζουν τη διοχέτευση λογισμικού και την αύξηση για να βελτιώσουν τον instruction level parallelism ακόμα περισσότερο από ότι αν εφάρμοζαν κάθε τεχνική χωριστά.

ΚΕΦΑΛΑΙΟ 6^{ο}

ΣΥΣΤΗΜΑΤΑ ΜΝΗΜΗΣ

6.1 Στόχοι

Τα τελευταία κεφάλαια έχουν εξετάσει τα διάφορα στοιχεία του σχεδίου επεξεργαστών, συμπεριλαμβανομένης της οργάνωσης αρχείων καταλόγων, της αρχιτεκτονικής συνόλου οδηγίας, της διοχέτευσης, και του παραλληλισμού επιπέδων οδηγίας.Σε αυτό το κεφάλαιο, σχεδιάζουμε τη συζήτησή μας για τα συστημάτα μνήμης, η οποία θα συνεχιστεί στα επόμενα δύο κεφάλαια.

6.2 Εισαγωγή

Μέχρι τωρα, έχουμε μεταχειριστεί τα συστήματα μνήμης ως "μαύρο πεδίο" όπου ο επεξεργαστής θα μπορούσε να τοποθετήσει τα στοιχεία για να τα ανάκτηση αργότερα. Έχουμε υποθέσει ότι όλες οι διαδικασίες μνήμης παίρνουν το ίδιο χρονικό διάστημα να ολοκληρώσουν, και ότι κάθε λειτουργία μνήμης έπρεπε να τελειώσει προτού να μπορέσει να αρχίσει η επόμενη.Αρχίζοντας από αυτό το κεφάλαιο, θα ερευνήσουμε σε εκείνο το μαύρο κιβώτιο για να εξερευνήσουμε πώς τα συστήματα μνήμης εφαρμόζονται στα σύγχρονα συγκροτήματα ηλεκτρονικών υπολογιστών.

Το κεφάλαιο αυτό αρχίζει με μια συζήτηση για την λανθάνουσα κατάσταση, ρυθμοαπόδοση, και εύρος ζώνης , οι τρεις ποσότητες που χρησιμοποιούνται για να μετρήσουν την απόδοση συστημάτων μνήμης.Έπειτα, συζητάμε τις ιεραρχίες μνήμης, που εξηγούν γιατί και πώς η πολλαπλάσια τεχνολογία μνήμης χρησιμοποιείται για να εφαρμόσει ένα ενιαίο σύστημα μνήμης.Τέλος, καλύπτουμε τις τεχνολογίες μνήμης, την εξήγηση πώς τα τσιπ μνήμης εφαρμόζονται, τη διαφορά μεταξύ SRAM και DRAM, και πώς οι διαφορετικοί τρόποι πρόσβασης που βρίσκονται στα DRAM εφαρμόζονται .

6.3 Λανθάνουσα κατάσταση, ρυθμοαπόδοση, και εύρος ζώνης

Στη συζήτηση των επεξεργαστών διοχέτευσης, χρησιμοποιήσαμε τους όρους λανθάνουσα κατάσταση και τη ρυθμοαπόδοση για να περιγράψουμε το χρόνο που λαμβάνεται για να ολοκληρώσουμε μια μεμονωμένη λειτουργία και το ποσοστό στο οποίο οι διαδικασίες μπορούν να ολοκληρωθούν.Αυτοί οι όροι χρησιμοποιούνται επίσης στη συζήτηση των συστημάτων μνήμης και έχουν την ίδια έννοια όπως στη συζήτηση επεξεργαστών μας. Ένας πρόσθετος όρος που χρησιμοποιείται στη συζήτηση των συστημάτων μνήμης είναι το εύρος ζώνης, το οποίο περιγράφει το συνολικό ποσοστό στο οποίο τα στοιχεία μπορούν να κινηθούν μεταξύ του επεξεργαστή και του συστήματος μνήμης.Το εύρος ζώνης μπορεί να θεωρηθεί ως προϊόν της ρυθμοαπόδοσης και του ποσού στοιχείων που παραπέμπονται από κάθε λειτουργία μνήμης .

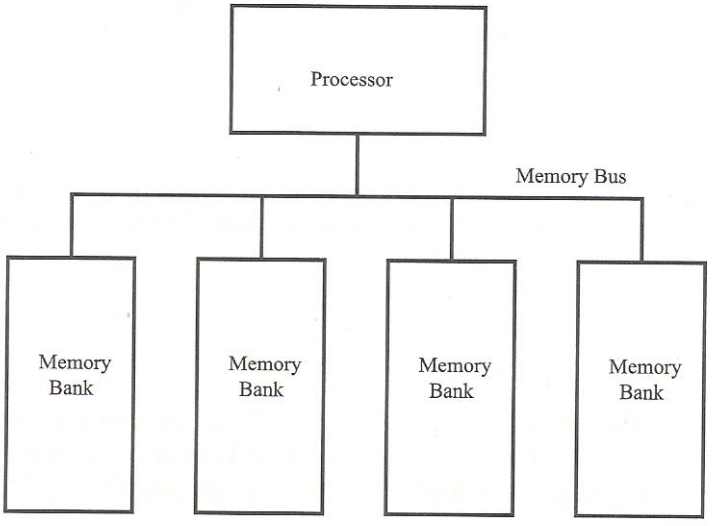
6.3.1 Διοχέτευση, παραλληλισμός, και προφόρτισμα

Εάν όλες οι διαδικασίες μνήμης εκτελέσθηκαν διαδοχικά, ο υπολογισμός της λανθάνουσας κατάστασης και του εύρους ζώνης ενός συστήματος μνήμης θα ήταν απλός. Εντούτοις, πολλά συστήματα μνήμης σχεδιάζονται με τρόπους που κάνουν τη σχέση μεταξύ της λανθάνουσας κατάστασης και του εύρους ζώνης πιό σύνθετη.Τα συστήματα μνήμης μπορούν να διοχετευθούν με τον ίδιο τρόπο που και οι

επεξεργαστές διοχετεύονται, επιτρέποντας στις διαδικασίες να επικαλύψουν την εκτέλεση για να βελτιώσουν τη ρυθμοαπόδοση.Επίσης, πολλές τεχνολογίες μνήμης απαιτούν ένα ορισμένο ποσό μη απασχόλησης χρονικής περιόδου μεταξύ των προσβάσεων μνήμης.Αυτή η περίοδος χρησιμοποιείται για προετοίμασια, ή προφόρτισμα,του κυκλώματος για την επόμενη πρόσβαση.Το προφόρτισμα του κυκλώματος κάνει μερική από την εργασία της πρόσβασης της μνήμης προτού να φθάσει η διεύθυνση. Αυτό μειώνει την καθυστέρηση από το χρόνο που μια διεύθυνση στέλνεται στο σύστημα μνήμης έως ότου ολοκληρώσουν οι διαδικασίες μιας μνήμης.Εάν το σύστημα μνήμης είναι μη απασχόλημένο για ένα μεγάλο μέρος του χρόνου, κάνοντας το προφόρτισμα στο τέλος κάθε λειτουργίας μνήμης βελτιώνει την απόδοση επειδή εκεί συνήθως δεν είναι μια άλλη λειτουργία που περιμένει να χρησιμοποιήσει το σύστημα μνήμης.Εάν το σύστημα μνήμης χρησιμοποιείται τις περισσότερες φορές, το ποσοστό στο οποίο οι διαδικασίες μπορούν να ολοκληρωθούν καθορίζεται από το ποσό της λανθάνουσας κατάστασης μνήμης και του προφορτισμένου χρόνου.

Ένας άλλος τρόπος που οι σχεδιαστές βελτιώνουν την απόδοση των συστημάτων μνήμης είναι η σχεδίαση αυτών για να υποστηρίξουν τις πολλαπλάσιες αναφορές μνήμης παράλληλα. Αυτό είναι συνηθέστερο να γίνεται με την ένωση των πολλαπλάσιων μνημών με την αρτηρία μνήμης του επεξεργαστή, περισσότερο απο μια αναφορά μνήμης για να αρχίσει ή να τελειώσει συγχρόνως, δεδομένου ότι μόνο ένα αίτημα μπορεί να χρησιμοποιήσει τη δίοδο σε οποιοδήποτε χρόνο. Εντούτοις, ο επεξεργαστής μπορεί να στείλει τα αιτήματα μνήμης στις αδρανής μνήμες ενώ περιμένει άλλα αίτηματα να ολοκληρώθουν, η ρύθμιση αυξάνει το ποσοστό στο οποίο τα αιτήματα μνήμης μπορούν να αντιμετωπιστούν χωρίς ο αριθμός I/O ακροδεκτών που απαιτείται στο τσιπ επεξεργαστών, το οποίο θα αύξανει το κόστος του επεξεργαστή.

Τα συστήματα που υποστηρίζουν τα παράλληλα αίτημα μνήμης διαιρούνται σε δύο τύπους.Τα επαναλαμβανώμενα συστήματα μνήμης παρέχουν πολλαπλάσια αντίγραφα ολόκληρης της μνήμης.Αυτό σημαίνει ότι κάθε αντίγραφο της μνήμης μπορεί να χειριστεί οποιοδήποτε αίτημα μνήμης, αλλά αυτό αυξάνει το ποσό της μνήμης που απαιτείται από έναν παράγοντα ίσο με τον αριθμό των αντιγράφων.Για να κρατήσει το περιεχόμενο κάθε μνήμης ίδιο ,όλες οι αποθηκευμένες διαδικασίες πρέπει να στειλουν αντίγραφο της μνήμης, φτίαχνουν αποθήκες περισσότερο ακριβές απο τα φορτία από την άποψη του ποσού εύρους ζώνης που καταναλώνουν .



ΣΧΗΜΑ6-1 banked/replicated memory system

Ο πιό κοινός τύπος παράλληλου συστήματος μνήμης είναι ένα κατατεθειμένο σύστημα μνήμης.Σε ένα κατατεθειμένο σύστημα μνήμης, το στοιχείο είναι διαιρεμένο, ή εναλλακτικό ,στις μνήμες έτσι ώστε κάθε μνήμη να περιέχει μόνο ένα μέρος των στοιχείων.Χαρακτηριστικά, μερικά από τα bits της διεύθυνσης χρησιμοποιούνται για να επιλέξουν σε ποια τράπεζα δεδομένων ένα δεδομένο ανήκει.Παραδείγματος χάριν, στο σύστημα που διευκρινίστηκε στο σχήμα 6.1,τα bytes των οποίων δύο χαμηλές διευθύνσεις bits είναι 0b00 κ πρέπει να τοποθετηθούν στην επόμενη τράπεζα, και τα λοιπά.Διαδοχικά,δύο άλλα bits διευθύνσεων θα μπορούσαν να χρησιμοποιηθούν για να επιλέξουν μια τράπεζα.Γενικά, σχετικά οι χαμηλές διευθύνσεις bits χρησιμοποιούνται για να επιλέξουν την τράπεζα, έτσι ώστε οι αναφορές στις διαδοχικές διευθύνσεις μνήμης να πηγαίνουν στις διαφορετικές τράπεζες .

Τα κατατεθειμένα συστήματα μνήμης έχουν το πλεονέκτημα ότι δεν απαιτούν άλλη μνήμη από ένα ισοδύναμο σύστημα μνήμης με μόνο μια μνήμη, και είναι μόνο απαραίτητο να στελνούν διαδικασίες αποθήκευσεις στην τράπεζα που περιέχει τα στοιχεία που γράφονται.¨Ομως,υποφέρουν από το πρόβλημα ότι μερικά ζευγάρια των αναφορών μνήμης στοχεύουν στην ίδια τράπεζα, που απαιτεί μια από τις διαδικασίες να περιμένει έως ότου ολοκληρώθει άλλη.Στις περισσότερες περιπτώσεις, η πρόσθετη μνήμη που απαιτείται για να εφαρμόσει ένα πολλαπλό σύστημα μνήμης είναι ένα μεγαλύτερο μειονέκτημα απο το εύρος ζώνης που χάνεται λόγω των συγκρούσεων για τις τράπεζες μνήμης, έτσι τα κατατεθειμένα συστήματα μνήμης είναι πιό κοινά απο τα πολλαπλά συστήματα μνήμης.

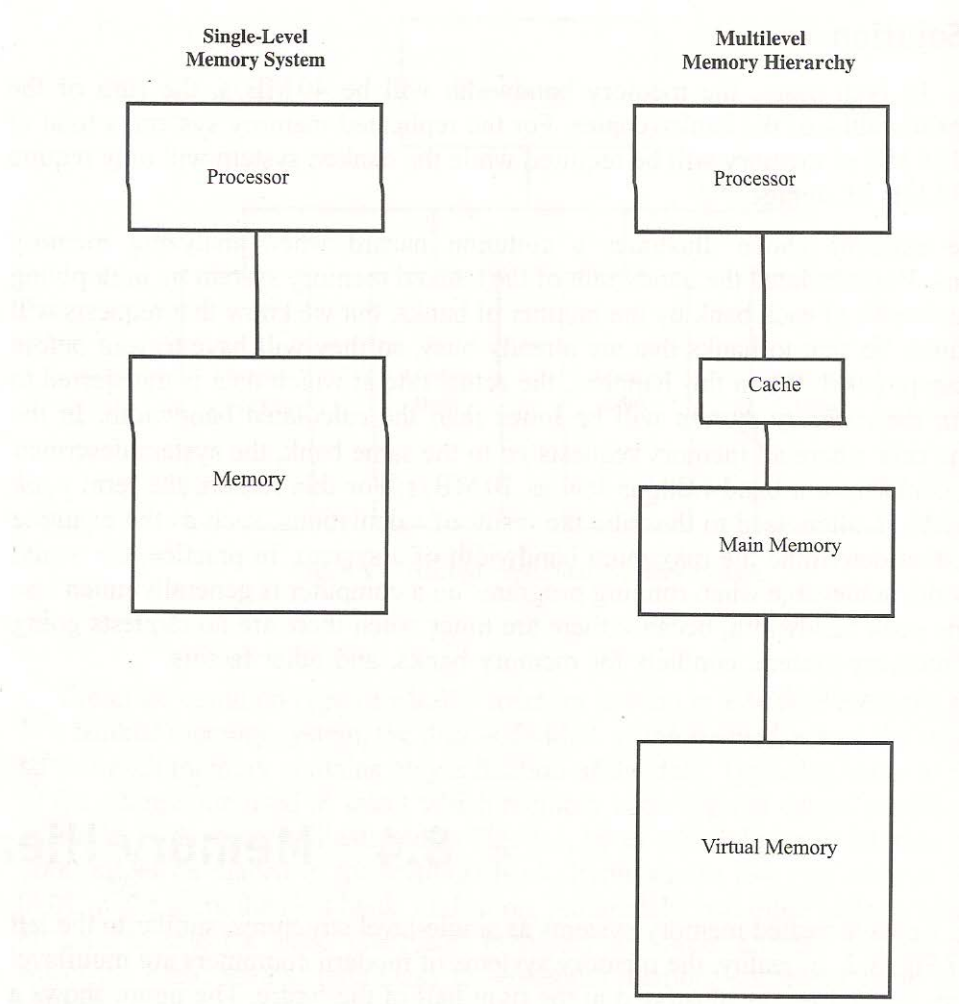
Το παραπάνω παράδειγμα επεξηγεί έναν κοινό κίνδυνο κατά την ανάλυση των συστημάτων μνήμης.Υπολογίσαμε το εύρος ζώνης του κατατεθειμένου συστήματος μνήμης με τον πολλαπλασιασμό του εύρους ζώνης κάθε τράπεζας με τον αριθμό τραπεζών, αλλά ξέρουμε ότι τα αιτήματα θα σταλούν μερικές φορές στις τράπεζες που είναι ήδη πολυάσχολες, έτσι θα πρέπει να περιμένουν προτού να μπορέσουν να προχωρήσουν.Όταν αυτό συμβαίνει, το πραγματικό ποσοστό στο οποίο το στοιχείο μεταφέρεται προς ή από το σύστημα μνήμης θα είναι χαμηλότερο από το υπολογισμένο εύρος ζώνης.Στην ακραία περίπτωση όπου όλα τα αιτήματα μνήμης πηγαίνουν στην ίδια τράπεζα, το σύστημα που περιγράφηκε ανωτέρω θα μπορούσε να έχει ένα εύρος ζώνης τόσο χαμηλό όπως 10 MB/s.Για αυτό το λόγο, το μέγιστο εύρος ζώνης όρου χρησιμοποιείται συχνά για να περιγράψει το αποτέλεσμα των υπολογισμών, όπως το παραπάνω παράδειγμα , οι οποίοι καθορίζουν το μέγιστο εύρος ζώνης ενός συστήματος.Στην πράξη, το πραγματικό επιτεύξιμο εύρος ζώνης όταν τα προγραμμάτα τρέχουν σε έναν υπολογιστή είναι γενικά πολύ λιγότερο από το μέγιστο εύρος ζώνης, επειδή υπάρχουν χρόνοι όταν δεν υπάρχει κανένα αίτημα που πηγαίνει στο σύστημα μνήμης, συγκρούεται για τις τράπεζες μνήμης, και άλλους παράγοντες.

6.4 Ιεραρχίες μνήμης

Μέχρι τώρα, έχουμε μεταχειριστεί τα συστήματα μνήμης ως μονοεπίπεδες δομές, παρόμοιες με το αριστερό μισό του σχήματος 6-2.Στη πραγματικότητα, τα συστήματα μνήμης των σύγχρονων υπολογιστών είναι πολλαπλής στάθμης ιεραρχίες μνήμης.Ο αριθμός παρουσιάζει ιεραρχία μνήμης τριών επιπέδων, που αποτελείται από μια κρύφη μνήμη, μια κύρια μνήμη, και μια εικονική μνήμη.

Ο αρχικός λόγος που τα συστήματα μνήμης κατασκευάζονται ως ιεραρχίες είναι ότι το κόστος ανά bit μιας τεχνολογίας μνήμης είναι γενικά ανάλογο προς την ταχύτητα της τεχνολογίας.Οι γρήγορες μνήμες, όπως στατική RAMs (SRAMs), τείνουν να έχουν ένα υψηλό κόστος ανά bit , καθιστώντας αυτο απαγορευτικά ακριβό για να κατασκευάσουν τη μνήμη ενός υπολογιστή εξ ολοκλήρου από αυτές τις συσκευές.Χαμηλοτερες τεχνολογίες, όπως η δυναμική RAM (DRAM), είναι λιγότερο ακριβες, καθιστώντας το πρακτικό να κατασκευάσουν μεγαλύτερες μνήμες που χρησιμοποιούν αυτές οι τεχνολογίες.

Σε μια ιεραρχία μνήμης, τα επίπεδα που είναι κοντά στον επεξεργαστή, όπως η κρύφη μνήμη που παρουσιάζεται στο σχήμα, περιέχουν ένα σχετικά μικρό ποσό μνήμης που εφαρμόζεται σε μια γρήγορη τεχνολογία μνήμης για να δώσει έναν χαμηλό χρόνο πρόσβασης.Προχωρώντας κάτω στην ιεραρχία, κάθε επίπεδο περιέχει περισσότερη αποθήκευση και παίρνει περισσότερο για την πρόσβαση από το επίπεδο επάνω από αυτο.Ο στόχος μιας ιεραρχίας μνήμης είναι η διατήρηση των στοιχείων που θα παραπεμφθούν πιό πολύ από ένα πρόγραμμα στα κορυφαία επίπεδα της ιεραρχίας, έτσι ώστε τα περισσότερα αιτήματα μνήμης να μπορούν να αντιμετωπιστούν από το κορυφαίο επίπεδο ή τα επίπεδα.Αυτό οδηγεί σε ένα σύστημα μνήμης που έχει έναν μέσο χρόνο πρόσβασης παρόμοιο με το χρόνο πρόσβασης του γρηγορότερου επιπέδου, αλλά με ένα μέσο κόστος ανά bit παρόμοιο με αυτό του πιό αργού επιπέδου.



ΣΧΗΜΑ6-2 memory hierachies

Γενικά, δεν είναι δυνατό να προβλεφθεί ποιες θέσεις μνήμης θα προσεγγιστούν πολύ συχνά, έτσι οι υπολογιστές χρησιμοποιούν ένα βασισμένο στην απαίτηση σύστημα για να ορίσει ποια στοιχεία να κρατήσουν τα κορυφαία επίπεδα της ιεραρχίας.Όταν ένα αίτημα μνήμης στέλνεται στην ιεραρχία, το κορυφαίο επίπεδο ελέγχει για να δει εάν αυτό περιέχει τη διεύθυνση.Σε αυτή την περίπτωση, το
αίτημα ολοκληρώνει.Αν όχι, το επόμενο χαμηλότερο επίπεδο ελέγχεται, με τη διαδικασία να επαναλαμβάνεται έως ότου το στοιχείο βρεθεί ή το κατώτατο επίπεδο της ιεραρχίας επιτυγχάνεται, η οποία είναι εγγυημένη για να περιέχει τα στοιχεία.

Εάν ένα αίτημα μνήμης δεν μπορεί να αντιμετωπιστεί από το κορυφαίο επίπεδο στην ιεραρχία, ο φραγμός των διαδοχικών θέσεων που περιέχουν την παραπεμφθείσα διεύθυνση αντιγράφεται από το πρώτο επίπεδο που περιέχει τη διεύθυνση σε κάθε επίπεδο επάνω από αυτό.Αυτό γίνεται για δυο λόγους. Ο πρώτος είναι ότι πολλές τεχνολογίες αποθήκευσης, όπως η τροποποίηση σελίδων DRAms που θα συζητηθεί αργότερα στο κεφάλαιο αυτό και οι σκληροί δίσκοι, επιτρέπουν στις πολλαπλάσιες διαδοχικές λέξεις των στοιχείων να διαβαστούν ή να γραφτούν σε λιγότερο χρόνο

από ένας ίσος αριθμός τυχαία τοποθετημένων λέξεων, που κάνει αυτο γρηγορότερο για να μεταφέρει έναν φραγμό multibyte στοιχείων στα κορυφαία επίπεδα της ιεραρχίας από το να προσκομίσουν κάθε byte στο φραγμό από τα χαμηλότερα επίπεδα της ιεραρχίας χωριστά.Δεύτερον, τα περισσότερα προγράμματα επιδεικνύουν την *τοποθεσία της αναφοράς* - αναφορές μνήμης που εμφανίζονται κοντά στο χρόνο τείνουν να έχουν διευθύνσεις που είναι κοντά η μια στην άλλη, που καθιστά αυτο πιθανό ότι οι άλλες διευθύνσεις μέσα σε έναν φραγμό θα παραπεμφθούν σύντομα μετά από τη πρώτη αναφορά σε μια διεύθυνση στο φραγμό.

Αφού η πιθανότητα ότι κάθε διεύθυνση είναι μέσα στο φραγμό θα είναι αρκετά υψηλή, η χρησιμοποίηση των multibyte φραγμών μειώνει το μέσο χρόνο πρόσβασης, επειδή για να προσκομίσει το φραγμό παίρνει το λιγότερο χρόνο που χρησιμοποιει γαι να προσκομισει κάθε λέξη μέσα στο φραγμό χωριστά.Τα διαφορετικά επίπεδα στην ιεραρχία μνήμης θα έχουν συχνά διαφορετικά μεγέθη φραγμών, ανάλογα με τα χαρακτηριστικά των επιπέδων κάτω από αυτα στην ιεραρχία.Παραδείγματος χάριν, οι κρύφες μνήμες τείνουν να έχουν τα μεγέθη φραγμών περίπου 64 bytes , ενώ οι κύριες μνήμες έχουν γενικά τα μεγέθη φραγμών περίπου 4 KB, επειδή ο χρόνος για να προσκομιστεί ένας μεγάλος φραγμός στοιχείων από την εικονική μνήμη είναι μόνο ελαφρώς πιό μακροχρόνιος από το χρόνο που χρειάζεται για να προσκομιστεί 1 byte , ενώ ο χρόνος για να προσκομιστεί ένας φραγμός στοιχείων στην κρύφη μνήμη οπου η κύρια μνήμη είναι πολύ πιό στενός από το χρόνο που χρειάζεται για να προσκομιστεί κάθε byte χωριστά.

6.4.1 Επίπεδα στην ιεραρχία

Στην ιεραρχία μνήμης που παρουσιάστηκε στο σχήμα 6-2, τα διαφορετικά επίπεδα της ιεραρχίας είχαν συγκεκριμένα ονόματα.Αυτό προκύπτει από το γεγονός ότι τα διαφορετικά επίπεδα μιας ιεραρχίας μνήμης τείνουν να εφαρμοστούν πολύ διαφορετικά, έτσι οι αρχιτέκτονες υπολογιστών χρησιμοποιούν διαφορετικούς όρους για να τους περιγράψουν.Το κορυφαίο επίπεδο ή τα επίπεδα της ιεραρχίας αναφέρεται χαρακτηριστικά ως κρύφη μνήμη.Οι κρύφες μνήμηες εφαρμόζονται γενικά χρησιμοποιώντας SRAM, και οι περισσότεροι σύγχρονοι υπολογιστές έχουν τουλάχιστον δύο επίπεδα κρύφης μνήμης στην ιεραρχία μνήμης τους.Οι κρύφες μνήμες έχουν το υλικό για την παρακολουθήση των διευθύνσεων που αποθηκεύονται σε αυτές , τείνουν να είναι σχετικά μικρές, και να έχουν τα μικρά μεγέθη φραγμών.Συνήθως 32 έως 128 bytes.Η κύρια μνήμη ενός υπολογιστή κατασκευάζεται γενικά από τη DRAM, στηρίζεται στο λογισμικό για να παρακολουθήσει τις διευθύνσεις που περιλαμβάνονται σε αυτην , και έχει ένα μεγάλο μέγεθος φραγμών, συχνά τα διάφορα kilobyte.Τέλος, η εικονική μνήμη εφαρμόζεται συνήθως χρησιμοποιώντας τους δίσκους και περιέχει όλα τα στοιχεία στο σύστημα μνήμης.Το κεφάλαιο 7 συζητά για τις κρύφες μνήμες λεπτομερέστερα, ενώ το κεφάλαιο 8 καλύπτει τα συστήματα εικονικής μνήμης.

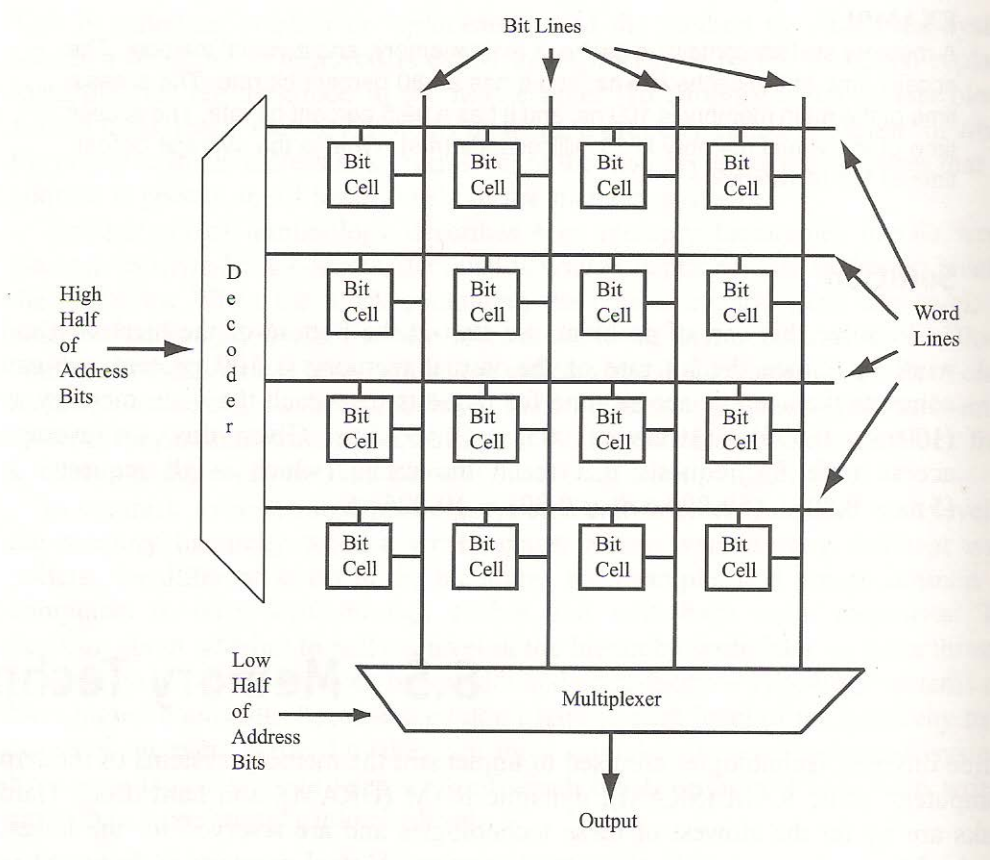
6.5 Τεχνολογίες μνήμης

Τρεις διαφορετικές τεχνολογίες χρησιμοποιούνται για να εφαρμόσουν τα συστήματα μνήμης των σύγχρονων υπολογιστών: στατική RAM (SRAM), δυναμική RAM (DRAM), και σκληροί δίσκοι. Οι σκληροί δίσκοι είναι κατά πολύ πιό αργοί αυτών των τεχνολογιών και είναι διατηρημένοι για το χαμηλότερο επίπεδο συστήματος μνήμης, η εικονική μνήμη. Η εικονική μνήμη συζητείται λεπτομερέστερα στο κεφάλαιο 8.Η SRAM και η DRAM είναι μέχρι έναν παράγοντα 1.000.000 γρηγορότερές από την βασισμένη στον δίσκο μνήμη και είναι οι τεχνολογίες που χρησιμοποιούνται για να εφαρμόσουν τις κρυφές και τις κύριες μνήμες σχεδόν όλων των υπολογιστών .

6.5.1 Οργάνωση των τσιπ μνήμης

Τα τσιπ μνήμης SRAM και DRAM έχουν την ίδια βασική δομή, η οποία παρουσιάζεται στο σχήμα 6-3.Το στοιχείο αποθηκεύεται σε ένα ορθογώνιο πίνακα απο κελία bits, κάθε ένα από τα οποία φυλάσσει 1 bit δεδομένων. Για να διαβάσει τα στοιχεία από τον πίνακα, η μισή από τη διεύθυνση που διαβάζεται (γενικά τα υψηλού-βαθμού bits) τροφοδοτείται σε έναν αποκωδικοποιητή.Ο αποκωδικοποιητής βεβαιώνει (κινήσεις υψηλές) τη γραμμή λέξης που αντιστοιχεί στην τιμή των bits εισόδου του, η οποία αναγκάζει όλα τα κελία bits στην αντίστοιχη σειρά να οδηγήσουν τις τιμές τους επάνω στις γραμμές bits όπου αυτές είναι συνδεδεμένες.Το άλλο μισό της διεύθυνσης χρησιμοποιείται έπειτα ως εισόδος σε έναν πολυπλέκτη που επιλέγει την κατάλληλη γραμμή bits και οδηγεί αυτο στην έξοδο επάνω στην έξοδο των ακροδεκτών και των τσιπ.Για να αποθηκεύσει τα στοιχεία όσον αφορά το τσιπ, η ίδια διαδικασία χρησιμοποιείται, εκτός από την τιμή που γράφεται που οδηγείται στην κατάλληλη γραμμή bits και γράφεται στο επιλεγμένο κελί bits.

Τα περισσότερα τσιπ μνήμης παράγουν περισσότερο από 1 bit εξόδου.Αυτό που έγινε είτε με την οικοδόμηση διάφορων σειρών κελίων bits, κάθε μια από τις οποίες παράγει ένα bit εξόδου , είτε με το σχεδιασμό ενός πολυπλέκτη που επιλέγει τις εξόδους απο αρκετές από τις γραμμές bits και τα οδηγεί στα αποτελέσματα του τσιπ.



ΣΧΗΜΑ6-3 memory chip organization

Η ταχύτητα ενός τσιπ μνήμης καθορίζεται από διάφορους παράγοντες συμπεριλαμβανομένου του μήκους των γραμμών bits και λέξης και πώς είναι τα κελία bits.Οι πιό μακροχρόνιες γραμμές bits και λέξης έχουν τις υψηλότερες ικανότητες και τις αντιστάσεις, έτσι πέρνει περισσότερο για να οδηγήσει ένα σήμα

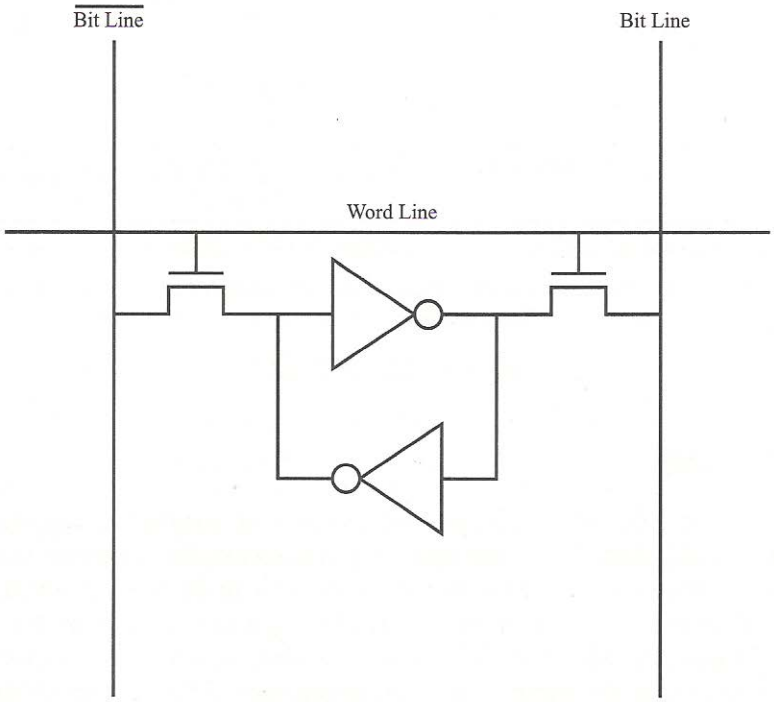
σε αυτά τα καλώδια όσο το μήκος τους αυξάνεται.Για αυτόν τον λόγο, πολλά σύγχρονα τσιπ μνήμης κατασκευάζοντι από πολλές μικρές σειρές κελίων bits για να κρατήσουν τις γραμμές λέξης και bits σύντομες.

Οι τεχνικές που χρησιμοποιούνται στην κατασκευή των κελίων bits έχουν επιπτώσεις στην ταχύτητα του τσιπ μνήμης επειδή έχουν επιπτώσεις στο πόσο ρεύμα είναι διαθέσιμο για να οδηγήσει την έξοδο του κελίου bits επάνω στις γραμμές bits , το οποίο καθορίζει πόσο καιρό παίρνει στη διαδόση της έξοδου του κελίου bits στον πολυπλέκτη.Όπως θα δούμε στα επόμενα δύο τμήματα, τα κελία bits SRAM μπορούν να οδηγήσουν πιό συχνά από τα κελία bits DRAM, ο οποίος είναι ένας από τους κύριους λόγους για τους οποίους οι SRAMs τείνουν να είναι πολύ γρηγορότερές από τις DRAM .

6.5.2 SRAMS

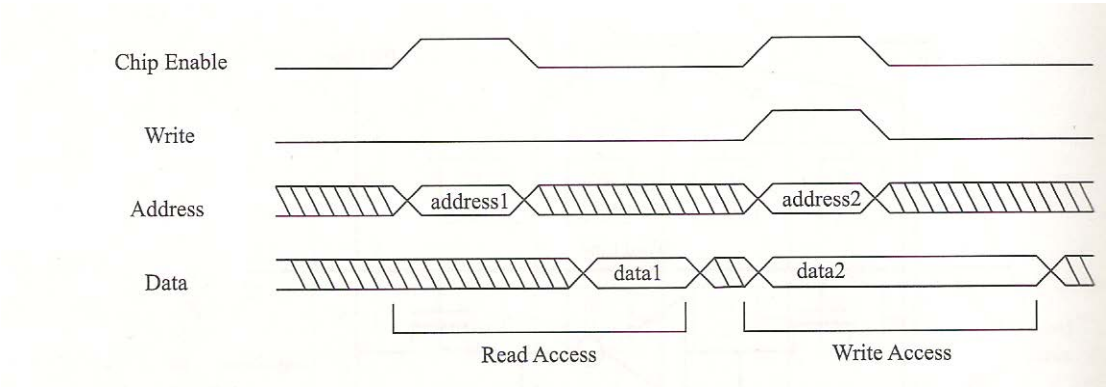
Η κύρια διαφορά μεταξύ SRAMs και DRAMs είναι στο πώς τα κελία των bit τους κατασκευάζονται. Όπως φαίνεται στο σχήμα 6-4, ο πυρήνας ενός κελίου SRAM bit αποτελείται από δύο αναστροφείς συνδεμένος σε μια back-to-back διαμόρφωση .Μόλις τοποθετηθεί μια τιμή στο κελί bits, οι δομήμένοι δαχτυλιδιοι των δύο αναστρέφεων θα διατηρήσουν την τιμή κατά τρόπο αόριστο, επειδή κάθε εισοδος του αναστροφέα είναι διαφορετική απο τους άλλους.Αυτός είναι ο λόγος για τον οποίο οι SRAMs καλούνται στατικές RAMs,οι τιμές που αποθηκεύονται στην RAM παραμένουν εκεί εφ' όσον εφαρμόζεται η δύναμη στη συσκευή.Οι DRAMs, αφ' ετέρου, θα χάσουν τις αποθηκευμένες τιμές τους κατά τη διάρκεια του χρόνου αυτός είναι ο λόγος για τον οποίο είναι γνωστές ως δυναμικές RAMs.

Για να διαβάσει μια τιμή από το κελί bits , η γραμμή λέξης είναι οδηγημένη υψηλά, η οποία αναγκάζει τις δύο κρυσταλλολυχνίες(transistors) να συνδέσουν τα αποτελέσματα των αναστροφέων με τη γραμμή bits και τη αντίστροφη γραμμή bits.Αυτά τα σήματα μπορούν έπειτα να διαβαστούν από τον πολυπλέκτη και να σταλούν στην έξοδο του τσιπ. Το γράψιμο ενός κελίου bits SRAM ολοκληρώνεται με τη βεβαίωση της γραμμής λέξης και την οδήγηση των κατάλληλων τιμών στη γραμμή bits και την αντίστροφη γραμμή bits.Εφ' όσον η συσκευή που οδηγεί τη γραμμή bits είναι ισχυρότερη από τον αναστροφέα, οι τιμές στη γραμμή bits θα εξουσιάσουν την τιμή που αποθηκεύεται αρχικά στο κελί bits , και θα αποθηκευτούν στο κελί bits όταν η γραμμή λέξης είναι επιβεβαιωμένη.



ΣΧΗΜΑ 6-4 SRAM bit cell

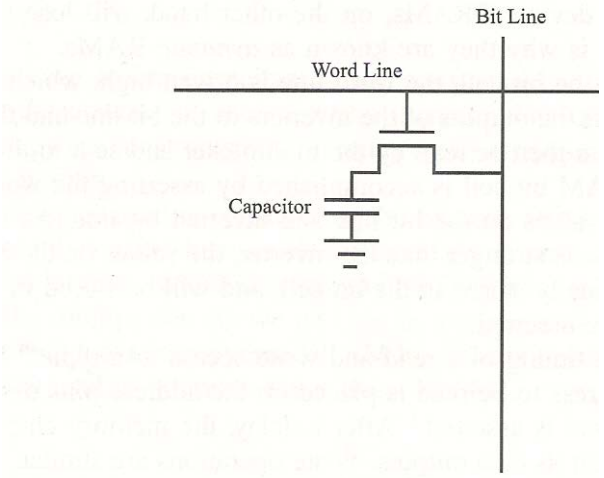
Το σχήμα 6-5 παρουσιάζει το συγχρονισμό της πρόσβασης διαβασματός και γράψιματος σε ένα χαρακτηριστικό SRAM.Για να διαβάσει τη συσκευή, η διεύθυνση που διαβάζεται τοποθετείται στους ακροδέκτες διευθύνσεων της συσκευής, και το τσιπ βεβαιώνει επιτρέπει το σήμα .Μετά από μια καθυστέρηση, το τσιπ μνήμης τοποθετεί το περιεχόμενο της διεύθυνσης στα αποτελέσματα στοιχείων του.Γράψτε ότι οι διαδικασίες είναι παρόμοιες, εκτός από το ότι ο επεξεργαστής τοποθετεί τα στοιχεία για να είναι στους ακροδέκτες στοιχείων την ίδια στιγμή που η διεύθυνση στέλνεται στο τσιπ, και το σήμα ελέγχου χρησιμοποιείται για να δείξει ότι το γράψιμο έγινε.



ΣΧΗΜΑ6-5 SRAM access

6.5.3 DRAM

Το σχήμα 6-6 παρουσιάζει το κελί bits DRAM.Αντί ενός ζευγαριού αναστροφέων, ένας πυκνωτής χρησιμοποιείται για να αποθηκεύσει τα στοιχεία του κελίου bits.Όταν η γραμμή λέξης βεβαιώνεται, ο πυκνωτής συνδέεται με τη γραμμή bits, που επιτρέπει στην αξία που αποθηκεύεται στο κελί να διαβαστεί απο τον έλεγχο της τάση που αποθηκεύεται στον πυκνωτή ή που γράφεται με την τοποθέτηση μιας νέας τάσης στον πυκνωτή.Αυτός ο αριθμός επεξηγεί γιατί οι DRAM έχουν γενικά πολύ μεγαλύτερες περιεκτικότητες από τις SRAMs που κατασκευάζονται στην ίδια τεχνολογία επεξεργασίας: οι SRAMs απαιτούν πολλές περισσότερες συσκευές για να εφαρμόσουν ένα κελί bits.Κάθε αναστροφέας τυπικά απαιτεί δύο κρυσταλλολυχνίες, για συνολικά έξι κρυσταλλολυχνίες στο κελί bits (μερικές εφαρμογές χρησιμοποιούν κάπως περισσότερες ή λιγότερες κρυσταλλολυχνίες). Αντίθετα, ένα κελί bits DRAM απαιτεί μόνο μια κρυσταλλολυχνία και έναν πυκνωτή, ο οποίος λαμβάνει πολύ λιγότερο διάστημα στο τσιπ.



ΣΧΗΜΑ6-6 DRAM bit cell

Τα σχήματα 6-4 και 6-6 επίσης παρουσιάζουν γιατί οι SRAMs είναι χαρακτηριστικά πολύ γρηγορότερες από τις DRAM.Στις DRAM, ο πυκνωτής συνδέεται με τη γραμμή bits όταν βεβαιώνεται η γραμμή λέξης, η οποία είναι ένα πιό αδύνατο σήμα από αυτό που παράγεται από τους αναστροφείς σε ένα κελί bits SRAM.Κατά συνέπεια, παίρνει πολύ περισσότερο για την έξοδο ενός κελίου bits DRAM που οδηγείται επάνω στη γραμμή bits του που παίρνει για ένα κελί bits SRAM για να οδηγήσει μια ισοδύναμη γραμμή bits.

6.5.4 Ανανέωση DRAM

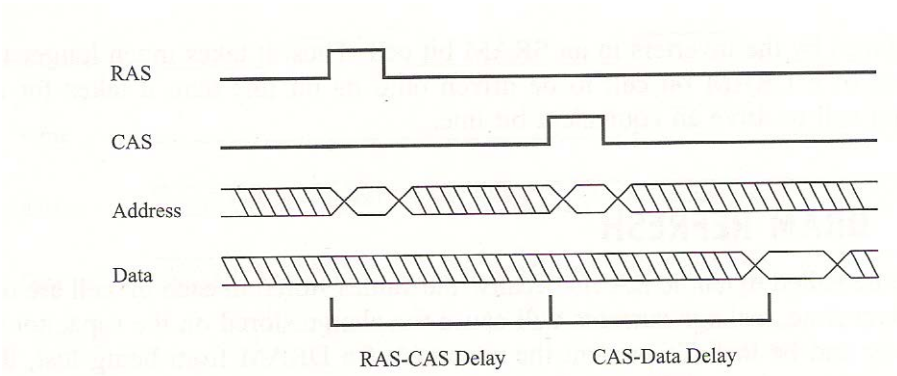
Οι DRAM καλούνται δυναμικές RAMs επειδή οι τιμές που αποθηκεύονται σε κάθε κελί bits δεν είναι σταθερές.Κατά τη διάρκεια του χρόνου, τα ρεύματα διαρροής θα αναγκάσουν τη δαπάνη που αποθηκεύεται στον πυκνωτή να εκκενώσει και να χαθει. Για να αποτρέψει το περιεχόμενο μιας DRAM από την απώλεια,η DRAM πρέπει να αναζωογονηθεί.Ουσιαστικά ,μια ανανεωμένη λειτουργία διαβάζει το περιεχόμενο κάθε κελίου bits σε μια σειρά στη σειρά του κελίου bits, και έπειτα αυτή γράφει την ίδια τιμή πίσω στο κελί bits, ξανααποθηκεύει αυτες στην αρχική τους τιμή.Εφ' όσον αναεώνεται κάθε σειρά σε μια DRAM αρκετά συχνά όπου κανένας από τους

πυκνωτές του δεν χρεώνει την αποσύνθεση αρκετά χαμηλά οπου το υλικό παρερμηνεύει τις τιμές που αποθηκεύονται στη σειρά, η DRAM μπορεί να κρατήσει το περιεχόμενό της κατά τρόπο αόριστο .Μια από τις προδιαγραφές σε ένα τσιπ DRAM είναι η ανανέωση χρόνου,η οποία είναι πόσο συχνά μια σειρά μπορεί να πάει χωρίς να ανανεωθεί προτού να είναι σε κίνδυνο το περιεχόμενό της.

6.5.5 Συγχρονισμοί πρόσβασης DRAM

Το σχήμα 6-7 παρουσιάζει το συγχρονισμό μιας λειτουργίας ανάγνωσης σε μια χαρακτηριστική DRAM.Αντίθετα από τη SRAMs, η εισοδος διευθύνσεων σε μία DRAM διαιρείται σε δύο μέρη, διεύθυνση σειρών και διεύθυνση στηλών, τα οποία στέλνονται στη DRAM σε χωριστές διαδικασίες.Χαρακτηριστικά, τα υψηλά bits μιας διεύθυνσης μνήμης χρησιμοποιούνται για τη διεύθυνση σειρών, και τα χαμηλά bits χρησιμοποιούνται για τη διεύθυνση στηλών.Όπως αναμένεται από το όνομα, η διεύθυνση σειρών επιλέγει τη σειρά του πίνακα DRAM, ενώ η διεύθυνση στηλών επιλέγει ένα bit ή σύνολο bit από εκείνη την σειρά .

Το σήμα RAS(Row Address Strobe) δείχνει ότι μια διεύθυνση σειρών στέλνεται, και το σήμα CAS (Column Address Storbe) δείχνει ότι μια διεύθυνση στήλης στέλνεται.Ο συνολικός χρόνος για να διαβαστεί η DRAM είναι το ποσό της καθυστέρησης της RAS-CAS και η CAS - καθυστέρηση στοιχείων .Οι διαδικάσιες γραψίματος έχουν παρόμοιους συγχρονισμούς, αλλά το στοιχείο που γράφεται οδηγείται γενικά στους ακροδέκτες στοιχείων ταυτόχρονα με τη διεύθυνση στηλών.



ΣΧΗΜΑ6-7 DRAM access

Η αποστολή της διεύθυνσης στην DRAM σε δυο μέρη μειώνει τον αριθμό ακροδεκτών διευθύνσεων που απαιτείται στη DRAM, επειδή οι ιδιοι ακροδέκτες μπορούν να χρησιμοποιηθούν για τη σειρά και τι στήλες διευθύνσεων .Διαιρώντας τις διευθυνσεις σε αυτά τα δύο μέρη δεν αυξάνουν σημαντικά το χρόνο πρόσβασης της DRAM, επειδή η διεύθυνση σειρών επιλέγει τη σειρά του κελίου bits του οποίου τα περιεχόμενα οδηγουνται στις γραμμές bits,ενώ η διεύθυνση στηλών επιλέγει ποια γραμμή bit οδηγείται στην έξοδο.Επομένως, η διεύθυνση στηλών δεν απαιτείται από τη DRAM έως ότου τα κελία bits έχουν οδηγηθεί στις εξόδουσ τους επάνω στις γραμμές bits, έτσι η αποστολή του στη DRAM μετά από τη διεύθυνση σειρών δεν αυξάνει το χρόνο πρόσβασης .

6.5.6 Τροποποιήση σελίδων και νεότερες DRAM

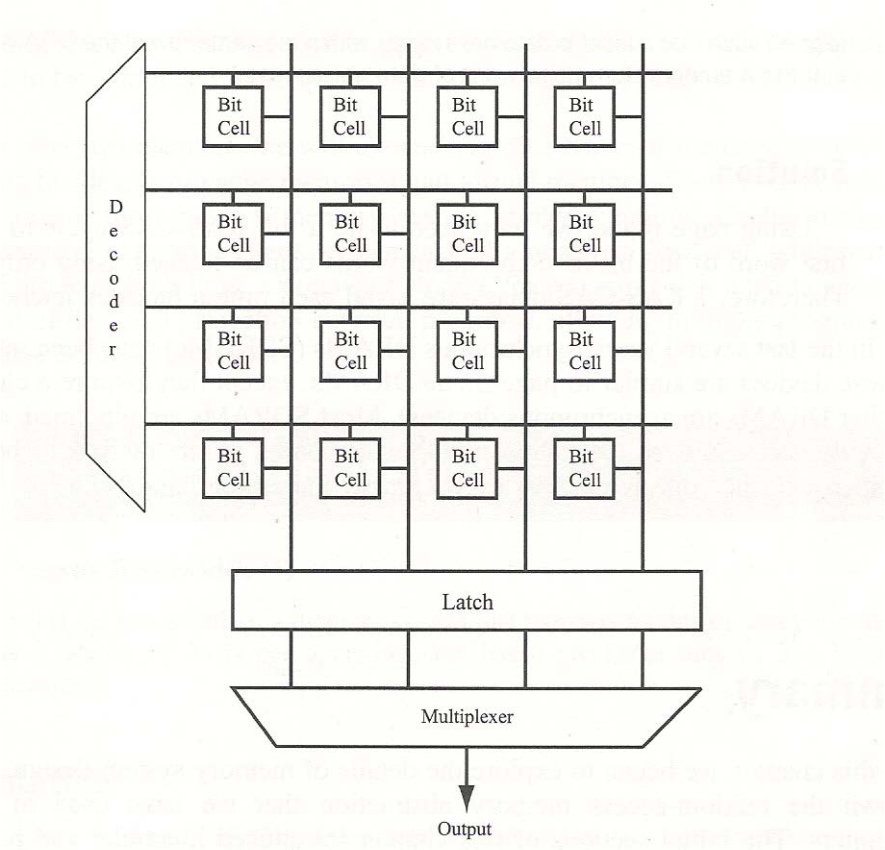
Μια αδυναμία του σχεδίου τσιπ μνήμης που παρουσιάζεται στο σχήμα 6-3 είναι ότι το περιεχόμενο μιας ολοκληρης σειράς που στέλνεται στον πολυπλεκτή κατά τη διάρκεια κάθε λειτουργίας, αλλά μόνο 1 bit από τη σειρά στέλνεται πραγματικά στην είσοδο.Εάν το περιεχόμενο της σειράς θα μπορούσε να κρατηθεί μέσα ή κοντά στον πολυπλεκτή, θα ήταν δυνατό να διαβαστούν άλλα bits μέσα στην ίδια σειρά ακριβώς για να στείλει μια διαφορετική διεύθυνση στηλών στη DRAM, παρά να κάνει έναν πλήρη κύκλο RAS-CAS.Οι DRAM που κάνουν αυτό καλούνται *τροποποίηση σελίδων* DRAM, και χρησιμοποιούν μια οργάνωση όπως αυτή που παρουσιάζεται στο σχήμα 6-8

.

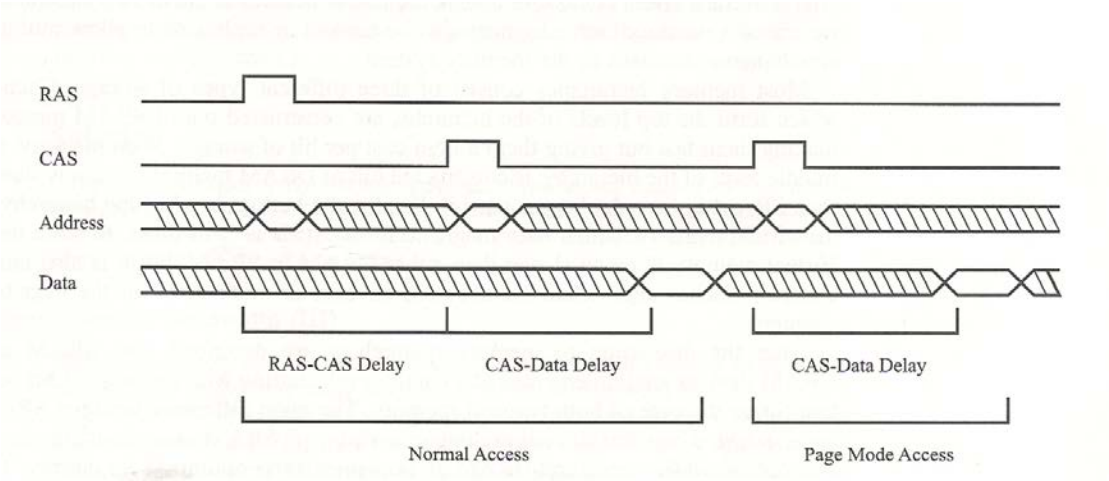
Η τροποίηση σελίδων DRAMs προσθέτει έναν σύρτη μεταξύ των αποτελεσμάτων του κελίου bits και του πολυπλεκτή.Όποτε μια διεύθυνση σειρών στέλνεται στη DRAM, ολόκληρο το περιεχόμενο της σειράς αποθηκεύεται στο σύρτη.Αυτό επιτρέπει τις επόμενες προσβάσεις που παραπέμπουν μια στήλη μέσα στην ίδια σειρά για να στείλουν απλά μια δεύτερη διεύθυνση στηλών στη DRAM, όπως φαίνεται στο σχήμα 6-9, μειώνοντας πολύ το χρόνο που απαιτείται για να προσκομίσει έναν παρακείμενο φραγμό των στοιχείων από τη DRAM.

.

Τα τελευταία χρόνια,οι σύγχρονες DRAMs (SDRAMs) έχουν εισαχθεί.Αυτές οι συσκευές είναι παρόμοιες με την τροποποίηση σελίδων DRAM, εκτός αν απαιτούν ένα χρονιστή εισόδου (άλλες DRAM είναι ασύγχρονες συσκευές).Οι περισσότερες SDRAMs διοχετεύονται, και πολλές παρέχουν τους τρόπους πρόσβασης που επιτρέπουν στις πολλαπλάσιες διαδοχικές λέξεις των στοιχείων για να διαβαστούν ή να γραφτούν με μόνο έναν RAS-CAS κύκλο, περαιτέρω αυξανόμενο εύρος ζώνης.



ΣΧΗΜΑ6-8 page mode DRAM



ΣΧΗΜΑ6-9 page mode access

ΚΕΦΑΛΑΙΟ 7^{ο}

CACHES-ΚΡΥΦΕΣ ΜΝΗΜΕΣ

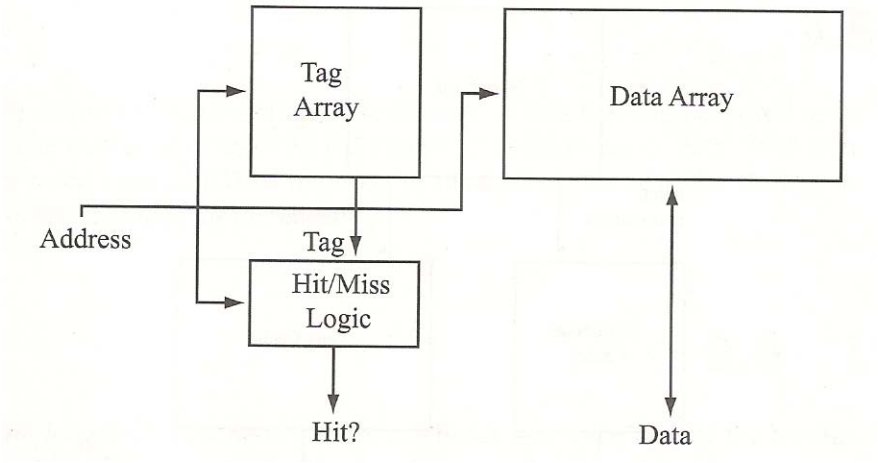
7.1 Στόχοι

Αυτό το κεφάλαιο συνεχίζει τη συζήτηση για τα συστήματα μνήμης, περιγράφοντας τις κρυφές μνήμες, που είναι μικρές και γρήγορες και τοποθετούνται κοντά στον επεξεργαστή.

7.2 Εισαγωγή

Οι κρυφές μνήμες τοποθετούνται στα κορυφαία επίπεδα της ιεραρχίας της μνήμης και σχεδόν πάντα κατασκευάζονται από SRAM. Η κύρια δομική διαφορά μεταξύ μιας κρυφής μνήμης και των άλλων επιπέδων της ιεραρχίας της μνήμης είναι ότι οι κρυφές περιέχουν και hardware για τον εντοπισμό των διευθύνσεων μνήμης που περιέχουν και για τη μετακίνηση πληροφοριών μέσα και έξω από αυτές όταν χρειάζεται. Τα κατώτερα επίπεδα της ιεραρχίας γενικά, βασίζονται στο λογισμικό ή σε συνδυασμό υλικού και λογισμικού για τη πραγματοποίηση των πιο πάνω λειτουργιών.

Οι κρυφές μνήμες περιέχουν ένα πίνακα ετικετών(tag array) και ένα πίνακα δεδομένων (data array), όπως φαίνεται στο σχήμα 7-1. Ο πίνακας ετικετών περιέχει διευθύνσεις των δεδομένων που περιέχονται στη μνήμη, ενώ ο πίνακας δεδομένων περιέχει τα δεδομένα. Αυτός ο διαχωρισμός των πινάκων μειώνει το χρόνο πρόσβασης της μνήμης, γιατί ο πίνακας ετικετών περιέχει τυπικά λιγότερα bits από τον πίνακα πληροφοριών και έτσι προσπελάζεται πιο γρήγορα από τον πίνακα δεδομένων ή ακόμα από ένα συνδυασμένο πίνακα ετικετών και δεδομένων. Από τη στιγμή που ο πίνακας ετικετών έχει προσπελαστεί οι έξοδοί του πρέπει να συγκριθούν με την αναφορά διεύθυνσης μνήμης για να καθοριστεί αν ταιριάζουν. Ο διαχωρισμός της κρυφής μνήμης σε πίνακες ετικετών και δεδομένων επιτρέπει σε αυτό τον καθορισμό να γίνεται παράλληλα με τη διαδικασία ψαξίματος στοιχείων του πίνακα δεδομένων, μειώνοντας έτσι το συνολικό χρόνο προσπέλασης.

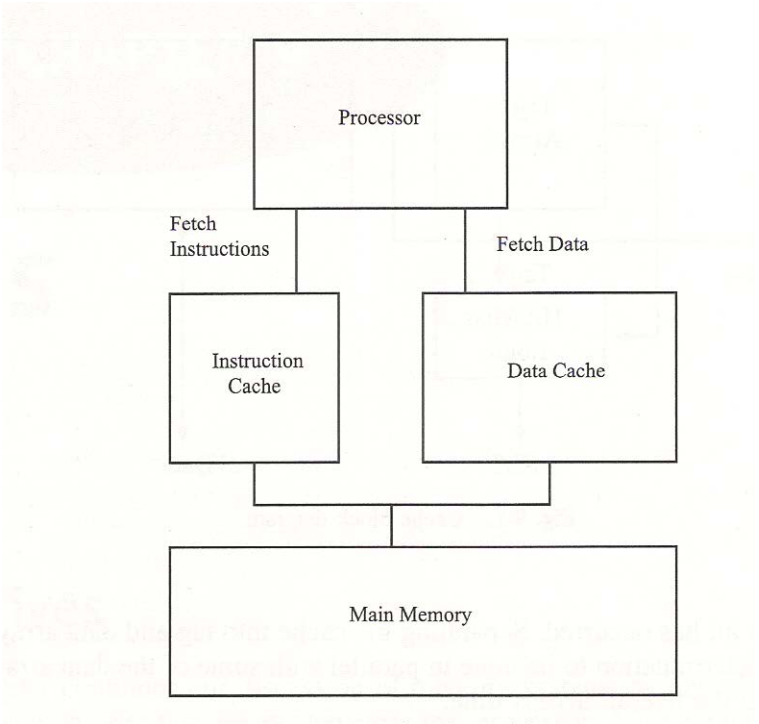


ΣΧΗΜΑ7-1 cache block diagram

7.3 Κρυφές μνήμες, μνήμες εντολών και ενοποιημένες μνήμες.

Στις συζητήσεις μας για τα συστήματα μνήμης, έχουμε υποθέσει ότι οι εντολές και τα δεδομένα μοιράζονται χώρο μέσα σε κάθε επίπεδο της ιεραρχίας της μνήμης. Για την κύρια και την εικονική μνήμη αυτό ισχύει. Όμως για τις κρυφές μνήμες τα δεδομένα και οι εντολές συχνά αποθηκεύονται σε διαφορετικές μνήμες όπως φαίνεται στο σχήμα 7-2. Αυτός ο διακανονισμός που μερικές φορές λέγεται Harvard cache ή Harvard architecture, και χρησιμοποιείται γιατί επιτρέπει στον επεξεργαστή να φέρνει εντολές από τη μνήμη εντολών και πληροφορίες από την μνήμη δεδομένων ταυτοχρόνως. Όταν μια μνήμη περιέχει και εντολές και δεδομένα, ονομάζεται ενοποιημένη κρυφή μνήμη (unified cache).

Ένα άλλο πλεονέκτημα του διαχωρισμού αυτού της μνήμης σε εντολές και δεδομένα, είναι ότι τα προγράμματα δεν τροποποιούν τις δικές τους εντολές. Γι αυτό το λόγο, οι μνήμες εντολών μπορούν να σχεδιαστούν ως συσκευές ανάγνωσης μόνο που δεν επιτρέπουν την τροποποίηση των εντολών που περιέχουν. Αυτό σημαίνει ότι η μνήμη εντολών μπορεί να αποβάλλει μπλόκ αντί να τα γράφει πάλι στην κύρια μνήμη αφού τα δεδομένα που περιέχουν δεν θα αλλάζουν από τότε που μπήκαν στην κρυφή μνήμη. Εν τέλει, με το να έχουμε ξεχωριστή μνήμη για τις εντολές και ξεχωριστή για τα δεδομένα προλαβαίνουμε συγκρούσεις ανάμεσα σε μπλοκ εντολών και μπλοκ μνήμης που μπορεί να είχαν χαρτογραφηθεί στον ίδιο αποθηκευτικό χώρο σε μια ενοποιημένη μνήμη.



ΣΧΗΜΑ7-2 Harvard cache architecture

Ένα μειονέκτημα της χρήσης διαφορετικής μνήμης για εντολές και δεδομένα είναι ότι δυσκολεύει το γράψιμο αυτοτροποποιήσιμων προγραμμάτων. Όταν ένα πρόγραμμα τροποποιεί τις εντολές του, οι εντολές αυτές αντιμετωπίζονται σαν δεδομένα και αποθηκεύονται στη μνήμη δεδομένων και όχι στη μνήμη εντολών. Για να εκτελεστούν οι τροποποιημένες εντολές το πρόγραμμα πρέπει να χρησιμοποιήσει μια ειδική κρυφή μνήμη για να διασφαλίσει ότι οι αρχικές εντολές δεν υπάρχουν ποια μέσα στην μνήμη εντολών και να εξαναγκάσει το πρόγραμμα να φέρει τις τροποποιημένες εντολές από την κύρια μνήμη για να εκτελεστούν. Αν η μνήμη δεδομένων είναι επανεγγράψιμη χρειάζονται επιπλέον λειτουργίες που θα εξασφαλίσουν ότι οι τροποποιημένες εντολές έχουν γραφτεί στην κύρια μνήμη πριν διαβαστούν από την κρυφή.Το overhead(πλεονάζων χώρος) που επιβάλλεται από αυτές τις λειτουργίες μειώνει την απόδοση του αυτοτροποιποιήσιμου κώδικα, και τον κάνει λιγότερο χρήσιμο.

Συχνά, οι μνήμη εντολών ενός συστήματος είναι σημαντικά μικρότερη (δύο με τέσσερις φορές) από την μνήμη δεδομένων. Αυτό συμβαίνει γιατί γενικότερα οι εντολές ενός προγράμματος καταλαμβάνουν λιγότερη μνήμη από τα δεδομένα του. Ακόμα, τα περισσότερα προγράμματα αφιερώνουν περισσότερο χρόνο σε βρόγχους που ξανα χρησιμοποιούν τις ίδιες εντολές πολλές φορές. Σαν αποτέλεσμα, η μνήμη εντολών του συστήματος μπορεί να είναι αρκετά μικρότερη από τη μνήμη δεδομένων και να έχει τον ίδιο λόγο επιτυχίας (hit rate),γιαυτό και οι σχεδιαστές συχνά αφιερώνουν περισσότερο χώρο και chip στη μνήμη δεδομένων

7.4 Περιγραφή κρυφών μνήμων

Για να συγκρίνουν τις μνήμες οι σχεδιαστές συζητούν τη χωρητικότητά τους, το μήκος γραμμής, το συσχετισμό (τον αριθμό των πιθανών χώρων όπου μπορεί να ανήκει η διεύθυνση) την πολιτική αντικατάστασης και αν η μνήμη είναι επανεγγράψιμη (write back) ή δι-εγγράψιμη (write through).

7.5 Χωρητικότητα

Η χωρητικότητα μιας κρυφής μνήμης είναι η ποσότητα των δεδομένων που μπορούν να αποθηκευτούν σε αυτή, έτσι μια κρυφή μνήμη 32KB μπορεί να αποθηκεύσει 32kilobyte δεδομένων. Μια τέτοια κρυφή μνήμη απαιτεί περισσότερη από 32KB μνήμη για να υλοποιηθεί, γιατί η αποθήκευση στους πίνακες ετικετών (tag arrays) δε συμπεριλαμβάνεται στην χωρητικότητα.

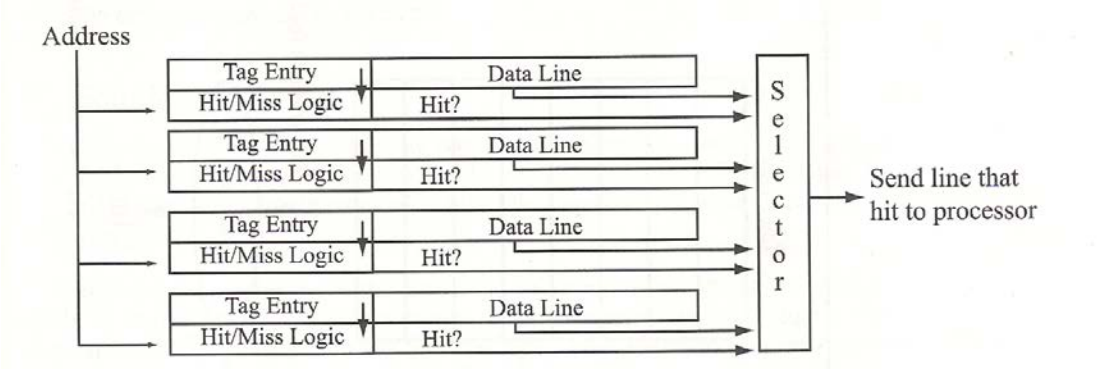
7.6 Μήκος γραμμής

Το μήκος γραμμής (line length) μια κρυφής μνήμης είναι το μέγεθος των μπλοκ της-δηλ το μέγεθος των μπλοκ δεδομένων που περιλαμβάνονται και απορρίπτονται από μια κρυφή μνήμη σε περίπτωση αποτυχίας. Για παράδειγμα, όταν μια κρυφή μνήμη 32byte έχει αποτυχία (cach miss)για να φέρνει ένα μπλοκ 32byte που περιέχει τη διεύθυνση της αποτυχίας πρέπει να εκβάλλει εκ των προτέρων ένα άλλο μπλοκ 32byte για να κάνει χώρο για τα νέα δεδομένα.. Οι γραμμές κρυφής μνήμης είναι ευθυγραμμισμένες (aligned), δηλαδή η διεύθυνση του πρώτου byte σε μια γραμμή είναι πολλαπλάσιο του μήκους της. Αυτό απλοποιεί τη διαδικασία καθορισμού

επιτυχίας ή αποτυχίας της κρυφής μνήμης γιατί τα τελευταία bits μιας διεύθυνσης προσδιορίζουν σε ποιο byte αναφέρεται μια διεύθυνση μέσα στη γραμμή που το περιέχει, και έτσι μόνο τα πρώτα bits της διεύθυνσης χρειάζεται να σταλούν στον πίνακα ετικετών και να καθοριστεί να έχουμε επιτυχία ή αποτυχία. Ο ακριβής αριθμός των bits που χρειάζεται να συγκριθούν για να εντοπιστεί η επιτυχία καθορίζεται από το μέγεθος της κρυφής μνήμης, το μήκος των γραμμών της και το συσχετισμό της. Αυτό θα αναλυθεί αργότερα στο κεφάλαιο.

7.7.1. Πλήρως συσχετισμένες κρυφές μνήμες

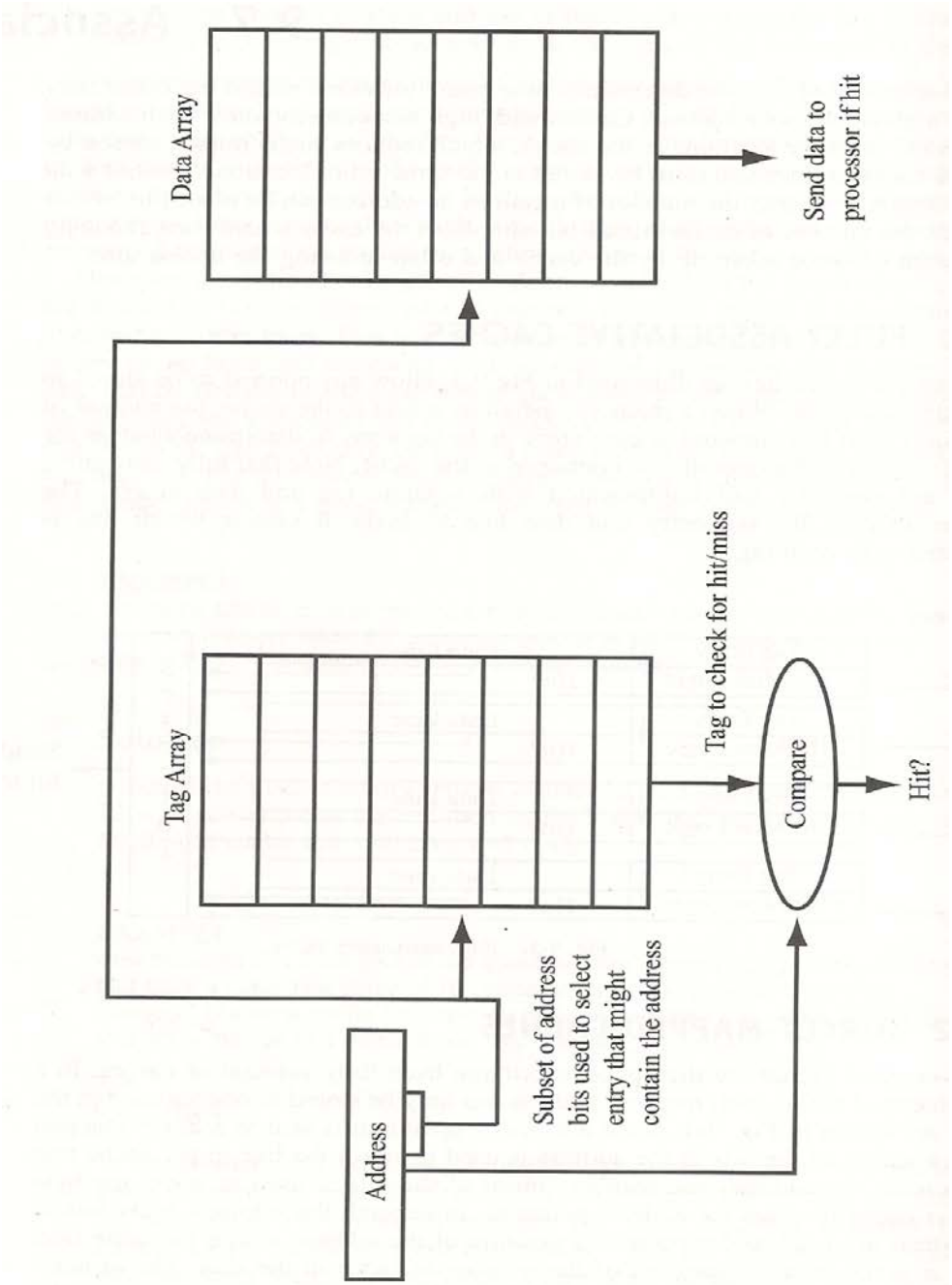
Οι πλήρως συσχετισμένες κρυφές μνήμες (fully associative caches), όπως φαίνεται και στο σχήμα 7-3, επιτρέπουν σε οποιαδήποτε διεύθυνση να αποθηκευτεί σε οποιαδήποτε γραμμή της κρυφής μνήμης. Όταν μια λειτουργία μνήμης στέλνεται στην κρυφή μνήμη, η διεύθυνση της αίτησης πρέπει να συγκριθεί μα κάθε καταχώρηση στον πίνακα ετικετών για να καθοριστεί εάν τα δεδομένα στα οποία αναφέρεται η αίτηση περιέχονται στην κρυφή μνήμη. Προσέξτε ότι οι πλήρως συσχετισμένες κρυφές μνήμες ακόμα υλοποιούνται με ξεχωριστούς πίνακες ετικετών και δεδομένων. Το διάγραμμα αποδίδει την καταχώρηση ετικέτας και τη γραμμή δεδομένων για να γίνει εμφανές ποια γραμμή συσχετίζεται με ποια ετικέτα.



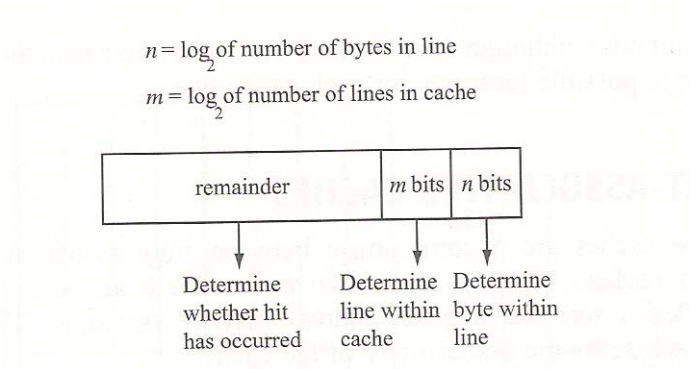
ΣΧΗΜΑ7-3 fully-associative cache

7.7.2. Κρυφές μνήμες άμεσης απεικόνισης

Οι κρυφές μνήμες άμεσης απεικόνισης (direct-mapped caches) είναι το άκρως αντίθετο από τις πλήρως συσχετισμένες κρυφές μνήμες. Σε μια μνήμη άμεσης απεικόνισης κάθε διεύθυνση μνήμης μπορεί να αποθηκευτεί σε μία θέση μέσα στη μνήμη. Όπως φαίνεται στο σχήμα 7-4, όταν μια λειτουργία μνήμης στέλνεται σε μια κρυφή μνήμη άμεσης απεικόνισης, ένα υποσετ των bits της διεύθυνσης χρησιμοποιείται για την επιλογή του byte μέσα στη γραμμή μνήμης, στο οποίο οδηγεί η διεύθυνση. Γενικότερα τα n τελευταία bit στη διεύθυνση χρησιμοποιούνται για να καθοριστεί η θέση της διεύθυνσης μέσα στη γραμμή μνήμης, όπου n είναι ο με βάση 2 λογάριθμος του αριθμού των bytes στη γραμμή. Τα επόμενα m πρώτα bits, όπου m είναι ο με βάση 2 λογάριθμος του αριθμού των γραμμών στην κρυφή μνήμη, χρησιμοποιούνται για την επιλογή της γραμμής μέσα στη οποία μπορεί να αποθηκευτεί η διεύθυνση, όπως φαίνεται στο σχήμα 7-5.



ΣΧΗΜΑ7-4direct-mapped cache



ΣΧΗΜΑ7-5 address breakdown

Οι μνήμες άμεσης απεικόνισης έχουν το πλεονέκτημα ότι απαιτούν ένα σημαντικά μικρότερο χώρο chip για να υλοποιηθούν, σε σχέση με τις πλήρως συσχετισμένες κρυφές μνήμες, γιατί οι πρώτες απαιτούν μια συσκευή σύγκρισης για να καθορίσουν αν έχουν επιτυχία ή αποτυχία, ενώ οι τελευταίες απαιτούν μία συσκευή σύγκρισης για κάθε γραμμή της κρυφής μνήμης. Επιπρόσθετα, οι μνήμες άμεσης απεικόνισης συνήθως έχουν χαμηλότερο χρόνο προσπέλασης, γιατί υπάρχει μόνο μία σύγκριση για να καθοριστεί η επιτυχία ή αποτυχία ενώ οι πλήρως συσχετισμένες πρέπει να εξετάσουν κάθε σύγκριση και να επιλέξουν την κατάλληλη λέξη δεδομένων για να στείλουν τον επεξεργαστή.

Παρόλα αυτά, οι μνήμες άμεσης απεικόνισης τείνουν να έχουν μικρότερα ποσοστά επιτυχίας από τις πλήρως συσχετισμένες, λόγω συγκρούσεων μεταξύ των γραμμών που χαρτογραφούνται στον ίδιο χώρο μέσα στην κρυφή μνήμη. Κάθε διεύθυνση μπορεί να τοποθετηθεί σε μία θέση στην κρυφή μνήμη, η οποία καθορίζεται από τα m bits διεύθυνσης, όπως φαίνεται στο σχήμα 7-5. Αν δύο διευθύνσεις έχουν την ίδια τιμή σε αυτά τα bits, χαρτογραφούνται στην ίδια γραμμή της μνήμης και δε μπορούν να διανεμηθούν στη μνήμη την ίδια στιγμή. Ένα πρόγραμμα που εναλλάσσεται σε αναφορές σε αυτές τις δύο διευθύνσεις δε θα έχει ποτέ επιτυχία στη κρυφή μνήμη, αφού η γραμμή που περιέχει τη διεύθυνση θα απορριπτόταν πριν την επόμενη αναφορά στη μνήμη. Έτσι, η μνήμη θα πετύχαινε 0% ποσοστό επιτυχίας ακόμα και αν το πρόγραμμα μόνο αναφερόταν στις δύο διευθύνσεις. Πρακτικά, οι μνήμες άμεσης απεικόνισης, και ειδικά οι μεγάλες μνήμες άμεσης απεικόνισης, μπορεί να επιτύχει καλά ποσοστά επιτυχίας, παρόλο που τείνουν να είναι μικρότερα από εκείνα των μνημών που παρέχουν πολλαπλές πιθανές θέσεις για κάθε γραμμή μνήμης .

7.8 Πολιτική αντικατάστασης.

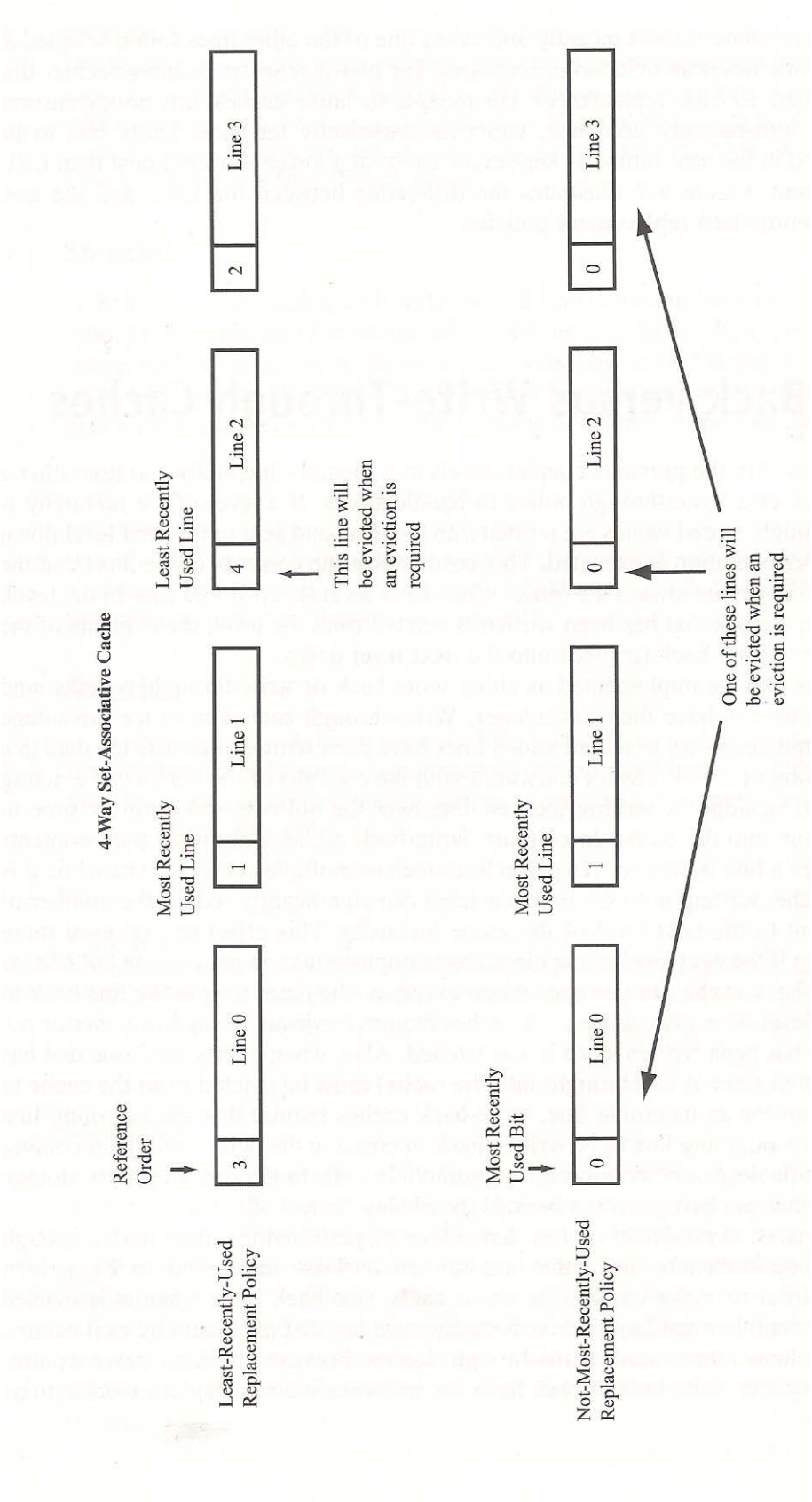
Όταν μια γραμμή πρέπει να απομακρυνθεί από την κρυφή μνήμη για να γίνει χώρος για νέα δεδομένα, είτε γιατί η μνήμη είναι πλήρης, είτε εξαιτίας συγκρούσεων σε κάποιο σετ, η πολιτική αντικατάστασης καθορίζει ποια γραμμή θα απομακρυνθεί. Σε μνήμες άμεσης απεικόνισης δεν υπάρχει τέτοια επιλογή αφού η νέα γραμμή μπορεί να τοποθετηθεί μόνο σε μία θέση μνήμης, αλλά οι μνήμες συσχετισμένων σετ και οι πλήρως συσχετισμένες μνήμες περιέχουν πλήθος γραμμών που μπορούν να αντικατασταθούν. Σε αυτές τις μνήμες , ο γενικός στόχος της πολιτικής

αντικατάστασης είναι να ελαχιστοποιηθούν οι μελλοντικές αποτυχίες, απομακρύνοντας γραμμές που δε θα ζητηθούν συχνά στο μέλλον. Οι σχεδιαστές των πολιτικών αντικατάστασης πρέπει να λάβουν υπ'οψιν τους και το κόστος της πολιτικής τους. Αν μια πολιτική μειώνει τις μελλοντικές αποτυχίες λίγο άλλα χρειάζεται πολύ υλικό έτσι ώστε η χωρητικότητα της κρυφής μνήμης να πρέπει να μειωθεί για να προσαρμοστεί στην πολιτική αντικατάστασης, τότε οι επιπλέον αποτυχίες που θα προκληθούν από τη μειωμένη χωρητικότητα μπορεί να υπερβούν τα οφέλη της βελτιωμένης πολιτικής αντικατάστασης.

Η τέλεια πολιτική αντικατάστασης θα εξέταζε τη μελλοντική συμπεριφορά του προγράμματος που εκτελείται και θα απομάκρυνε τη γραμμή που οδηγεί σε λιγότερες αποτυχίες μνήμης. Αφού όμως οι υπολογιστές δεν ξέρουν πώς θα συμπεριφερθούν τα προγράμματα στο μέλλον, η πολιτική αντικατάσταση θα επιλέξει βασιζόμενη στο τι έκανε το πρόγραμμα στο παρελθόν.

Μια συνηθισμένη πολιτική είναι ή λιγότερο πρόσφατα χρησιμοποιημένη γραμμή.(least recently used- LRU) .Στην LRU αντικατάσταση η κρυφή μνήμη κατατάσσει τις γραμμές σε σετ ανάλογα με το πόσο πρόσφατα προσπελάστηκαν και απορρίπτει την λιγότερο χρησιμοποιημένη από το σετ όταν χρειάζεται.. Αυτή η λογική βασίζεται στην παρατήρηση ότι οι γραμμές που δεν προσπελάστηκαν πρόσφατα είναι απίθανο να ζητηθούν στο κοντινό μέλλον. Μία άλλη πολιτική είναι η τυχαία αντικατάσταση (random replacement) στην οποία επιλέγεται τυχαία μια γραμμή από το κατάλληλο σετ για να γίνει χώρος για τα εισερχόμενα δεδομένα.

Οι μελέτες έχουν αποδείξει ότι η LRU αντικατάσταση δίνει ελαφρώς καλύτερα ποσοστά από την τυχαία αντικατάσταση, αλλά οι διαφορές αυτές είναι πολύ μικρές σε μνήμες κάποιου λογικού μεγέθους. Η LRU αντικατάσταση είναι σχετικά περίπλοκη να υλοποιηθεί. Κάθε φορά που μια γραμμή σε ένα σετ αναφέρεται πρέπει να ενημερώνεται, και έτσι οδηγούμαστε σε σχετικά περίπλοκο hardware. Γιαυτό το λόγο ορισμένες κρυφές μνήμες χρησιμοποιούν τη μη-πιο-προσφατα-χρησιμοποιημένη (not-most-recently-used) πολιτική αντικατάστασης. Σε αυτή την πολιτική η κρυφή μνήμη συγκρατά τη γραμμή σε κάθε σετ που αναφέρθηκε πιο πρόσφατα και απορρίπτει μια από τις άλλες γραμμές(συχνά επιλέγοντάς την τυχαία) όποτε χρειαστεί. Για κρυφές μνήμες συσχετισμένων σετ διπλής κατεύθυνσης αυτή η πολιτική ισοδυναμεί με την LRU. Για κρυφές μνήμες συσχετισμένων σετ τετραπλής κατεύθυνσης η πολιτική αυτή διασφαλίζει ότι η πιο πρόσφατα χρησιμοποιημένη γραμμή, η οποία στατιστικά είναι πιο πιθανό να αναφερθεί και στο κοντινό μέλλον, μένει στη μνήμη. Η πολιτική αυτή έχει μικρότερο κόστος υλικού από την LRU. Το σχήμα 7-6 επεξηγεί τη διαφορά μεταξύ της LRU και της μη-πιο-προσφατα-χρησιμοποιημένης πολιτικής αντικατάστασης.



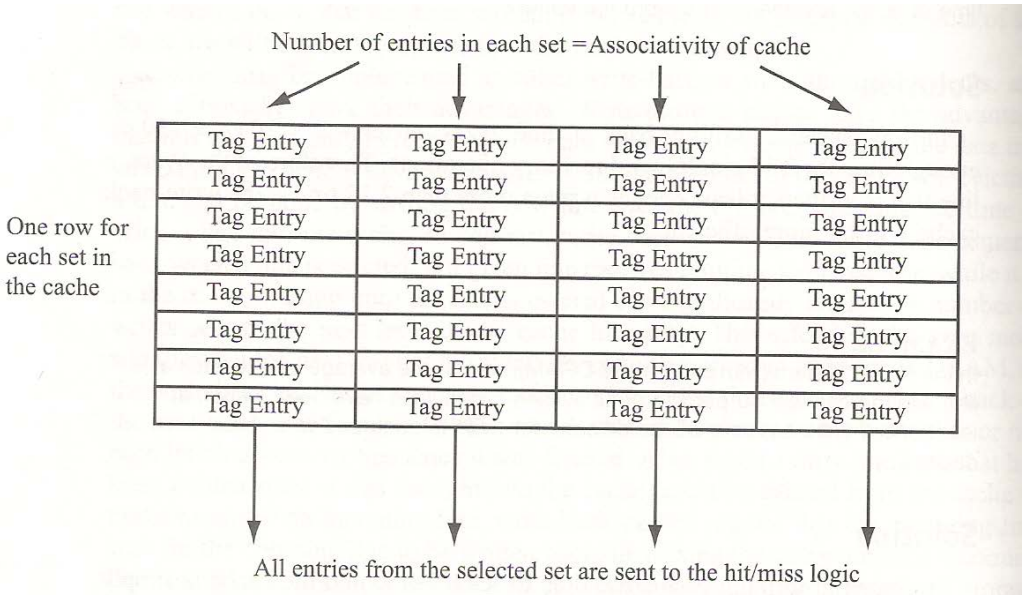
ΣΧΗΜΑ7-6LRU vs. Not-most-resently-used replacement

7.9 Υλοποίηση κρυφής μνήμης

Ως τώρα είδαμε πως είναι δομημένες οι κρυφές μνήμες, χωρίς να προσέχουμε τις λεπτομέρειες υλοποίησής τους. Το Σχήμα 7-1 δείχνει ένα βασικό διάγραμμα του τρόπου υλοποίησης των περισσότερων κρυφών μνημών με τη χρήση πινάκων ετικετών και δεδομένων και Λογική εντοπισμού επιτυχίας ή αποτυχίας. Στα επόμενα τμήματα αυτού του κεφαλαίου θα ασχοληθούμε με την υλοποίηση αυτών των τριών συστατικών.

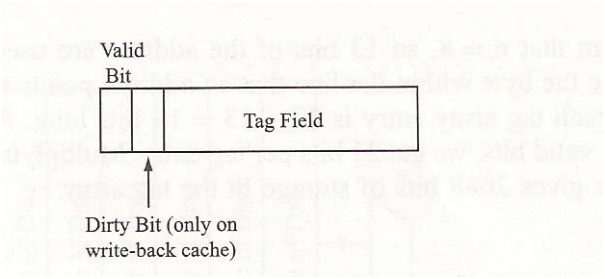
7.10 Πίνακες ετικετών

Γενικότερα ένας πίνακας ετικετών (tag array) οργανώνεται σε πίνακα δύο διαστάσεων που περιέχει μια στήλη καταχωρήσεων ετικετών για κάθε σετ στην κρυφή μνήμη ,με τον αριθμό των καταχωρήσεων κάθε στήλης να είναι ίσος με το συσχετισμό της κρυφής μνήμης. Το σχήμα 7-7 δείχνει τη δομή του πίνακα ετικετών για μία κρυφή μνήμη συσχετισμού σετ τεσσάρων κατευθύνσεων.



ΣΧΗΜΑ7-7 tag array for 4-way set-associative cache

Μια καταχώρηση ετικέτας περιέχει τις πληροφορίες που χρειάζονται να καταγράψει ποια γραμμή δεδομένων είναι αποθηκευμένη στη γραμμή της μνήμης δεδομένων που σχετίζεται με την καταχώρηση. Κάθε καταχώρηση περιγράφει μια γραμμή μνήμης δεδομένων. Όπως φαίνεται στο σχήμα 7-8 η καταχώρηση ετικέτας αποτελείται από ένα πεδίο ετικέτας που περιέχει το τμήμα της διεύθυνσης της γραμμής που δε χρησιμοποιείται για την επιλογή του σετ (το «υπόλοιπο» πεδίο του σχήματος 7-5), ένα bit επικύρωσης (valid bit) εάν η γραμμή που σχετίζεται με τη συγκεκριμένη καταχώρηση περιέχει κάποια έγκυρα δεδομένα, και ένα βρώμικο bit (dirty bit) που αφορά τις επενεγγράψιμες κρυφές μνήμες.



ΣΧΗΜΑ7-8 tag array

Ανάλογα με την πολιτική αντικατάστασης της κρυφής μνήμης η καταχώρηση ετικέτας μπορεί επίσης να περιέχει ένα ή περισσότερα επιπρόσθετα bits. Για παράδειγμα σε μια κρυφή μνήμη με LRU αντικατάσταση, κάθε καταχώρηση ετικέτας πρέπει να μπορεί να καταγράψει πόσες από τις άλλες γραμμές του σετ έχουν αναφερθεί από την τελευταία στιγμή που αναφέρθηκε η γραμμή στην οποία αντιστοιχεί η καταχώρηση. Με αυτό τον τρόπο η πολιτική αντικατάστασης μπορεί να εξακριβώσει ποιες ήταν οι λιγότερο πρόσφατα χρησιμοποιημένες γραμμές, όταν θα χρειαστεί να γίνει μια αντικατάσταση. Αυτό απαιτεί $\log_2$ (συσχετισμός κρυφής μνήμης) bits δεδομένων σε κάθε καταχώρηση ετικέτας.

Όταν ένας υπολογιστής πρωτοενεργοποιείται, όλα τα bits επικύρωσης στον πίνακα ετικετών θέτονται στο 0, για να καταγραφεί το γεγονός ότι δεν υπάρχουν δεδομένα στην κρυφή μνήμη. Όποτε μια γραμμή εισάγεται στην μνήμη το bit επικύρωσης στην αντίστοιχη εγγραφή του πίνακα ετικετών γίνεται 1, καταγράφοντας το γεγονός ότι η νέα γραμμή περιέχει έγκυρα δεδομένα. Γενικά, όταν γεμίσει μια γραμμή, παραμένει πλήρης γιατί τα δεδομένα παραμένουν στη μνήμη μέχρι να αντικατασταθούν από κάποια γραμμή δεδομένων. Εξαίρεση σε αυτό αποτελεί η σκόπιμη αφαίρεση μιας γραμμής από το πρόγραμμα (οι περισσότεροι επεξεργαστές παρέχουν εντολές για να το κάνουν αυτό).Σε αυτή την περίπτωση η γραμμή αδειάζει και το bit επικύρωσης γίνεται πάλι 0.

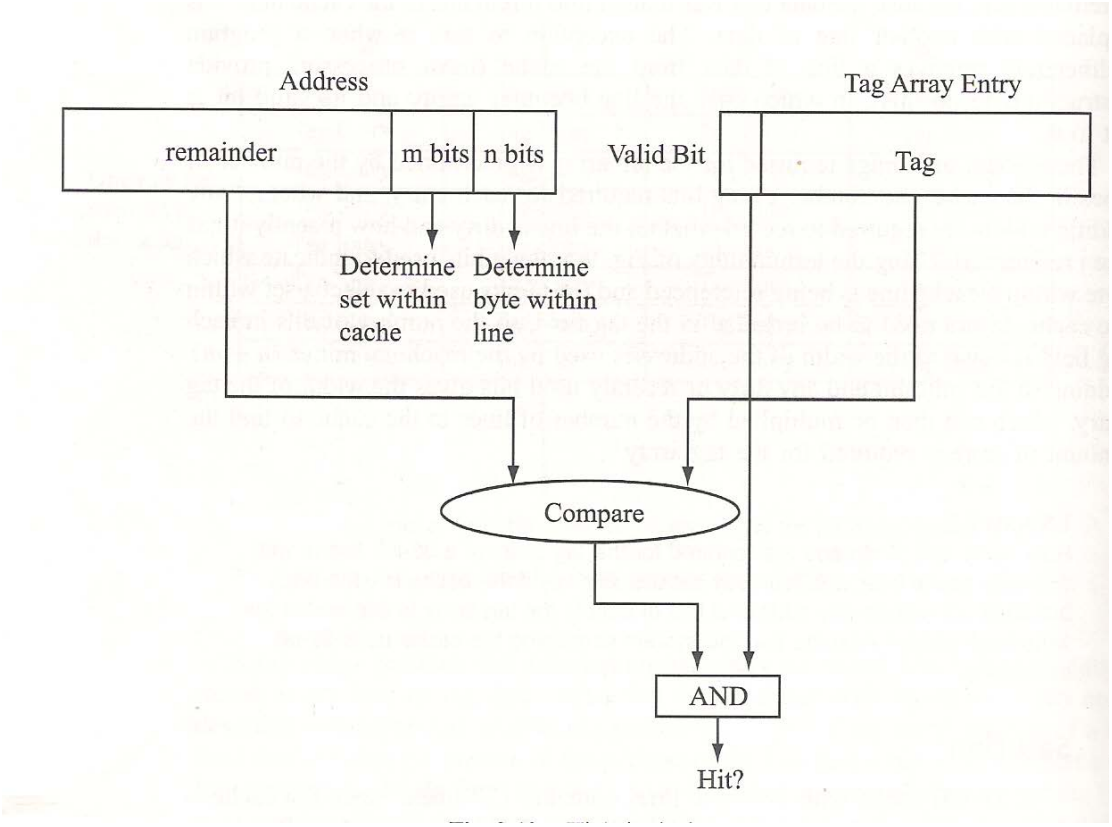
Η ποσότητα αποθήκευσης που απαιτείται για τον πίνακα ετικετών καθορίζεται από τον αριθμό των γραμμών στην κρυφή μνήμη, τον αριθμό των bits ετικετών που χρησιμοποιούνται για κάθε εγγραφή και από το αν είναι απαραίτητα επιπλέον bits ώστε να καταγραφεί αν η γραμμή ήταν βρώμικη και πόσο πρόσφατα αναφέρθηκε. Χρησιμοποιώντας την ορολογία του σχήματος 7-5, n bits χρησιμοποιούνται για να δείξουν το byte μέσα στη γραμμή της κρυφής μνήμης που αναφέρεται ενώ τα m bits που δείχνουν την επιλογή του σετ μέσα στη μνήμη δεν συμπεριλαμβάνονται στο πεδίο ετικέτας. Έτσι ο αριθμός των bits σε κάθε πεδίο ετικέτας είναι ίσος με το πλάτος της διεύθυνσης που χρησιμοποιεί η μηχανή μείον (n+m).Προσθέτοντας το bit επικύρωσης και οποιοδήποτε από τα βρώμικα ή πρόσφατα χρησιμοποιημένα bits παίρνουμε το πλάτος της καταχώρησης ετικέτας, η οποία αν πολλαπλασιαστεί με τον αριθμό των γραμμών στην κρυφή μνήμη μας δίνει το ποσοστό αποθήκευσης που απαιτείται για τον πίνακα ετικετών.

7.12 Λογική Επιτυχίας αποτυχίας (Hit/Miss Logic)

Η λογική επιτυχίας / αποτυχίας συγκρίνει τα υπόλοιπα bits της μνήμης που έχει αναφερθεί με τα περιεχόμενα του πεδίου ετικέτας, κάθε καταχώρησης του σετ. Αν αυτά τα πεδία bit ταιριάζουν και το bit επικύρωσης της καταχώρησης έχει τεθεί, τότε έχουμε επιτυχία, όπως φαίνεται στο σχήμα 7-9.

7.13 Πίνακες Δεδομένων (Data Arrays)

Η δομή του πίνακα δεδομένων μιας κρυφής μνήμης είναι παρόμοια με αυτή του πίνακα ετικετών. Ο πίνακας δεδομένων είναι ένας πίνακας γραμμών μνήμης δύο διαστάσεων , με μία στήλη για κάθε σετ που βρίσκεται στην κρυφή μνήμη, και αριθμό στηλών ίσο με το συσχετισμό της κρυφής μνήμης. Στο σχήμα 7-10 βλέπουμε πως μπορεί να σχεδιαστεί ένας πίνακας δεδομένων που ταιριάζει με τον πίνακα ετικετών του σχήματος 9-7. Όταν μια διεύθυνση στέλνεται στη κρυφή μνήμη ο πίνακας προβάλλει όλες τις γραμμές που ενδέχεται να περιέχουν τη διεύθυνση. Αν έχουμε επιτυχία, η γραμμή δεδομένων που αντιστοιχεί στην καταχώρηση ετικέτας επιλέγεται και τα bits της διεύθυνσης του «byte μέσα στη γραμμή» χρησιμοποιούνται για να επιλεγεί η λέξη δεδομένων που θα αποσταλεί στον επεξεργαστή. Κατά τη διάρκεια μιας λειτουργίας αποθήκευσης ,τα δεδομένα ρέουν προς την αντίθετη κατεύθυνση, και τα δεδομένα που εγγράφονται οδηγούνται στη σωστή θέση μέσα στο σετ.



ΣΧΗΜΑ7-9 hit/miss logic

ΣΧΗΜΑ7-10 data array organization

ΚΕΦΑΛΑΙΟ 8^{ο}

ΕΙΚΟΝΙΚΗ ΜΝΗΜΗ

8.1 Στόχοι

Η εικονική μνήμη είναι μια τεχνική που χρησιμοποιείται από τα μαγνητικά μέσα αποθήκευσης, όπως ο σκληρός δίσκος, για να χρησιμεύσει ως ένα επίπεδο στο σύστημα μνήμης και για να παρέχει προστασία μεταξύ των προγραμμάτων που τρέχουν στο ίδιο σύστημα, έτσι ώστε ένα πρόγραμμα να μην μπορεί να τροποποιήσει τα επόμενα στοιχεία

Αφου συμπληρώσετε αυτό το κεφάλαιο, πρέπει
1.Να καταλάβετε την εικονική μνήμη, τις εικονικές διευθύνσεις και τις φυσικές διευθύνσεις.
2.Να είστε σε θέση να λύσετε τα προβλήματα που περιλαμβάνουν την εικονική μνήμη και τη μετάφραση διευθύνσεων.
3.Να καταλάβετε τους απομονωτές lookaside μεταφράσεων (TLBs), και να είστε σε θέση να λύσετε τα προβλήματα σχετικά με αυτους.
4.Να κατανόησετε το λόγο και τον τρόπο με τον οποίο η εικονική μνήμη παρέχει την προστασία στα σύγχρονα συγκροτήματα ηλεκτρονικών υπολογιστών.

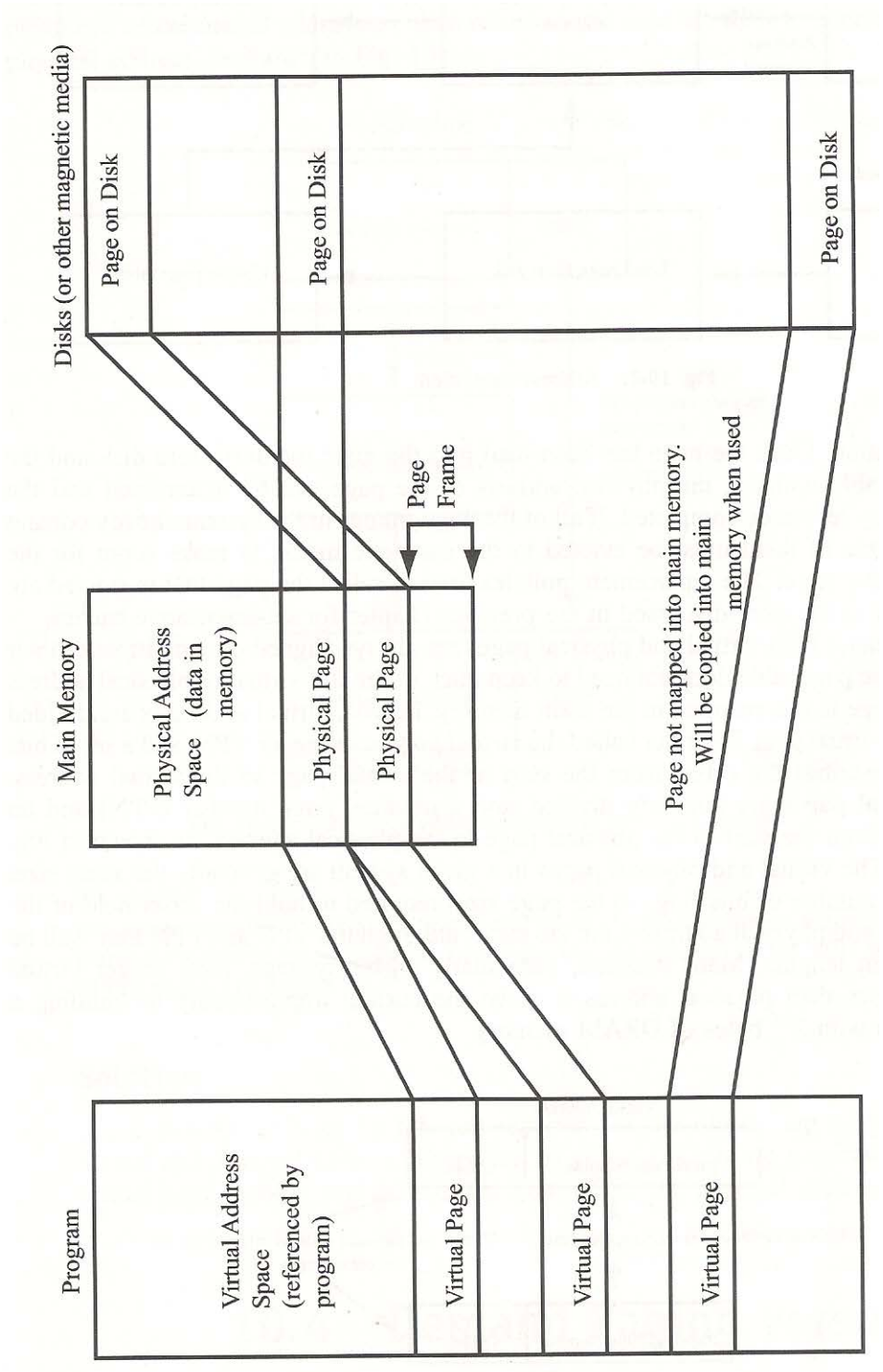
8.2 Εισαγωγή

Το κόστος της μνήμης ήταν ένας σημαντικός περιορισμός στα πρόωρα συγκροτήματα ηλεκτρονικών υπολογιστών.Πριν από την ανάπτυξη του ημιαγωγού DRAM, η πίο επικράτουσα τεχνολογία μνήμης ήταν μνήμη πυρήνων, στην οποία μία διαχτυλιοειδές-διαμορφωμένη κυκλική λίστα του μαγνητικού υλικού χρησιμοποιήθηκε για να αποθηκεύσει κάθε bit των στοιχείων.Η δαπάνη της παραγωγής αυτών των μαγνητικών κυκλικών λιστών και του κόστους που περιλαμβάνεται στη συγκέντρωση τους στις συσκευές μνήμης οδήγησε στις περιορισμένες ικανότητες μνήμης σε πολλές μηχανές.

Για να επιλυθεί αυτό το πρόβλημα,αναπτύχθηκε η εικονική μνήμη.Σε ένα σύστημα εικονικής μνήμης, οι σκληροί δίσκοι ή τα αλλα μαγνητικά μέσα διαμορφώνουν το κατώτατο στρώμα της ιεραρχίας μνήμης, με τη DRAM ή τη μνήμη πυρήνων που διαμορφώνουν το επίπεδο κύριας μνήμης της ιεραρχίας.Τα προγράμματα δεν μπορούν άμεσα να έχουν πρόσβαση στα στοιχεία που αποθηκεύονται στα μαγνητικά μέσα.Αντ' αυτού το διάστημα διευθύνσεων ενός προγράμματος διαιρείται σε σελίδες, παρακείμενοι φραγμοί των στοιχείων που αποθηκεύονται στα μαγνητικά μέσα. Στα σύγχρονα συστήματα, οι σελίδες είναι χαρακτηριστικά 2 έως 8 KB στο μέγεθος, αν και μερικά συστήματα παρέχουν την υποστήριξη για τις σελίδες των πολλαπλάσιων μεγεθών.Όταν μια σελίδα των στοιχείων είναι παραπεμφθείσα, το σύστημα την αντιγράφει στην κύρια μνήμη, που επιτρέπει σε αυτην για να προσεγγιστεί.Αυτό μπορεί να απαιτήσει ότι μια άλλη σελίδα των στοιχείων αντιγράφεται από την κύρια μνήμη στα μαγνητικά μέσα προκειμένου να γίνει ο χώρος για την εισερχόμενη σελίδα.

Το σχήμα 8-1 επεξηγεί πολλές από τις βασικές έννοιες στην εικονική μνήμη.Κάθε ένα προγράμμα έχει το εικονικό διάστημα διευθύνσεών του, το οποίο είναι το σύνολο

διευθύνσεων που τα προγράμματα χρησιμοποιούν για τις διαδικασίες φορτωσης και αποθήκευσης.Το φυσικό διάστημα διευθύνσεων είναι οι διευθύνσεις συνόλου που χρησιμοποιούνται στις θέσεις αναφοράς στην κύρια



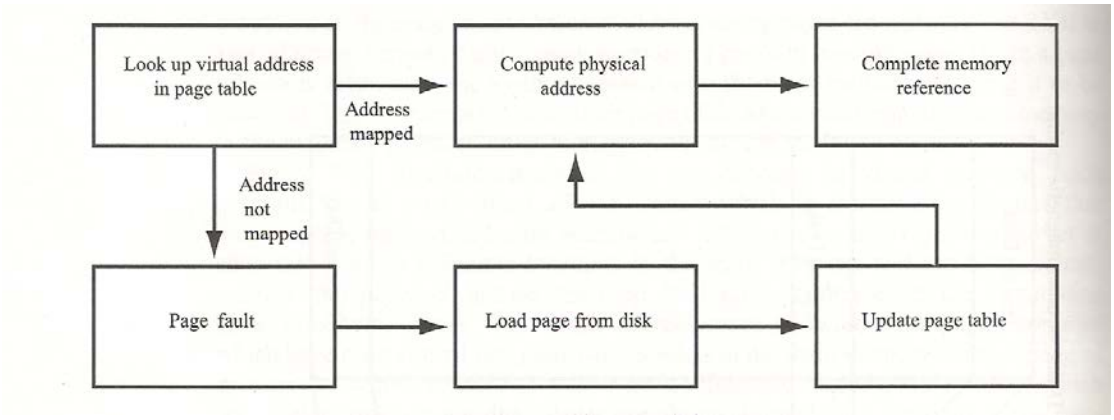
ΣΧΗΜΑ8-1 virtual memory

μνήμη, και η φυσική διεύθυνση χρησιμοποιείται για να περιγράψει τις διευθύνσεις στα εικονικά και φυσικά διαστήματα διευθύνσεων.Το εικονικό διάστημα διευθύνσεων διαιρείται σε σελίδες, μερικές από τις οποίες έχουν αντιγραφεί στα πλαίσια σελίδων (αυλακώσεις στην κύρια μνήμη όπου μια σελίδα των στοιχείων μπορεί να αποθηκευτεί) επειδή έχουν παραπεμφθεί πρόσφατα, και μερικές από αυτές κατοικούν μόνο στο δίσκο.Οι σελίδες ευθυγραμμίζονται πάντα σε ένα πολλαπλάσιο του μήκους σελίδων, έτσι δεν επικαλύπτουν ποτέ.Οι όροι της εικονικής σελίδας και της φυσικής σελίδας χρησιμοποιούνται για να περιγράψουν μια σελίδα στοιχείων στα εικονικά και φυσικά διαστήματα διευθύνσεων, αντίστοιχα.Οι σελίδες που έχουν φορτωθεί στην κύρια μνήμη από το δίσκο λέγεται ότι έχει χαρτογραφηθεί στην κύρια μνήμη .

Η εικονική μνήμη επιτρέπει σε έναν υπολογιστή να ενεργήσει σαν η κύρια μνήμη της να ήταν πολύ μεγαλύτερη από ότι είναι πραγματικά.Όταν ενα προγράμμα δηλώνει μια εικονική διεύθυνση, αυτό δεν μπορεί να πεί, εκτός από το συγχρονισμό της λανθάνουσας κατάστασης της λειτουργίας, εαν η εικονική διεύθυνση κατοικούσε στην κύρια μνήμη του υπολογιστή ή εάν έπρεπε να προσκομιστεί από τα μαγνητικά μέσα. Κατά συνέπεια, ο υπολογιστής μπορεί να μεταθέσει τις σελίδες και έξω απο τη κύρια μνήμη όπως απαιτείται, παρόμοια με τον τρόπο όπου οι γραμμές της κρυφής μνήμης παρουσιάζονται πρός και από την κρυφή μνήμη όπως απαιτείται, επιτρέποντας στα προγράμματα για να δηλώσουν περισσότερα στοιχεία από αυτά που μπορούν να αποθηκευτούν στην κύρια μνήμη σε οποιοδήποτε χρόνο.

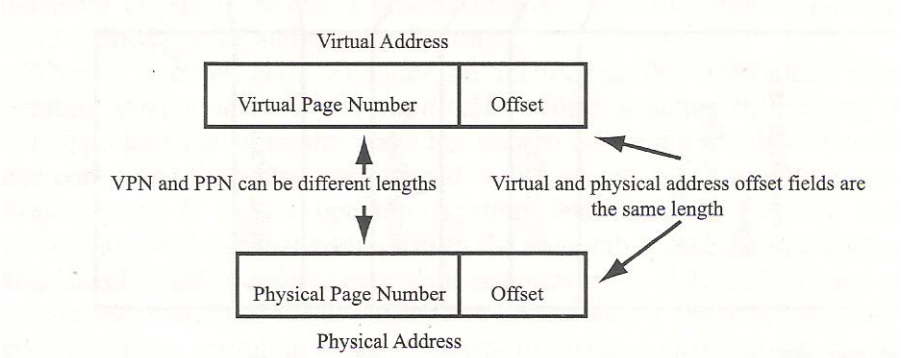
8.3 Μετάφραση διευθύνσεων

Τα προγράμματα που τρέχουν για ένα σύστημα με την εικονική μνήμη χρησιμοποιούν τις εικονικές διευθύνσεις ως επιχειρήματα για να φορτώσουν και να αποθηκεύσουν τις οδηγίες, αλλά η κύρια μνήμη χρησιμοποιεί τις φυσικές διευθύνσεις για να καταγράψει τις θέσεις όπου το στοιχείο αποθηκεύεται πραγματικά.Όποτε ένα πρόγραμμα εκτελεί μια αναφορά μνήμης, η εικονική διεύθυνση που χρησιμοποιεί αυτό πρέπει να μετατραπεί στη ισοδύναμη φυσική διεύθυνση, μια διαδικασία γνωστή ως μετάφραση διευθύνσεων .Το σχήμα 8-2 παρουσιάζει ένα διάγραμμα ροής της μετάφρασης διευθύνσεων .



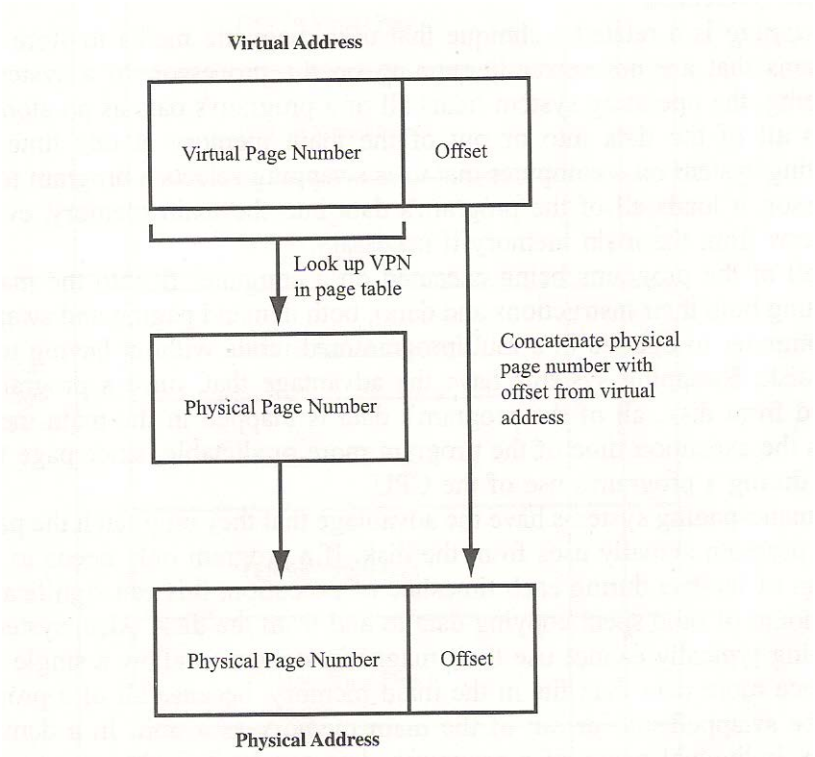
ΣΧΗΜΑ 8-2 address translation

Όταν το πρόγραμμα ενός χρήστη εκτελεί μια οδηγία όπου η μνήμη αναφερει ,το λειτουργικο σύστημα προσπελαύνει τον πινακα σελίδων, μια δομή δεδομένων σε μια μνήμη που κρατά τη απεικόνιση των εικονικών σε φυσικές διευθύνσεις, για να καθορίσει εάν εικονική σελίδα περιέχει ή όχι τη διεύθυνση που παραπέμπεται από τη λειτουργία είναι συχρόνως απεικονισμένη επάνω σε μια φυσική σελίδα.Σε αυτή την περίπτωση, το λειτουργικό σύστημα καθορίζει τη φυσική διεύθυνση που αντιστοιχεί στην εικονική διεύθυνση από τον πίνακα σελίδων, χρησιμοποιώντας τη φυσική διεύθυνση για να έχει πρόσβαση στην κεντρική μνήμη.Εάν η εικονική σελίδα που περιέχει την παραπεμφθείσα διεύθυνση δεν απεικονίζεται επάνω σε μια φυσική σελίδα, εμφανίζεται ένα ελάττωμα σελίδων, και το λειτουργικό σύστημα προσκομίζει τη σελίδα που περιέχει τα ζητούμενα στοιχεία από τη μνήμη,φορτώνει αυτα σε μια σελίδα πλαισιων και ενημερώνει τον πίνακα σελίδων για τη νέα μετάφραση.Μόλις διαβαστεί η σελίδα μέσα στην κύρια μνήμη από το δίσκο και τον ενημερωμένο πίνακα σελίδων,η φυσική διεύθυνση της σελίδας μπορεί να καθοριστεί και η αναφορά μνήμης ολοκληρωνεται.Εάν όλα τα πλαίσια σελίδων στο σύστημα περιέχουν ήδη την εισερχόμενη σελίδα.Οι πολιτικές αντικατάστασης που χρησιμοποιούνται για να επιλέξουν τη σελίδα που εκδιώκεται είναι παρόμοιες με αυτές που συζητούνται στο προηγούμενο κεφάλαιο για το σύνολο - συνειρμικές κρύφές μνήμες.



ΣΧΗΜΑ8-3 virtual and physical addresses

Επειδή και οι εικονικές και φυσικές σελίδες ευθυγραμμίζονται πάντα σε ένα πολλαπλάσιο του μεγέθους τους, ο πίνακας σελίδων δεν πρέπει να διατηρήσει τη πλήρη εικονική ή φυσική διεύθυνση μιας σελίδας που απεικονίζεται στην κύρια μνήμη.Αντ' αυτού, οι εικονικές διευθύνσεις διαιρούνται σε ένα προσδιοριστικό εικονικό σελίδων αποκαλούμενο ως *εικονικό αριθμό σελίδων*, ή VPN, και ένα σύνολο bits που περιγράφει την μετατόπιση από την έναρξη της εικονικής σελίδας στην εικονική διεύθυνση.Οι φυσικές σελίδες όμοια διαιρούνται σε *φυσικό αριθμό σελίδων* (PPN) και στη μετατόπιση από την έναρξη της φυσικής σελίδας στη φυσική διεύθυνση, όπως φαίνεται στο σχήμα 8-3.Οι εικονικές και φυσικές σελίδες σε ένα δεδομένο σύστημα έχουν γενικά το ίδιο μέγεθος, έτσι ο αριθμός των bits (log2 του μεγέθους της σελίδας) που απαιτείται για να κρατήσει τον τομέα μετατόπισης των εικονικών και φυσικών διευθύνσεων είναι ο ίδιος, αν και το VPN και το PPN μπορούν να είναι διαφορετικά μήκη.Πολλά συστήματα, ιδιαίτερα συστήματα 64-bits, έχουν τις πιό μακροχρόνιες εικονικές διευθύνσεις από τις φυσικές διευθύνσεις, λαμβάνοντας υπόψη το τρέχον θεωριτικό της οικοδόμησης ενός συστήματος με 264 bytes της μνήμης DRAM .



ΣΧΗΜΑ8-4 converting virtual addresses to physical addresses

Όταν μια εικονική διεύθυνση μεταφράζεται, το λειτουργικό σύστημα ανατρέχει στην είσοδο που αντιστοιχεί στο VPN στον πίνακα σελίδων και επιστρέφει την αντιστοιχία PPN.Τα bits μετατόπισης της εικονικής διεύθυνσης συνδέονται έπειτα επάνω στο PPN για να παραγάγουν τη φυσική διεύθυνση, όπως φαίνεται στο σχήμα 8-4.

8.4 Απαίτηση σελιδοποίησης εναντίον ανταλλαγής

Το σύστημα εικονικής μνήμης που περιγράφεται ανωτέρω είναι ένα παράδειγμα της *απαίτησης σελιδοποίησης*, ο τύπος εικονικής μνήμης είναι ο συνηθέστερος που χρησιμοποιείται σήμερα.Στην απαίτηση σελιδοποίησης,οι σελίδες των στοιχείων παρουσιάζονται μόνο στην κύρια μνήμη όταν έχει πρόσβαση ένα πρόγραμμα σε αυτες.Όταν εμφανίζεται ένας διακόπτης πλαισίου, το λειτουργικό σύστημα δεν αντιγράφει καμία από τις παλίες σελίδες του προγράμματος έξω στο δίσκο ή καμία από τις νέες σελίδες του προγράμματος στην κύρια μνήμη.Αντ' αυτού, αρχίζει ακριβώς η εκτέλεση του νέου προγράμματος και εξάγει τις σελίδες του προγράμματος όπως αυτές αναφέρθηκαν.

Η *ανταλλαγή* είναι μια σχετική τεχνική που χρησιμοποιεί τα μαγνητικά μέσα για να αποθηκεύσει την κατάσταση των προγραμμάτων που δεν τρέχουν αυτήν την περίοδο στον επεξεργαστή.Σε ένα σύστημα που χρησιμοποιεί την ανταλλαγή, τα λειτουργικά συστήματα μεταχειρίζονται τα στοιχεία ενός προγράμματος ως ατομική μονάδα και κινούν όλα τα στοιχεία μεσα ή έξω από την κύρια μνήμη συγχρόνως.Όταν το λειτουργικό σύστημα στον υπολογιστή που χρησιμοποιει την ανταλλαγή επιλέγει ένα πρόγραμμα για να τρέξει στον επεξεργαστή, αυτό φορτώνει τα στοιχεία όλου του προγράμματος στην κύρια μνήμη, εκδιώκοντας άλλα προγράμματα από την κύρια μνήμη εάν είναι απαραίτητο.

Εάν όλα τα προγράμματα που εκτελούνται για τον υπολογιστή αρμόζουν στην κύρια μνήμη (μετρώντας και τις οδηγίες τους και τα στοιχεία), και οι η απαιτηση σελιδοποιησης και η ανταλλάγη επιτρέπουν στον υπολογιστη να λειτουργήσει μέσα σε μια πολυπρογραμματική κατασταση χωρίς να πρέπει να προσκομιστούν τα στοιχεία από το δίσκο.Τα συστήματα ανταλλάγης έχουν το πλεονέκτημα ότι, μόλις ένα προγραμμα εξάχθει από το δίσκο,όλα τα στοιχεία του προγράμματος απεικονίζονται στην κύρια μνήμη.Αυτό κάνει το χρόνο εκτέλεσης του προγράμματος πιό προβλέψιμο, δεδομένου ότι τα ελαττώματα σελίδων δεν εμφανίζονται ποτέ κατά τη διάρκεια της χρήσης ενός προγράμματος της ΚΜΕ.

Τα συστήματα απαίτησης-σελιδοποίησης έχουν το πλεονέκτημα ότι προσκομίζουν μόνο τις σελίδες των στοιχείων που ένα πρόγραμμα χρησιμοποιεί πραγματικά από το δίσκο. Εάν ένα πρόγραμμα πρέπει μόνο να παραπέμψει ένα μέρος των στοιχείων του κατά τη διάρκεια κάθε εκτέλεσης χρονοθυρίδας, αυτό μπορεί σημαντικά να μειώσει τα ξοδευμένα στοιχεία αντιγραφής χρονικού διαστήματος προς και από το δίσκο.Επίσης, τα συστήματα που χρησιμοποιούν την ανταλλαγη τυπικά δεν μπορούν να χρησιμοποιήσουν τα μαγνητικά μέσα αποθήκευσής τους για να επιτρέψουν σε ένα ενιαίο πρόγραμμα να παραπεμψει περισσότερα στοιχεία που ταιριάζουν στην κύρια μνήμη, επειδή τα στοιχεία όλων των προγραμμάτων πρέπει να ανταλλαχθούν προς ή από την κύρια μνήμη ως μονάδα.Στα συστήματα απαίτησης-σελιδοποίησης ,μεμονωμένες σελίδες των στοιχείων ενός προγράμματος μπορεί να παρουσιαστουν στη μνήμη όπως απαιτείται, φτίαχνωντας το όριο στο μέγιστο ποσό δεδομένων που ένα πρόγραμμα μπορεί να παραπέμψει το διαθέσιμο ποσό διαστήματος στο δίσκο, όχι το ποσό κύριας μνήμης.Για τις περισσότερες εφαρμογές, τα πλεονεκτήματα της απαίτησης σελιδοποίησης ξεπερνούν τα μειονεκτήματα, κανοντας την απαίτηση σελιδοποίησης την επιλογή για τους περισσότερους τερματικούς σταθμούς/PC λειτουργικά συστήματα.

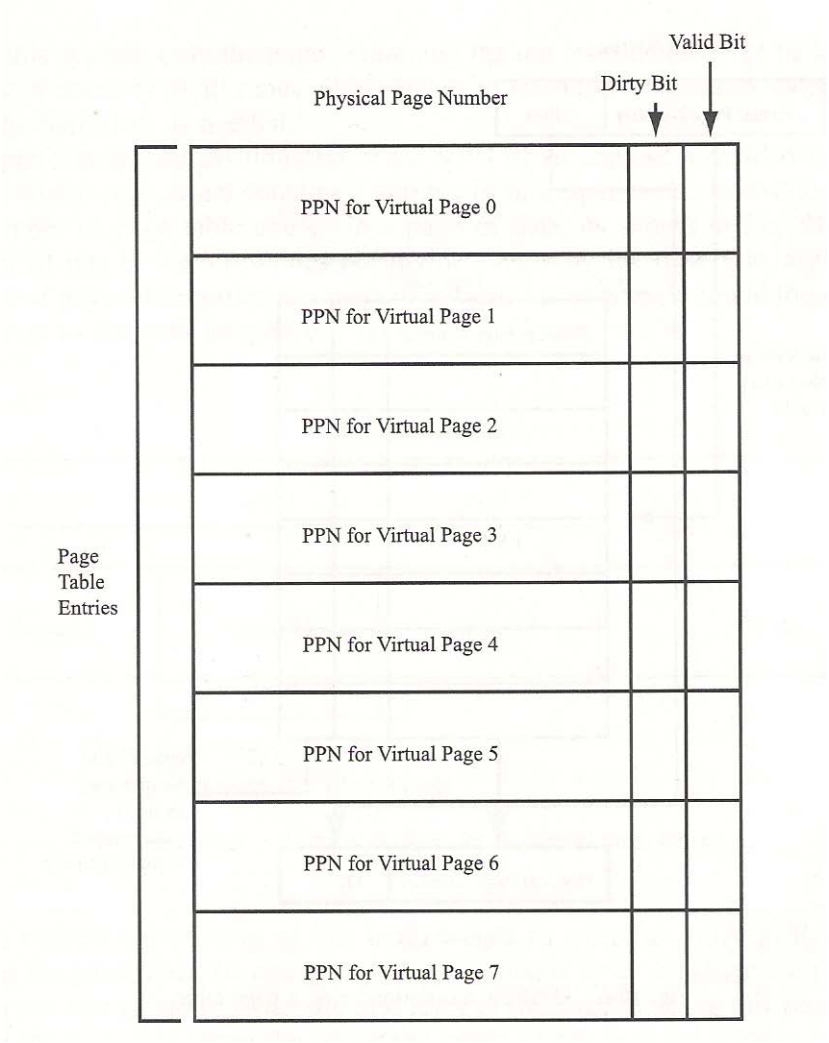
8.5 Πίνακες σελίδων

Όπως συζητήθηκε νωρίτερα στο κεφάλαιο, το λειτουργικό σύστημα χρησιμοποιεί μια δομή δεδομένων γνωστή ως πίνακα σελίδων για να παρακολουθήσει το πώς οι εικονικές διευθύνσεις απεικονίζουν στις φυσικές διευθύνσεις.Επειδή κάθε πρόγραμμα έχει σχεδιάσει τις εικονικές- φυσικές διευθύνσεις του,απαιτούνται για κάθε πρόγραμμα χωριστοί πίνακες σελίδων για το σύστημα.Ο απλούστερη εφαρμογή πίνακα σελίδων είναι ακριβώς μια σειρά απο καταχωρήσεις του πινάκα σελίδων, μια είσοδος ανά εικονική σελίδα, και είναι γνωστή ως απλού επιπέδου πίνακας σελίδων που διακρίνεται από τους πολλαπλου επιπέδου πίνακα σελίδων που θα περιγράψουμε αργότερα.Για να εκτελέσει μια μετάφραση διευθύνσεων, το λειτουργικό σύστημα χρησιμοποιεί τον εικονικό αριθμό σελίδων της διεύθυνσης ως δείκτη στη σειρά των καταχωρήσεων του πίνακα σελίδων για να εντοπίσει το φυσικό αριθμό σελίδων που αντιστοιχεί στην εικονική σελίδα.Αυτό παρουσιάζεται στο σχήμα 8-5, το οποίο παρουσιάζει ένα πίνακα σελίδων για ένα σύστημα του οποίου το εικονικό διάστημα διευθύνσεων είναι μόνο οκτώ μακροχρόνιες σελίδες.

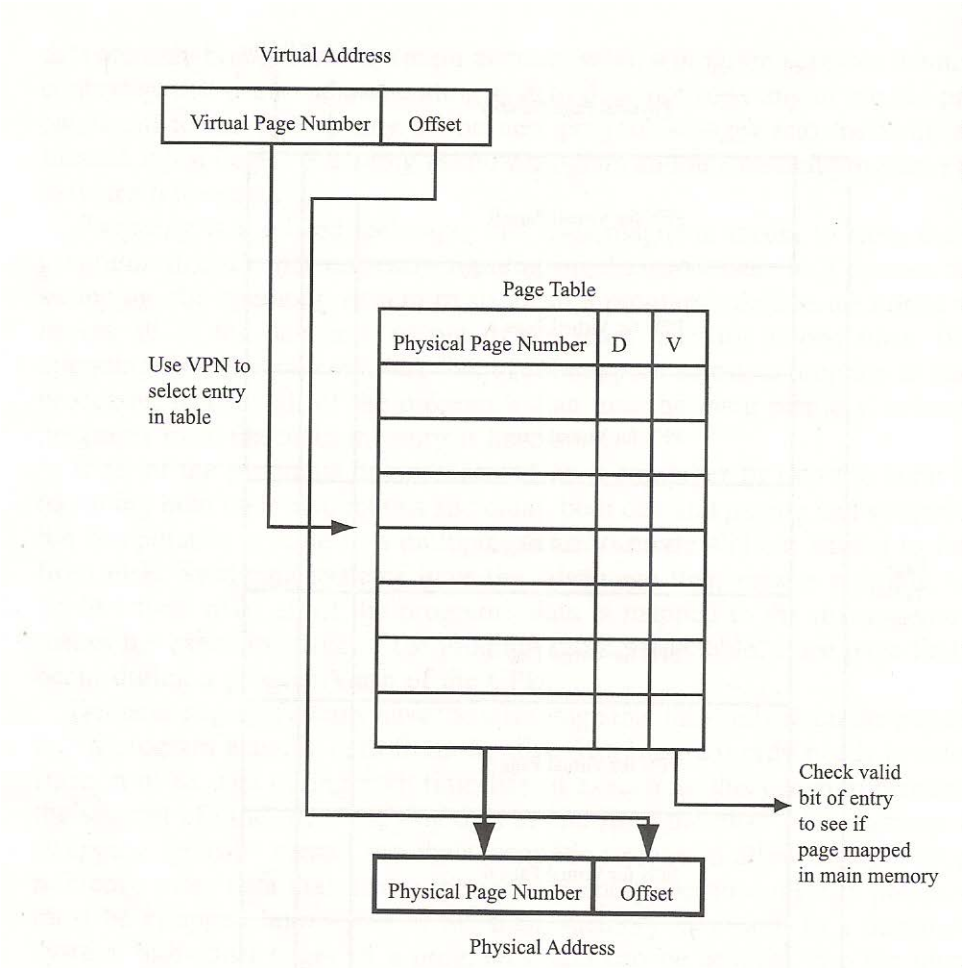
Όπως διευκρινίζεται στο σχήμα 8-5, οι καταχωρήσεις του πίνακα σελίδων περιέχουν γενικά έναν φυσικό αριθμό σελίδων,ένα έγκυρο bit, και ένα μη έγκυρο bit.Το έγκυρο bit κωδικοποιεί εάν η εικονική σελίδα που αντιστοιχεί στην είσοδο απεικονίζεται συγχρόνως στη φυσική μνήμη.Εάν το έγκυρο bit εφαρμόζεται, κατόπιν το πεδίο του φυσικού αριθμου σελίδων περιέχει το φυσικό αριθμό σελίδων του πλαισίου σελίδων που περιέχει τα στοιχεία της εικονικής σελίδας.Το μη έγκυρο bit καταγράφει εάν η σελίδα έχει γραφτεί ή όχι δεδομένου ότι παρουσιάστηκε στην κύρια μνήμη.Αυτό χρησιμοποιείται για να καθορίσει εάν το περιεχόμενο της σελίδας πρέπει να γραφτεί πίσω στο δίσκο όταν εκδιώκεται η σελίδα από την κύρια μνήμη, παρόμοια με το μη έγκυρο bit στο write-back της κρυφής μνήμης.(Λόγω του χρόνου που απαιτείται για

να έχει πρόσβαση στα μαγνητικά μέσα, όλα τα συστήματα εικονικής μνήμης είναι write-back αντί write-through.)

Στα διαγράμματα του σχήματος 8-6 φαίνεται η χρήση ενός πίνακα σελίδων για να μεταφράσει μια εικονική διεύθυνση.Τα συστήματα απλού επιπέδου πίνακες σελίδων γενικά απαιτούν η καταχώρηση του πίνακα σελίδων να διατηρειται στη φυσική μνήμη πάντα, έτσι ώστε το λειτουργικό σύστημα να μπορεί να έχει πρόσβαση στον πίνακα για τις μεταφράσεις των διευθύνσεων.



ΣΧΗΜΑ 8-5 single-level page table



ΣΧΗΜΑ8-6 address translation usingge table

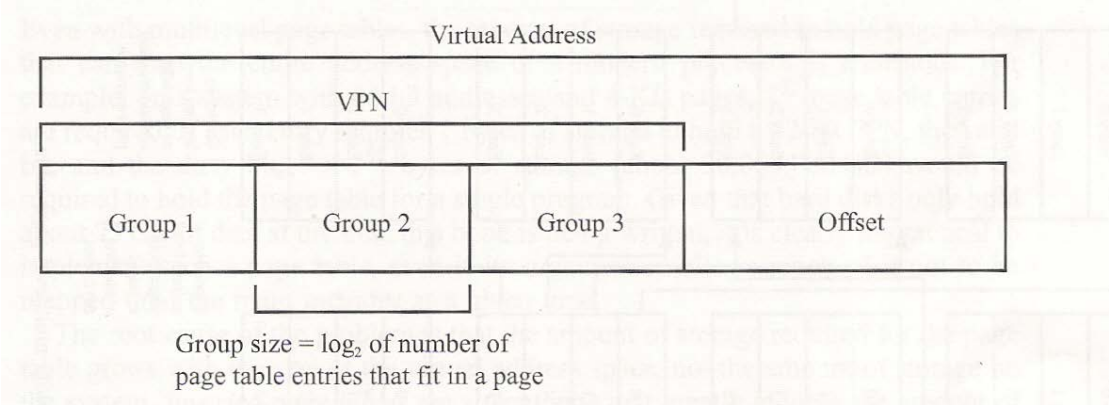
8.5.1 Πολλαπλής στάθμης πίνακες σελίδων

Οι πίνακες σελίδων μπορούν να απαιτήσουν μεγάλη αναλογία αποθήκευσης.Παραδείγματος χάριν, ένας απλού επιπέδου πίνακας σελίδων για ένα σύστημα με ένα 32 bit διάστημα διευθύνσεων (και εικονικός και φυσικός) και σελίδες 4-KB θα απαιτούσαν 220 καταχωρήσεις.Εάν κάθε είσοδος απαίτει 3 bytes της αποθήκευσης, η συνολική αποθήκευση που απαιτείται για τον πίνακα σελίδων θα ήταν 3 MB.Σε ένα σύστημα με 64 MB της κύριας μνήμης, αυτό θα απαιτούσε σχεδόν το 5 τοις εκατό της κύριας μνήμης να αφιερώνοταν στον πίνακα σελίδων, αρκετά μεγάλος πλεονάζων χώρος.Στα συστήματα με τη λιγότερη μνήμη,ο πλεονάζων χώρος θα ήταν ακόμα μεγαλύτερος, και δεν θα ήταν δυνατό να εφαρμοστεί η εικονική μνήμη καθόλου σε ένα σύστημα με λιγότερο από 3 MB της αποθήκευσης .

Για να εξετάσουν αυτό το πρόβλημα, οι σχεδιαστές χρησιμοποιούν τους πολλαπλής στάθμης πίνακες σελίδων, οι οποίοι επιτρέπουν σε ένα μεγάλο μέρος του πίνακα σελίδων να αποθηκευτεί στην εικονική μνήμη και να κρατηθεί στο δίσκο όταν δεν είναι σε χρήση.Σε έναν πολλαπλής

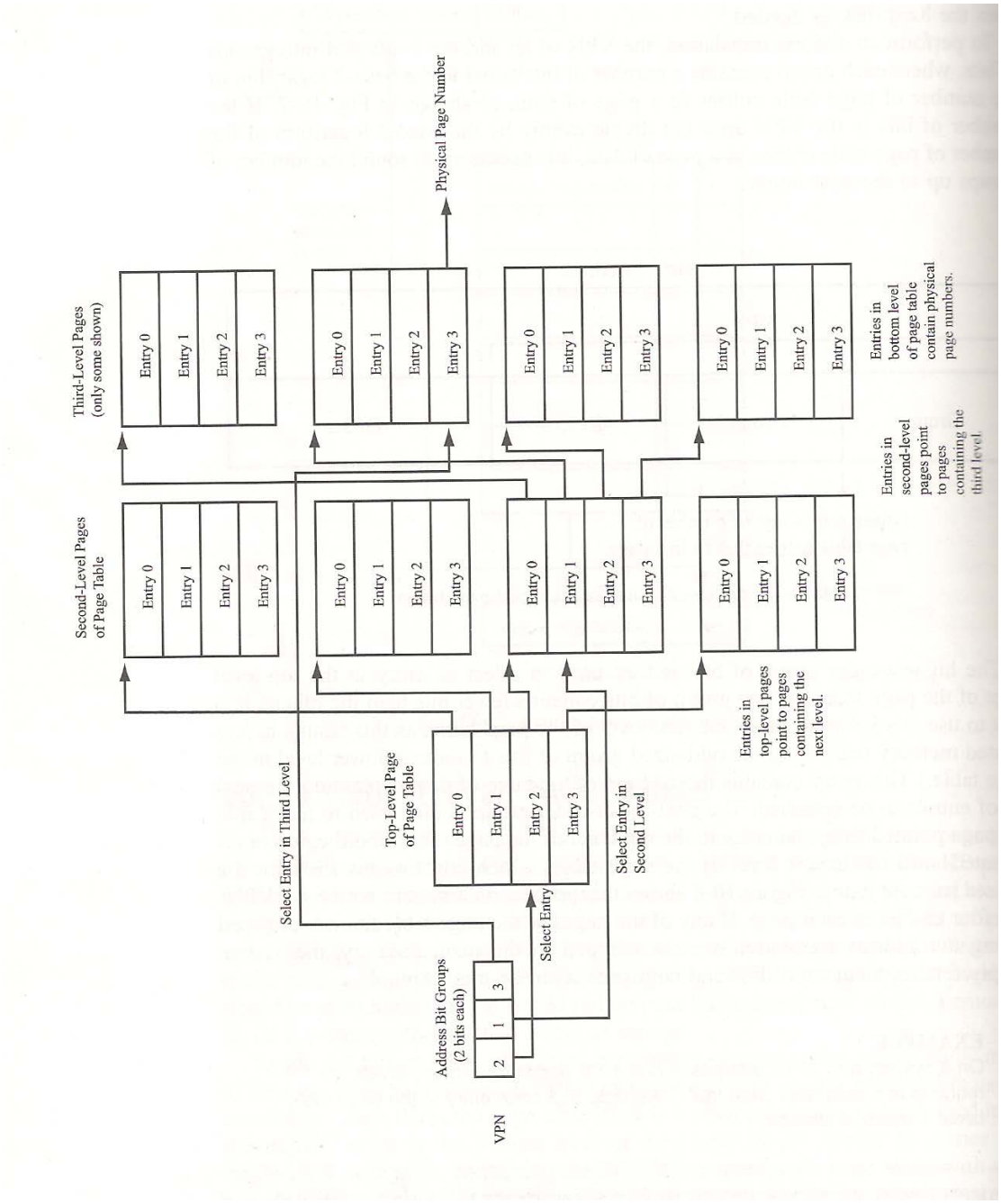
στάθμης πίνακα σελίδων, ο ίδιος ο πίνακας σελίδων είναι σπασμένος σε σελίδες και τακτοποιημένος σε μια ιεραρχία.Οι καταχωρήσεις στο κατώτατο επίπεδο της ιεραρχίας είναι παρόμοιες με τις καταχωρήσεις σε έναν μονοεπίπεδο πίνακα σελίδων, περιέχει το PPN της κατάλληλης σελίδας μαζί με τα έγκυρα και μη εγκυρα bits.Οι καταχωρήσεις στα άλλα επίπεδα του πίνακα σελίδων προσδιορίζουν τη σελίδα στη μνήμη που περιέχει το επόμενο επίπεδο της ιεραρχίας για τη διεύθυνση που μεταφράζεται .Χρησιμοποιώντας αυτό το σύστημα, μόνο η σελίδα που περιέχει το κορυφαίο επίπεδο του πίνακα σελίδων πρέπει να κρατηθεί στη μνήμη πάντα.Αλλες σελίδες στον πίνακα σελίδων μπορούν να αντιγραφουν προς και από το σκληρό δίσκο όπως απαιτούνται.

Για να εκτελέσει μια μετάφραση διευθύνσεων, το VPN μιας διεύθυνσης διαιρείται σε ομάδες bits, όπου κάθε ομάδα περιέχει διάφορα bit ίσα με το λογάριθμο βάση 2 του αριθμού των καταχωρήσεων του πίνακα σελίδων σε μια σελίδα των στοιχείων, όπως φαίνεται στο σχήμα 8-7.Εάν ο αριθμός bits στο VPN δεν διαιρεί ομοιόμορφα με το λογάριθμο βάση 2 του αριθμού των καταχωρήσεων του πίνακασελίδων σε μια σελίδα των στοιχείων, είναι απαραίτητο να στρογγυλευτεί ο αριθμός ομάδων μέχρι τον επόμενο ακέραιο αριθμό.



ΣΧΗΜΑ8-7 address division for multilevel page tables

Η υψηλού βαθμού ομάδα bits χρησιμοποιείται έπειτα για να επιλέξει μια είσοδο στην κορυφαία σελίδα επιπέδων του πίνακα σελίδων. (Εάν μια ομάδα bits περιέχει λιγότερα bits από άλλες, είναι καλύτερο να χρησιμοποιηθεί αυτή η ομάδα για να συντάξει το πρώτο επίπεδο του πίνακα σελίδων, αυτό οδηγεί στη λιγότερο σπαταλημένη μνήμη από τη χρησιμοποιείται η περίεργα-ταξινομημένη ομάδα bits για να συντάξει ένα χαμηλότερο επίπεδο στον πίνακα σελίδων.)Αυτή η είσοδος περιέχει τη διεύθυνση της σελίδας των στοιχείων που περιέχουν το επόμενο σύνολο καταχωρήσεων που αναζητώνται.Η επόμενη χαμηλότερη ομάδα διαταγής χρησιμοποιείται έπειτα για να βάλει δείκτη στη σελίδα που δείχνεται από την είσοδο στο κορυφαίο επίπεδο του πίνακα σελίδων, και η διαδικασία επαναλαμβάνεται έως ότου βρίσκεται στο χαμηλότερο επίπεδο του πίνακα σελίδων, που περιέχει το PPN για την επιθυμητή σελίδα.Το σχήμα 8-8 παρουσιάζει αυτήν την διαδικασία σε ένα σύστημα με το 6-bit VPNs και τέσσερις καταχωρήσεις σε κάθε σελίδα.Εάν οποιεσδήποτε από τις σελίδες στον πίνακα σελίδων που απαιτούνται κατά τη διάρκεια της μετάφρασης διευθύνσεων δεν απεικονίζουνται στην κύρια μνήμη, το σύστημα τις προσκομίζει απλά από το δίσκο και συνεχίζει με τη μετάφραση.



ΣΧΗΜΑ 8-8 address translation with multilevel page table

8.5.2 Αντίστροφοι πίνακες σελίδων

Ακόμη και με τους πολλαπλής στάθμης πίνακες σελίδων, το ποσό αποθήκευσης που απαιτείται για να κρατήσει τους πίνακες σελίδων που μπορούν να απεικονίσουν ολόκληρο το διάστημα διευθύνσεων ενός σύγχρονου επεξεργαστή είναι τεράστιο. Παραδείγματος χάριν, στο συστημα με τις 64-bit διευθύνσεις και τις 4-KB σελίδες,απαιτούνται 252 καταχωρήσεις του πίνακα σελίδων.Εάν κάθε είσοδος απαιτεί 7 bytes της αποθήκευσης για να κρατήσει ένα 52-bit PPN, το έγκυρο bit, και το μη έγκυρο bit, 7 X 252 bytes της αποθήκευσης (περίπου 30.000.000 MB) θα

απαιτούνταν για να κρατήσουν τον πίνακα σελίδων για ένα ενιαίο πρόγραμμα.Δεδομένου ότι οι σκληροί δίσκοι κρατούν περίπου 75 GB μόνο των στοιχείων στο χρόνο , είναι σαφώς μη πρακτικό να εφαρμοστεί ένας τέτοιος πίνακας σελίδων, ακόμα κι αν η οργάνωσή της επιτρέπει σε ένα μεγάλο μέρος από αυτον να μην απεικονθστεί επάνω στην κύρια μνήμη σε μία δεδομένη στιγμή .

Η πρωταρχική αιτία του προβλήματος είναι ότι το ποσό αποθήκευσης που απαιτείται για τον πίνακα σελίδων αυξάνεται με το μέγεθος του εικονικού διαστήματος διευθύνσεων, όχι το ποσό αποθήκευσης στο σύστημα.Οι πίνακες σελίδων είναι μια τεχνική που μειώνει πολύ το ποσό αποθήκευσης που απαιτείται για τους πίνακες σελίδων στα συστήματα με τα μεγάλα διαστήματα διευθύνσεων.Μία διεξοδική συζήτηση του πίνακα σελίδων είναι πέρα από το σκοπό αυτού του βιβλίου, αλλά η βασική ιδέα είναι ότι ο πινακας σελίδων αποτελείται απο ενα συνόλου καταχωρήσεων, μια για κάθε φυσικό πλαίσιο σελίδων στο σύστημα.Κάθε είσοδος στον πίνακα σελίδων περιέχει το VPN της εικονικής σελίδας που απεικονίζεται σε εκείνο το πλαίσιο σελίδων.Κατά συνέπεια, το ποσό αποθήκευσης που απαιτείται για τον πίνακα σελίδων εξαρτάται από το ποσό της κύριας μνήμης στο σύστημα, όχι το μέγεθος του εικονικού διαστήματος διευθύνσεων. Επειδή οι καταχωρήσεις του αντίστροφου πίνακα σελίδων οργανώνονται από τη φυσική διεύθυνση, όχι απο την εικονική διεύθυνση, είναι πιό δύσκολο να ψάξουν από τους συμβατικούς πίνακες σελίδων, δεδομένου ότι η εικονική διεύθυνση δεν μπορεί να χρησιμοποιηθεί για να καθορίσει την κατάλληλη είσοδο στον πίνακα σελίδων. Χαρακτηριστικά, οι δομές δεδομένων όπως οι αχρηστοι πίνακες χρησιμοποιούνται για να μειώσουν το χρόνο για να αναζητηθεί ο πίνακας σελίδων.

Μια άλλη προσέγγιση σε αυτό το πρόβλημα είναι να επιτραπεί στο πίνακα σελίδων να απεικονίσει ένα υποσύνολο του εικονικού διαστήματος διευθύνσεων αντί του ολόκληρου διαστήματος διευθυνσης.Στα συστήματα που χρησιμοποιούν αυτήν την προσέγγιση, ο πίνακας σελίδων περιέχει έναν πρόσθετο τομέα που δείχνει τη σειρά των διευθύνσεων που απεικονίζουνται από τον πίνακα σελίδων.Αυτή η προσέγγιση απαιτεί τις καταχωρήσεις του πίνακα σελίδων για ολόκληρη τη σειρά των εικονικών διευθύνσεων μεταξύ των χαμηλότερων και υψηλότερων διευθύνσεων που χρησιμοποιούνται από το πρόγραμμα, έτσι η αποδοτικότητά της εξαρτάται από τον εάν ένα πρόγραμμα διαθέτει τα στοιχεία πυκνά ή αραιά .

8.6 Μετάφραση κρυφής ενδίαμεσης μνήμης

Ένα σημαντικό μειονέκτημα της χρησιμοποίησης των πινάκων σελίδων για τη μετάφραση διευθύνσεων είναι ότι ο πίνακας σελίδων πρέπει να προσεγγιστεί σε κάθε αναφορά μνήμης.Σε ένα σύστημα με έναν απλού επιπέδου πίνακα σελίδων , αυτό διπλασιάζει τον αριθμό προσβάσεων μνήμης που απαιτούνται, δεδομένου ότι κάθε λειτουργία φορτωσης ή αποθήκευσης απαιτεί μια αναφορά μνήμης για να έχει πρόσβαση στην κατάλληλη είσοδο του πίνακα σελίδων και μια για να εκτελέσει την πραγματική φορτωση ή την αποθήκευση .Αυτό αυξάνει πολύ τη λανθάνουσα κατάσταση μιας αναφοράς μνήμης, και το πρόβλημα είναι ακόμα μεγαλύτερο στα συστήματα που έχουν τους πολλαπλής στάθμης πίνακες σελίδων, επειδή πολλαπλάσιες αναφορές μνήμης απαιτούνται για να διαπεράσουν τον πίνακα σελίδων.

Για να μειώσουν αυτο το μειονέκτημα, οι επεξεργαστές που προορίζονται να χρησιμοποιήσουν την εικονική μνήμη ενσωματώνουν τη μετάφραση κρυφής ενδίαμεσης μνήμης (TLBs) που ενεργεί ως κρύφή μνημη για τον πίνακα σελίδων.Όποτε ένα πρόγραμμα εκτελεί μια αναφορά μνήμης, η εικονική διεύθυνση στέλνεται στο TLB για να καθορίσει εάν περιέχει μια μετάφραση για τη διεύθυνση.Σε αυτή την περίπτωση, το TLB επιστρέφει τη φυσική διεύθυνση των στοιχείων, και η αναφορά μνήμης συνεχίζεται.Εάν όχι,μια TLB αποτυχία εμφανίζεται, και το σύστημα

ψάχνει τον πίνακα σελίδων για μια μετάφραση.Μερικά συστήματα παρέχουν το υλικό για να εκτελέσουν την πρόσβαση του πίνακα σελίδων σε μια TLB αποτυχία, ενώ άλλοι απαιτούν τη λειτουργία για να έχουν πρόσβαση στον πίνακα σελίδων στο λογισμικό.Το σχήμα 8-9 παρουσιάζει διαδικασία μεταφράσεων διευθύνσεων σε ένα σύστημα που περιέχει ένα TLB.

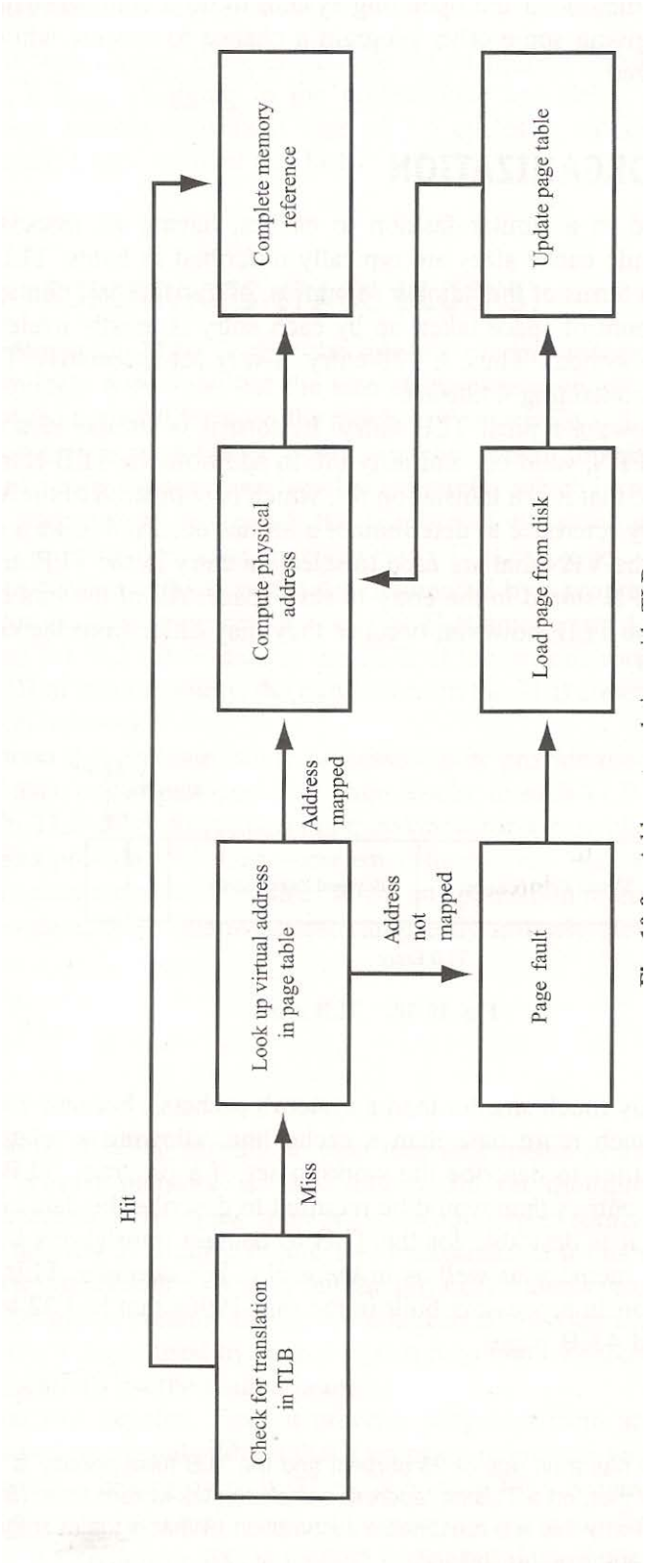
8.6.1 TLB αποτυχίες εναντίον ελαττωμάτων σελίδων

Σε ένα σύστημα που περιέχει ένα TLB, υπάρχουν τρία πιθανά σενάρια για το χειρισμό μιας αναφοράς μνήμης:

1. *Επιτυχία στο TLB* - σε αυτήν την περίπτωση, το TLB περιέχει μια μετάφραση για την εικονική διεύθυνση και η φυσική διεύθυνση της αναφοράς μπορεί να χρησιμοποιηθεί για να ολοκληρώσει την αναφορά μνήμης στο υλικό χωρίς συμμετοχή λογισμικού.Όταν μια σελίδα εκδιώκεται από την κύρια μνήμη,επίσης εκδιώκονται οι μεταφράσεις για τη σελίδα από το TLB, έτσι μια επιτυχία TLB σημαίνει ότι η φυσική σελίδα που περιέχει τη διεύθυνση απεικονίζεται στη μνήμη.
2. *Αποτυχία TLB, αλλά η σελίδα απεικονίζετα*ι - σε αυτήν την περίπτωση,το σύστημα προσπελαυνει το πίνακα σελίδων για να βρεί τη μετάφραση για την εικονική διεύθυνση, αντίγραφει τη μετάφραση στο TLB, και η αναφορά μνήμης προχωρεί.
3. *Αποτυχία TLB, και η σελίδα δεν απεικονίζεται* -σε αυτήν την περίπτωση,το σύστημα προσπελαυνει το πίνακα σελίδων , καθορίζει ότι η διεύθυνση δεν απεοκονιζεται, και ένα ελάττωμα σελίδων εμφανίζεται.Το λειτουργικό σύστημα φορτώνει έπειτα τα στοιχεία της σελίδας από το δίσκο με τον ίδιο τρόπο με ένα σύστημα εικονικής μνήμης που δεν περιέχει ένα TLB.

Οι TLB αποτυχίες και τα ελαττώματα σελίδων αντιμετωπίζονται πολύ διαφορετικά από το λειτουργικό σύστημα λόγω της διαφοράς στο χρονικό διάστημα που παίρνει για να επιλύσει, επειδή το σύστημα πρέπει ακριβώς να έχει πρόσβαση στον πίνακα σελίδων. Υποθέτοντας ότι κανένα ελάττωμα σελίδων δεν εμφανιζεται έχοντας πρόσβαση στον πίνακα σελίδων, οι TLB αποτυχίες μπορούν συνήθως να επιλυθούν σε μερικές εκατοντάδες κύκλους, έτσι ο χρήστης προγράμματος περιμένει ακριβώς μέχρι τη επίλυση της TLB αποτυχίας.

Τα ελαττώματα σελίδων, αφ' ετέρου, απαιτούν να προσκομίσουν στο δίσκο τη σελίδα.Η πρόσβαση ενός σκληρού δίσκου διαρκεί χαρακτηριστικά αρκετά χιλιοστά του δευτερολέπτου, ένα χρονικό διάστημα που είναι συγκρίσιμο με το χρονικό διάστημα όπου το λειτουργικό σύστημα αφήνει ένα πρόγραμμα να τρέξει πρίν δώσει μια άλλη πρόσβαση προγράμματος στον επεξεργαστή.(Πολλά λειτουργικά συστημάτα αλλάζουν θέση στα προγράμματα 60-100 φορές ανά δευτερόλεπτο, που επιτρέπουν σε κάθε πρόγραμμα να εκτελεσει για 16,7 με 10ms μεταξύ των διακοπτών πλαισίου.)Λαμβάνοντας υπόψη πόσο καιρό παίρνει για να χειριστεί ένα ελάττωμα σελίδων, δεν είναι ασυνήθιστο για το λειτουργικό σύστημα να γίνει ένας διακόπτης πλαισίου όταν εμφανίζεται ένα ελάττωμα σελίδων, δίνοντας σε κάποιο πρόγραμμα μια πιθανότητα να εκτελέσει ενώ το ελάττωμα σελίδων επιλύεται .

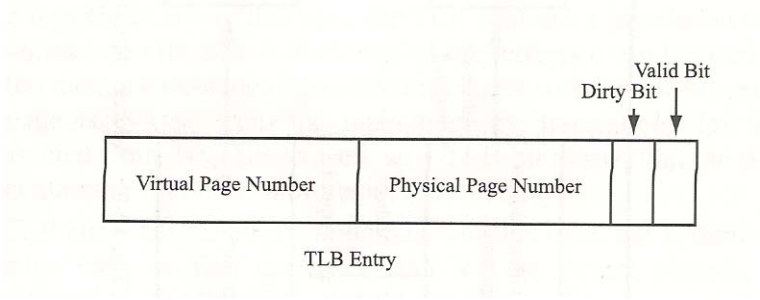


ΣΧΗΜΑ8-9 address tranclation with TLB

8.6.2 Οργάνωση TLB

Τα TLBs οργανώνονται με παρόμοιο τρόπο στις κρύφές μνήμες, που έχουν ένα συσχετισμό και έναν αριθμό συνόλων. Ενώ τα μεγέθη κρύφής μνήμης περιγράφονται χαρακτηριστικά στα bytes, τα μεγέθη TLB περιγράφονται συνήθως από την άποψη του αριθμού καταχωρήσεων ,ή των μεταφράσεων, που περιλαμβάνεται στο TLB, δεδομένου ότι το ποσό διαστήματος που απορροφείται από κάθε είσοδο είναι συνήθως άσχετο με την απόδοση του συστήματος.Κατά συνέπεια, 128 - είσοδοι, 4-τρόποι σύνολο - συσχετισμός TLB θα είχαν 32 σύνολα, κάθε ένα περιέχει 4 καταχωρήσεις.

Το σχήμα 8-10 παρουσιάζει μια τυπική είσοδο TLB. Το σχήμα του είναι παρόμοιο με μια καταχώρηση του πίνακα σελίδων, που περιέχει ένα PPN, ένα έγκυρο bit ,και ένα μη έγκυρο bit. Επιπλέον, η είσοδος TLB περιέχει το VPN της σελίδας όπου είναι μια μετάφραση , η οποία συγκρίνεται με το VPN της διεύθυνσης μιας αναφοράς μνήμης για να καθορίσει εάν έχει συμβεί επιτυχία.Όπως μία ετικέτα σειράς καταχώρησης μιας κρύφής μνήμης, τα bits του VPN που χρησιμοποιούνται για να επιλέξουν μια είσοδο στο TLB παραλείπονται χαρακτηριστικά από το VPN που αποθηκεύεται στην είσοδο για να σώσει το διάστημα.Όλα τα bits του PPN πρέπει να αποθηκευτούν στο TLB, εντούτοις, επειδή μπορούν να διαφέρουν από την αντιστοιχία στο VPN.



ΣΧΗΜΑ7-10 TLB entry

Τα TLBs είναι χαρακτηριστικά πολύ μικρότερα από την κρύφή μνήμη ενός συστήματος, επειδή κάθε είσοδος σε ένα TLB αναφέρεται σε πολύ περισσότερο στοιχεία από ότι μια γραμμή κρύφής μνήμης, που επιτρέπει σε έναν σχετικά μικρό αριθμό καταχωρήσεων TLB να περιγράψει το σύνολο εργασίας ενός προγράμματος.Τα TLBs περιέχουν τυπικά πολλές περισσότερες καταχωρήσεις από αυτες που απαιτούταν για να περιγράψει τα στοιχεία που περιλαμβάνονται στην κρύφή μνήμη, επειδή αυτό είναι επιθυμητό για το TLB να περιέχει τις μεταφράσεις για το στοιχείο που κατοικεί στην κύρια μνήμη καθώς επίσης και στην κρύφή μνήμη.Παραδείγματος χάριν, TLBs με 128 καταχωρήσεις ήταν κοινά στους επεξεργαστές που κατασκευάστηκαν στο μέσο - η δεκαετία του '90 όπου είχε 32 έως 64 KB πρώτου-επιπέδου κρύφής μνήμης σελίδες 4 KB.

8.6.3 SUPERPAGES (Φραγμοί σελίδων)

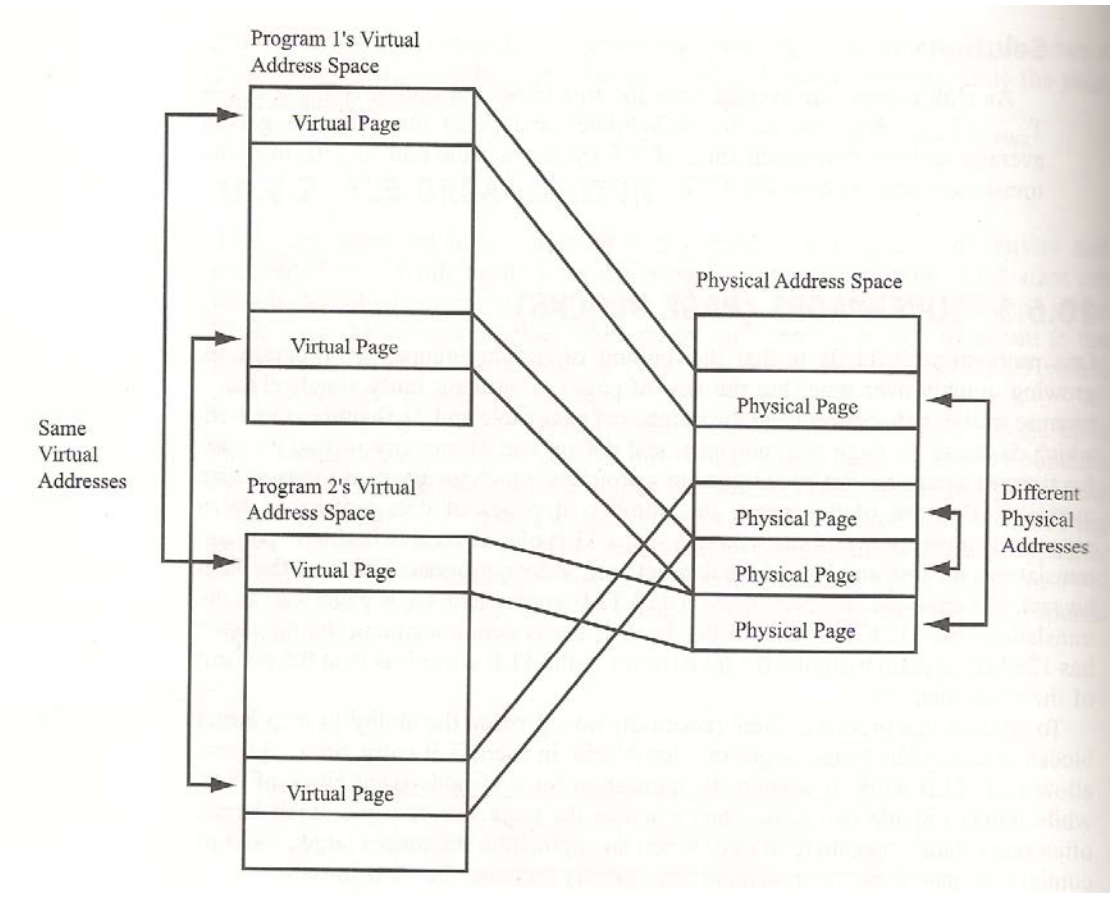
Ένα πρόβλημα με TLBs είναι ότι το ποσό στοιχείων που παραπέμπεται από το πρόγραμμα αυξάνεται γρήγορα κατά τη διάρκεια του χρόνου, αλλά το μέγεθος των σελίδων αυξάνεται αρκετά αργά.Αυτό είναι λόγω της ανταλλαγής μεταξύ του αριθμού του πίνακα σελίδων και των καταχωρήσεων TLB που απαιτούνται, οι οποίοι μειώνονται καθώς το μεγέθος των σελίδων αυξάνεται, και το ποσό μνήμης σπατάλίεται επειδή τα μέρη μιας σελίδας δεν μπορούν να οριστούν σε ένα πρόγραμμα, το οποίο μειώνεται καθώς το μεγέθος των σελίδων αυξάνεται.Λόγω αυτής της τάσης, ο αριθμός σελίδων των στοιχείων που παραπέμπονται από ένα

πρόγραμμα αυξάνεται κατά τη διάρκεια του χρόνου, που σημαίνει ότι ένα TLB ενός δεδομένου μεγέθους είναι σε θέση να περιέχει τις μεταφράσεις για όλο και λιγότερα στοιχεία που παραπέμπονται από το πρόγραμμα, που μειώνει το ποσοστό επιτυχίας TLB.Παραδείγματος χάριν, ένας επεξεργαστής με 128 καταχωρήσεις TLB και σελίδες 4-KB μπορεί να αποθηκεύσει τις μεταφράσεις για 512 KB των στοιχείων στο TLB.Εάν το σύστημα που περιέχει τον επεξεργαστή έχει 128 MB της κύριας μνήμης, οι μεταφράσεις στο TLB καλύπτουν λιγότερο από 0,5 τοις εκατό της κύριας μνήμης.

Για να εξετάσουν αυτό το πρόβλημα, μερικοί επεξεργαστές παρέχουν τώρα τη δυνατότητα να απεικονίσουν τους μεγαλύτερους φραγμούς των στοιχείων, αποκαλούμενα ως superpages ή φραγμούς σελίδων, σε κάθε είσοδο TLB.Μερικά συστήματα επιτρέπουν σε κάθε είσοδο TLB να περιέχουν τη μετάφραση για ένα μεταβλητη-μεγέθους φραγμό δεδομένων, ενώ άλλα παρέχουν δύο μεγέθη -το ένα ίσο με το μέγεθος σελίδων και ένα άλλο πολύ μεγαλύτερο, συχνά περισσότερο από1 megabyte στο μέγεθος.Όταν οι εφαρμογές αναφέρουν μεγάλους φραγμούς των παρακείμενων στοιχείων, αυτές οι βελτιώσεις μπορούν πολύ να αυξήσουν το ποσοστό επιτυχίας TLB .

8.7 Προστασία

Εξάλλου για να επιτρέψουν στα συστημάτα να αντιμετωπισούν τα μαγνητικά μέσα όπως ένα επίπεδο της ιεραρχίας μνήμης,η εικονική μνήμη είναι επίσης πολύ χρήσιμη στα πολαπλών προγραμμάτων συστήματα υπολογιστών επειδή παρέχουν την προστασία μνήμης, που εμποδίζει ένα πρόγραμμα από την πρόσβαση των στοιχείων απο κάτι άλλο.Το σχήμα 8-11 επεξηγεί πώς λειτουργεί αυτή η εργασία.Κάθε πρόγραμμα έχει το εικονικό διάστημα διευθύνσεών του, αλλά το φυσικό διάστημα διευθύνσεων μοιράζεται μεταξύ όλων των προγραμμάτων που τρέχουν για το σύστημα.Το σύστημα μεταφράσεων διευθύνσεων εξασφαλίζει ότι οι εικονικές σελίδες που χρησιμοποιούνται από κάθε πρόγραμμα απεικονίζουν επάνω στις διαφορετικές φυσικές σελίδες και τις διαφορετικές θέσεις στα μαγνητικά μέσα .



ΣΧΗΜΑ8-11 protection through virtual memory

Αυτό έχει δύο πλεονεκτήματα.Κατ' αρχάς, αποτρέπει τα προγράμματα από την πρόσβαση απο των στοιχείων απο κάτι άλλο,επειδή οποιαδήποτε εικονική διεύθυνση όπου ένα πρόγραμμα αναφέρει θα μεταφραστεί σε μια φυσική διεύθυνση που ανήκει σε αυτό. Δεν υπάρχει κανένας τρόπος για ένα πρόγραμμα να δημιουργησεί μια εικονική διεύθυνση οπου απεικονίζει επάνω σε μια φυσική διεύθυνση που ανήκει σε ένα άλλο πρόγραμμα, και έτσι δεν υπάρχει κανένας τρόπος για ένα πρόγραμμα για να προσεγγισει τα στοιχεία ενός άλλου προγράμματος.Εάν τα προγράμματα θέλουν να μοιραστούν τα στοιχεία το ένα με το άλλο, τα περισσότερα λειτουργικά συστήματα επιτρέπουν σε αυτα συγκεκριμένα να ζητήσουν μερικές από τις εικονικές σελίδες τους να απεικονιστούν επάνω στις ίδιες φυσικές διευθύνσεις.

Το δεύτερο πλεονέκτημα είναι ότι ένα πρόγραμμα μπορεί να δημιουργήσει και να χρησιμοποιήσει τις διευθύνσεις.Κατά συνέπεια, τα προγράμματα δεν πρέπει να γνωρίζουν πόσα άλλα προγράμματα τρέχουν στο σύστημα ή/και πόση μνήμη χρησιμοποιουν εκείνα τα άλλα προγράμματα.Κάθε πρόγραμμα έχει το δικό του εικονικό διάστημα διευθύνσεων και μπορεί να κάνει τον υπολογισμό διευθύνσεων και διαδικασιών μνήμης σε αυτό το δίαστημα διευθύνσεων, χωρίς να ανησυχεί για οποιαδήποτε άλλα προγράμματα που τρέχουν στη μηχανή.

Το μειονέκτημα αυτής της προσέγγισης είναι ότι η εικονική - φυσική απεικόνιση των διευθύνσεων γίνεται μέρος της κατάστασης ενός προγράμματος. Όταν το σύστημα μετακινηται από την εκτέλεση ενός προγράμματος στην εκτέλεση άλλου, πρέπει να αλλάξει τον πίνακα σελίδων που χρησιμοποιεί, και να ακυρώσει οποιεσδήποτε μεταφράσεις διευθύνσεων στο TLB.Διαφορετικά, το νέο πρόγραμμα θα χρησιμοποιούσε την εικονική - φυσική απεικόνιση των διευθύνσεων του παλαιού

προγράμματος, και θα ήταν σε θέση να έχει πρόσβαση στα στοιχεία του παλαιού προγράμματος.Αυτό αυξάνει τον πλεονάζων χώρο ενός διακόπτη πλαισίου, λόγω του χρόνου που απαιτείται για να ακυρώσει το TLB και να αλλάξει τον πίνακα σελίδων, και επειδή το νέο πρόγραμμα θα πάρει έναν μεγάλο αριθμό αποτυχιών TLB δεδομένου ότι αρχίζει με ένα άδειοTLB.

Μερικά συστήματα αντιμετωπίζουν αυτό το ζήτημα με την προσθήκη της διαδικασίας ID(ταυτοτητας) bits για κάθε καταχώρηση TLB που προσδιορίζει το πρόγραμμα για το οποίο η καταχώρηση ισχύει.Το υλικό συγκρίνει την ID διαδικασία του προγράμματος που κάνει μια αναφορά μνήμης στην διαδικασία ID στην μετάφραση ως τμήμα καθορισμού εάν μια επιτυχία TLB έχει συμβεί, επιτρέποντας μεταφράσεις από τα πολλαπλά προγράμματα για να παραμένει στο TLB συγχρόνως.Αυτό εξαλείφει την ανάγκη να ακυρωθεί το TLB σε κάθε διακόπτη πλαισίου αλλά αυξάνει το ποσό αποθήκευσης που απαιτείται για το TLB.

Στα σύγχρονα συστήματα, η εικονική μνήμη είναι συχνότερα ένα εργαλείο για να υποστηρίξει τα πολυπρογραμματα παρά ένα εργαλείο που επιτρέπει στα προγράμματα να χρησιμοποιησουν περισσότερη μνήμη που παρέχεται στην κύρια μνήμη του συστήματος.Το κόστος DRAM είναι αρκετά χαμηλό, και ο χρόνος για να επιλυθεί ένα ελάττωμα σελίδων αρκετά υψηλός,όπου τα περισσότερα συστήματα διαμορφώνονται με αρκετή κύρια μνήμη ενώ τα μεμονωμένα προγράμματα σπάνια, αν ποτέ, πρέπει να έχουν πρόσβαση στο επίπεδο μαγνητικών μέσων της ιεραρχίας μνήμης.Εντούτοις, έχοντας εικονική μνήμη για να παρέχει την προστασία μεταξύ των προγραμμάτων, και για να επιτρέψει την αυτόματη ανταλλαγή κάθε στοιχείου του προγράμματος μεταξύ του δίσκου κύριας μνήμης και σε έναν διακόπτη πλαισίου, είναι εξαιρετικά πολύτιμη, το οποίο εξηγεί γιατί ουσιαστικά όλα τα σύγχρονα λειτουργικά συστήματα και το υλικό υποστηρίζουν την εικονική μνήμη .

8.8 Κρυφή και εικονική μνήμη

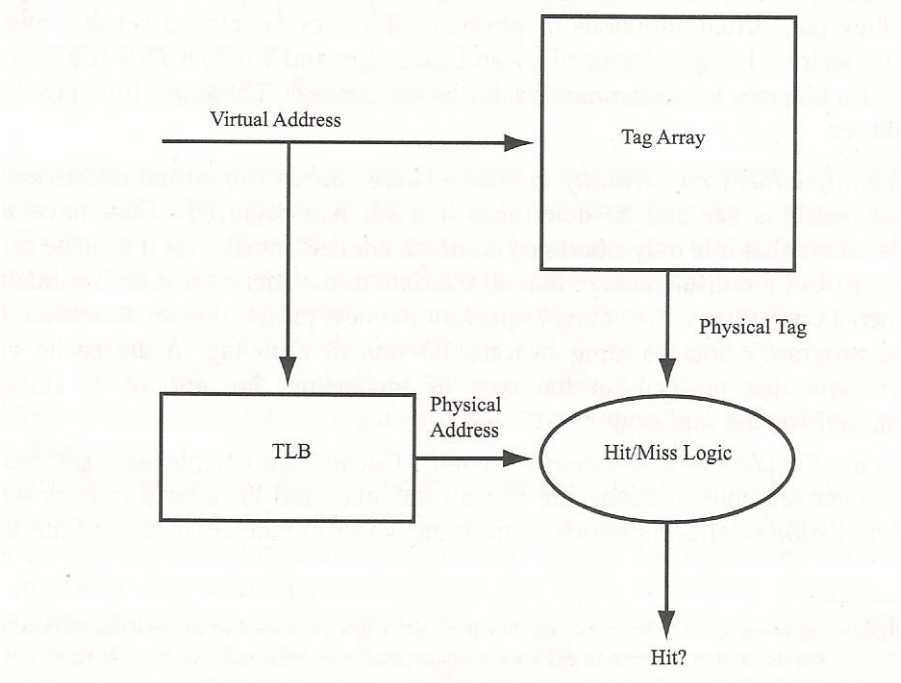
Πολλά συστήματα που χρησιμοποιούν την εικονική μνήμη ενσωματώνουν επίσης τις κρύφες μνήμες ως κορυφαία επίπεδα ή επίπεδα ιεραρχίας μνήμης τους.Στα συστήματα, οι εφαρμογές κρύφης μνήμης ποικίλλουν εάν χρησιμοποιούν τις εικονικές διευθύνσεις ή τις φυσικές διευθύνσεις για να επιλέξουν ένα σύνολο που να περιέχει τη διεύθυνση που παραπέμπεται από μια οδηγία και εάν χρησιμοποιούν τις εικονικές ή φυσικές διευθύνσεις για να καθορίσουν εάν η επιτυχία έχει συμβεί. Υπάρχουν τέσσερις πιθανοί συνδυασμοί:

1.Εικονικά διευθυνσιοδότημένη, εικονικά χαρακτηρισμένη - αυτές οι κρύφες μνήμες χρησιμοποιούν τις εικονικές διευθύνσεις για και να επιλέξουν ένα σύνολο και για να καθορίσουν εάν μια επιτυχία έχει συμβεί.Έχουν το πλεονέκτημα ότι είναι απαραίτητο να γίνει μια μετάφραση διευθύνσεων εάν μια αποτυχία κρύφης μνημης εμφανίζεται, αλλά έχουν το μειονέκτημα ότι όλα τα στοιχεία στην κρύφη μνήμη πρέπει να ακυρωθούν όταν εμφανίζεται ένας διακόπτης πλαισίου, για να αποτρέψουν το νέο πρόγραμμα από την πρόσβαση των στοιχείων του παλαιού προγράμματος.Η προσθήκη της ID διαδικασίας bits σε κάθε ετικέττα στην κρύφη μνήμη μπορεί να αποβάλει αυτό το πρόβλημα με κόστος το ποσό αποθήκευσης που απαιτείται για τη σειρά ετικεττών

- *Φυσικά διευθυνσιοδότημένη, φυσικά χαρακτηρισμένη* - οι κρύφες μνήμες που χρησιμοποιούν τις φυσικές διευθύνσεις για να επιλέξουν τα σύνολα και για να καθορίσουν εάν μια επιτυχία έχει εμφανιστεί δεν έχουν κανένα πρόβλημα με εικονικό aliasing διευθύνσεων - μεταξύ των προγραμμάτων και μπορούν να εκμεταλλευθούν το γεγονός ότι οι φυσικές διευθύνσεις είναι πιό σύντομες από τις εικονικές διευθύνσεις σε πολλά συστήματα για να μειώσουν το μέγεθος των σειρών ετικεττών τους.Εντούτοις, είναι απαραίτητο να εκτελεσθεί η

μετάφραση διευθύνσεων προτού να προσεγγιστεί η κρύφη μνήμη, η οποία αυξάνει το χρόνοεπιτυχίας κρύφης μνήμης .

- *Φυσικά διευθυνσιοδότημένη, εικονικά χαρακτηρισμένη* - αυτά τα συστήματα συνδυάζουν τις χειρότερες πτυχές της εικονικής και φυσικής εξέτασης και λίγο πολύ δεν χρησιμοποιούνται ποτέ.Η χρησιμοποίηση της φυσικής διεύθυνσης για να επιλέξει ένα σύνολο μέσα στην κρύφη μνήμη σημαίνει ότι η πρόσβαση κρύφης μνήμης πρέπει να περιμένει τη μετάφραση διευθύνσεων που ολοκληρώνει προτού να μπορέσει να αρχίσει, ενώ η εικονική ετικέττα σημαίνει ότι η κρύφη μνήμη δεν παρέχει την προστασία ενάντια στη χρήση της ίδιας εικονικής διεύθυνσης από τα πολλαπλάσια προγράμματα .
- *Εικονικά διευθυνσιοδότημένη,φυσικά χαρακτηρισμένη* - η χρησιμοποίηση των εικονικών διευθύνσεων για να επιλέξει ένα σύνολο μέσα στην κρύφη μνήμη και στις φυσικές διευθύνσεις για να καθορίσει εάν η επιτυχία που έχει εμφανιστεί επιτρέπει στη κρύφη μνήμη για να αρχίσει παράλληλα με τη μετάφραση διευθύνσεων αλλά παρέχει προστασία, αντίθετα από τα εικονικά διευθυνσιοδότημένα/εικονικά χαρακτηρισμένα συστήματα.Εφ' όσον η μετάφραση διευθύνσεων παίρνει λιγότερο χρόνο από την πρόσβαση στη σειρά ετικεττών, αυτός ο τύπος κρύφης μνήμης μπορεί να είναι τόσο γρήγορος όσο μια εικονικά διευθυνσιοδότημένη/εικονικά χαρακτηρισμένη κρύφη μνήμη, δεδομένου ότι η φυσική διεύθυνση δεν απαιτείται για τον προσδιορισμό επιτυχίας/αποτυχίας μέχρι να ολοκληρώνει η συμβούλευση σειράς ετικεττών.Τα TLBs τείνουν να περιέχουν λιγότερα στοιχεία από τη σειρά ετικεττών, έτσι η μετάφραση διευθύνσεων είναι γενικά γρηγορότερη από την πρόσβαση στη σειρά ετικεττών εκτός αν εμφανίζεται μια αποτυχία TLB.Αυτός ο συνδυασμός ταχύτητας και προστασίας κάνει τις εικονικα διευθυνσιοδότημένες/εικονικές χαρακτηρισμένες κρύφες μνήμες να είναι η επιλογή για τα περισσότερα τρέχοντα συστήματα.Το σχήμα 8-12 παρουσιάζει ένα διάγραμμα των προσβάσεων κρύφης μνήμης σε αυτόν τον τύπο κρύφης μνήμης.



ΣΧΗΜΑ8-12 virtually addreesses/ physically tagged cache access

Στα συστήματα με περισσότερα από ένα επίπεδο κρυφής μνήμης, μόνο το επίπεδο 1 εφαρμόζεται χαρακτηριστικά ως εικονικά διευθυνσιοδότημένη/εικονικά χαρακτηρισμένη, επειδή κάθε επίπεδο στην κρύφη μνήμη προσεγγίζεται στη σειρά.Οι χαμηλότερες κρύφες μνήμες είναι συνήθως οι φυσικά διευθυνσιοδότημένες/φυσικά χαρακτηρισμένες.Η μετάφραση των διευθύνσεων εκτελείται κατά τη διάρκεια του επιπέδου 1 της πρόσβασης κρύφης μνήμης, έτσι η φυσική διεύθυνση είναι διαθέσιμη ώσπου να γίνει η προσπελάση στο επίπεδο 2 και οι χαμηλότερες κρύφες μνήμες αρχίζουν, καθιστώντας αυτην απλούστερη και εξίσου γρήγορη στο να χρησιμοποιεί φυσικά διευθυνσιοδότημένες/φυσικά χαρακτηρισμένες κρύφες μνήμες για αυτά τα επίπεδα.Μερικά συστήματα ελέγχουν κάθε επίπεδο της κρύφης μνήμης παράλληλα με μια αναφορά μνήμης, για να μειώσουν το χρόνο πρόσβασης στα χαμηλότερα επίπεδα της κρύφης μνήμης. Τέτοια συστήματα μπορούν να χρησιμοποιήσουν τις εικονικά διευθυνσιοδότημένες/φυσικά χαρακτηρισμένες κρύφες μνήμες, για να επιτρέψουν τις προσβάσεις σε όλα τα επίπεδα της κρύφης μνήμης για να αρχίσουν αμέσως .

ΚΕΦΑΛΑΙΟ 9^{ο}

ΣΥΣΚΕΥΕΣ ΕΙΣΟΔΟΥ/ ΕΞΟΔΟΥ (I/O)

9.1 Στόχοι

Μέχρι τώρα συγκεντρωθήκαμε μόνο σε δύο από τα συστατικά στοιχεία ενός υπολογιστικού συστήματος: Τον επεξεργαστή και το σύστημα μνήμης. Οι συσκευές εισόδου/ εξόδου είναι το τρίτο σημαντικό στοιχείο των υπολογιστικών συστημάτων και είναι υπεύθυνες για την επικοινωνία με τον έξω κόσμο, αλλά και με την αποθήκευση πληροφοριών και την ανάκτησή τους. Σε αυτό το κεφάλαιο θα μελετήσουμε πώς οι I/O συσκευές αλληλεπιδρούν με τον επεξεργαστή και τη μνήμη, και θα αναλύσουμε την πιο κοινή συσκευή I/O: το σκληρό δίσκο.

9.2 Εισαγωγή

Οι αρχιτέκτονες υπολογιστών τείνουν να εστιάσουν την προσοχή τους, πρωτίστως στους επεξεργαστές, έπειτα στα συστήματα μνήμης και τελευταία στο σύστημα εισόδου/ εξόδου (Ε/Ε). Αυτό οφείλεται στα σημεία αναφοράς και τα μέτρα που χρησιμοποιούν για να συγκρίνουν δύο συστήματα, που γενικά είναι οι χρόνοι εκτέλεσης προγραμμάτων υπολογιστικά περίπλοκων προγραμμάτων που δεν χρησιμοποιούν ιδιαίτερα συσκευές Ε/Ε. Ακόμα οφείλεται σε κάποιες τεχνικές που χρησιμοποιούνται στην υλοποίηση των συσκευών Ε/Ε που δεν είναι εύκολο να βελτιώσουν την απόδοσή τους, όπως γίνεται με τις επιδόσεις του επεξεργαστή.
Παρόλα αυτά τα συστήματα Ε/Ε και ειδικά οι συσκευές αποθήκευσης είναι αποφασιστικής σημασίας για την απόδοση πολλών σημαντικών και προσοδοφόρων υπολογιστικών εφαρμογών στις μέρες μας. Τα συστήματα επεξεργασίας συναλλαγών, όπως αυτά που χρησιμοποιούνται για την κράτηση αεροπορικών εισιτηρίων ή οι συναλλαγές με κάρτα απαιτούν πολύ καλή επίδοση από τις συσκευές Ε/Ε, γιατί τα συστήματα αυτά πρέπει να εγγράψουν τα αποτελέσματα κάθε συναλλαγής σε κάποιο τύπο μόνιμης αποθήκευσης, όπως ο σκληρός δίσκος, πριν ολοκληρωθεί ακόμα η συναλλαγή. Το να αφήσεις τα αποτελέσματα στη μνήμη κάνει το σύστημα ευάλωτο στις διακοπές ενέργειας, καταρρεύσεις συστήματος και άλλα λάθη που καταστρέφουν τα δεδομένα στην κύρια μνήμη.
Οι επιδόσεις αυτών των συστημάτων μετριούνται με βάση τον αριθμό των συναλλαγών (όπως αγορές μα πιστωτική κάρτα), που το σύστημα μπορεί να εκτελέσει ανα πάσα στιγμή και το αναφέρει στο δίσκο. Συχνά, περιοριστικός παράγοντας είναι ο ρυθμός με τον οποίο μεταφέρονται τα δεδομένα από και προς το σκληρό δίσκο, με τον επεξεργαστή να σπαταλάει άεργο (idle) χρόνο περιμένοντας τα σύστημα Ε/Ε. Γιαυτό και οι αγοραστές τέτοιων συστημάτων βάσεων δεδομένων είναι διατεθειμένοι να πληρώσουν αρκετά χρήματα για βελτίωση στην επίδοση του συστήματος Ε/Ε, ενώ γενικά δεν ενδιαφέρονται το ίδιο για την ταχύτητα του επεξεργαστή.

Τα συστήματα Ε/Ε μπορούν να διαιρεθούν σε δύο μεγάλα συστατικά κομμάτια. Τις συσκευές Ε/Ε και τις τεχνολογίες που χρησιμοποιούνται για την αλληλεπίδρασή τους με το υπόλοιπο σύστημα, Αυτό το κεφάλαιο ξεκινά με μια συζήτηση σχετικά με τις

τεχνολογίες διεπαφής, όπως δίαυλοι, σήματα διακοπής, διαδοχική σταθμοσκόπηση και είσοδο/ έξοδο με απεικόνιση μνήμης. Δεδομένου της τεράστιας ποικιλίας των συσκευών Ε/Ε στις μέρες μας, μια συζήτηση εις βάθος για τις συσκευές ξεπερνά τα όρια και τους σκοπούς αυτού του βιβλίου. Όμως θα γίνει μια σύνοψη του τρόπου λειτουργίας των σκληρών δίσκων (μιας από τις πιο κοινές συσκευές αποθήκευσης δεδομένων) καθώς και του τρόπου σχεδίασης και προσπέλασής τους.

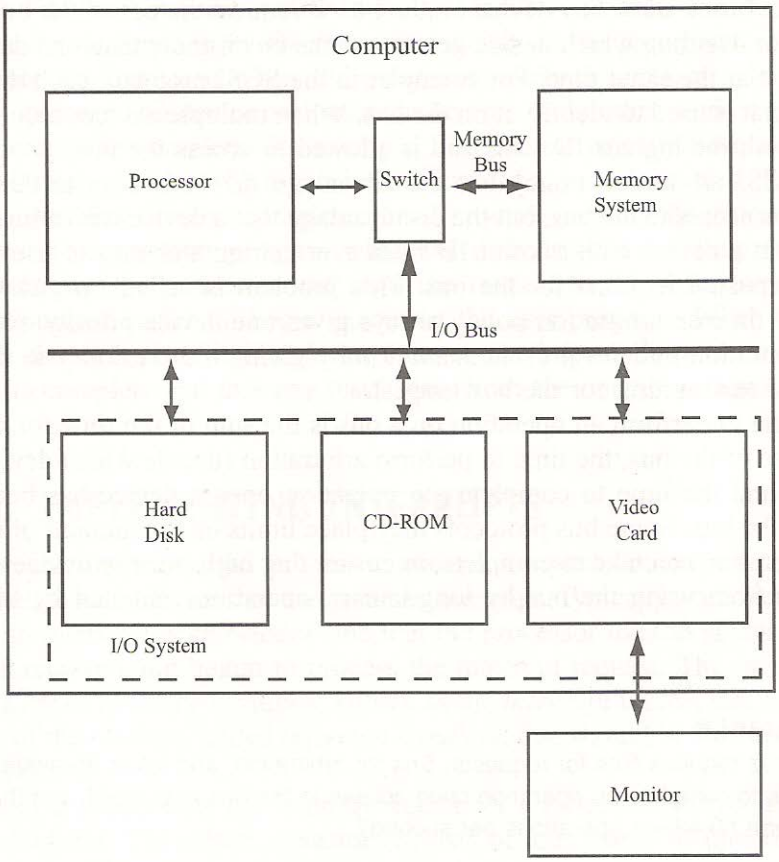
9.3 Δίαυλοι Ε/Ε

Στο κεφάλαιο 3 είδαμε για πρώτη φορά το διάγραμμα ενός τυπικού υπολογιστικού συστήματος, που το βλέπουμε και στο σχήμα 9-1. Ξεχωριστοί δίαυλοι μνήμης και Ε/Ε χρησιμοποιούνται για την επικοινωνία με τη μνήμη και το σύστημα Εισόδου/ Εξόδου. Αυτοί οι δίαυλοι επικοινωνούν με τον επεξεργαστή μέσω ενός διακόπτη (switch module).

Ο δίαυλος Εισόδου/ Εξόδου δημιουργεί μια διεπαφή που επιτρέπει στον υπολογιστή να αλληλεπιδρά με ένα ευρύ πεδίο συσκευών Ε/Ε, χρησιμοποιώντας ένα περιορισμένο σετ υλικό.

Κάθε δίαυλος Ε/Ε, όπως ο δίαυλος PCI που βρίσκεται στους περισσότερους προσωπικούς υπολογιστές και σταθμούς εργασίας, παρέχει μια προδιαγραφή για τον τρόπο μεταφοράς των εντολών και των δεδομένων ανάμεσα στον επεξεργαστή και τις συσκευές Ε/Ε καθώς και για το πόσες διαφορετικές συσκευές συναγωνίζονται για τη χρήση του διαύλου. Οποιαδήποτε συμβατή συσκευή με το δίαυλο μπορεί να προστεθεί (υποθέτοντας ότι υπάρχει κατάλληλο πρόγραμμα οδηγάς (driver)).Επίσης μια συσκευή που είναι συμβατή με ένα συγκεκριμένο τύπο διαύλου , μπορεί να λειτουργήσει σε οποιοδήποτε σύστημα με αυτό το δίαυλο. Αυτό κάνει τα συστήματα που χρησιμοποιούν διαύλους επικοινωνίας πιο ευέλικτα από εκείνα που εγκαθιστούν ξεχωριστά κυκλώματα επικοινωνίας ανάμεσα στον επεξεργαστή και κάθε συσκευή Ε/Ε, γιατί επιτρέπεται στο σύστημα να υποστηρίξει πολλές διαφορετικές συσκευές ανάλογα με τις ανάγκες των χρηστών, και ακόμα επιτρέπει την αλλαγή των συσκευών αυτών καθώς οι ανάγκες του χρήστη αλλάζουν.

Το κύριο μειονέκτημα των διαύλων επικοινωνίας είναι ότι έχουν προκαθορισμένο εύρος ζώνης το οποίο μοιράζονται μεταξύ τους όλες οι συσκευές που είναι συνδεδεμένες. Ακόμα περιορισμοί όπως το μήκος καλωδίου και επιρροές στη γραμμή μετάδοσης, μπορούν να προκαλέσουν μείωση του εύρους ζώνης του διαύλου αντί να χρησιμοποιεί το ίδιο εύρος για να συνδέσει δύο συσκευές. Γενικά υπάρχει μια ανταλλαγή μεταξύ εύρους ζώνης και απλότητας διεπαφής.



ΣΧΗΜΑ9-1 basic computer organization

9.3.1 Προσπέλαση του διαύλου Ε/Ε

Κάθε τύπος διαύλου (PCI, SCSI κτλ) έχει ένα καθορισμένο πρωτόκολλο για το πώς θα τον προσπελάζουν οι συσκευές, πόσα δεδομένα θα στέλνονται κτλ. Ένας από τους βασικούς παράγοντες είναι ή πολιτική διαιτησίας (arbitration policy), δηλαδή ο τρόπος που χρησιμοποιείται για να αποφασιστεί ποια συσκευή θα προσπελάσει το δίαυλο κάποια συγκεκριμένη χρονική στιγμή. Οι περισσότεροι δίαυλοι επιτρέπουν σε οποιαδήποτε συσκευή να τους χρησιμοποιήσει, και έχουν πολιτική διαιτησίας μόνο για περιπτώσεις που πάνω από μια συσκευές ζητήσουν πρόσβαση στο δίαυλο την ίδια στιγμή. Για παράδειγμα στο πρωτόκολλο SCSI, κάθε συσκευή έχει SCSI ID, που χρησιμοποιείται για την αναγνώρισή της από το δίαυλο. Όταν πολλές συσκευές διεκδικούν το δίαυλο, επιτρέπεται η πρόσβαση σε εκείνη τη συσκευή με μεγαλύτερο ID.

Η πολιτική διαιτησίας του SCSI έχει το πλεονέκτημα ότι είναι εύκολο να αποφασιστεί ποια συσκευή θα προσπελάσει το δίαυλο αλλά έχει και το μειονέκτημα ότι μια συσκευή με υψηλότερο ID μπορεί με διαδοχικές αιτήσεις να παίρνει συνέχεια την πρόσβαση από κάποια συσκευή με μικρότερο ID . Το πρόβλημα αυτό είναι γνωστό ως ανεπάρκεια (starvation) και μπορεί να συμβεί οποιαδήποτε στιγμή σε μια πολιτική διαιτησίας που δίνει προτεραιότητα σε κάποια συσκευή έναντι κάποιας άλλης. Κάποιες πολιτικές διαιτησίας προσπαθούν να προλάβουν ένα τέτοιο πρόβλημα

δίνοντας προτεραιότητα στια συσκευές που περιμένουν περισσότερο χρόνο για πρόσβαση στο δίαυλο.

Ο χρόνος εκτέλεσης μιας λειτουργίας στο δίαυλο είναι το άθροισμα του χρόνου που χρειάζεται μια συσκευή να ζητήσει τη χρήση του διαύλου, του χρόνου που χρειάζεται να γίνει επιλογή διαιτησίας(δηλ να αποφασιστεί ποια συσκευή θα χρησιμοποιήσει το δίαυλο) και του χρόνου που χρειάζεται για να ολοκληρωθεί η λειτουργία , από τη στιγμή που δίνεται η πρόσβαση σε μια συσκευή. Μερικοί δίαυλοι τοποθετούν περιορισμούς στο χρόνο που χρειάζεται μια λειτουργία να ολοκληρωθεί, για να διασφαλίσουν ότι οι συσκευές με υψηλή προτεραιότητα δε θα χάνουν τη σειρά τους από χρονοβόρες λειτουργίες που αφορούν συσκευές με χαμηλή προτεραιότητα.

9.4 Σήματα διακοπής (interrupts)

Πολλές συσκευές Ε/Ε δημιουργούν ασύγχρονα γεγονότα- δηλ. γεγονότα που συμβαίνουν σε στιγμές που ο επεξεργαστής δεν μπορεί να τα προβλέψει ή να τα ελέγξει, και στα οποία πρέπει να αντιδράσει αρκετά γρήγορα για να επιτύχει καλή επίδοση. Ένα τέτοιο παράδειγμα είναι το πληκτρολόγιο ενός σταθμού εργασίας ή pc. Ο επεξεργαστής δεν μπορεί να προβλέψει πότε ο χρήστης θα πατήσει ένα πλήκτρο αλλά πρέπει να αντιδράσει στο πάτημά του σε λιγότερο από 1 δευτερόλεπτο, αλλιώς η καθυστέρηση θα γίνει αντιληπτή από το χρήστη. Τα σήματα διακοπής είναι ο μηχανισμός αυτός που χρησιμοποιείται από τους περισσότερους επεξεργαστές για να χειριστούν τέτοια γεγονότα. Ουσιαστικά τα σήματα διακοπής επιτρέπουν στις συσκευές να αιτηθούν παύση του επεξεργαστή από ότι κάνει εκείνη τη στιγμή και να εκτελέσει το λογισμικό που θα επεξεργαστεί την αίτησή τους, πράγμα αρκετά ίδιο με την κλήση διαδικασίας που ξεκινά από μια εξωτερική συσκευή αντί από το πρόγραμμα που τρέχει στον επεξεργαστή.

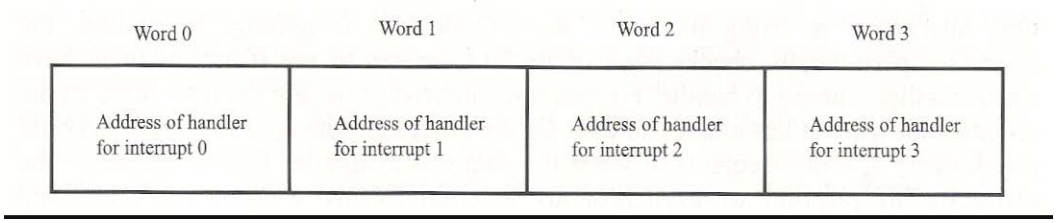
Τα σήματα διακοπής χρησιμοποιούνται επίσης όταν ο επεξεργαστής χρειάζεται να εκτελέσει κάποια χρονοβόρα εφαρμογή σε κάποια συσκευή αλλά παράλληλα να κάνει και άλλες εργασίες μέχρι αυτή να τελειώσει. Παραδείγματος χάρη, οι μηχανισμοί σκληρού δίσκου τυπικά έχουν χρόνους πρόσβασης περίπου 10ns, που αντιστοιχούν σε εκατομμύρια κύκλους του επεξεργαστή. Έτσι αντί να περιμένει το δίσκο να τελειώσει μια αίτηση ανάγνωσης ή εγγραφής, ο επεξεργαστής μπορεί να εκτελέσει κάποιο άλλο πρόγραμμα παράλληλα. Με τη χρήση σημάτων διακοπής ο επεξεργαστής στέλνει την αίτηση στο δίσκο, έπειτα με αλλαγή περιεχομένου αρχίζει την εκτέλεση του προγράμματος. Όταν ο δίσκος ολοκληρώσει την εργασία με σήμα διακοπής ενημερώνεται ο επεξεργαστής ότι τα δεδομένα που ζήτησε είναι έτοιμα.

9.4.1. Υλοποίηση σημάτων διακοπής

Για την υλοποίηση των σημάτων διακοπής ο υπολογιστής αναθέτει σε μια γραμμή γνωστή ως γραμμή σήματος αίτησης διακοπής (interrupt request line) σε κάθε συσκευή που μπορεί να εκπέμψει τέτοιο σήμα.. Τυπικά σε κάθε συσκευή ανατίθεται και μια γραμμή ως αναγνωριστικό διακοπής (interrupt acknowledge line)που επιτρέπει στον επεξεργαστή να αναγνωρίσει τη συσκευή που ζήτησε διακοπή, για να μπορέσει να επεξεργαστεί το αίτημά της. Ο επεξεργαστής επίσης παρέχει ένα σετ θέσεων μνήμης, που είναι γνωστό ως διάνυσμα σήματος διακοπής (interrupt vector) και περιέχει τις διευθύνσεις των σειρών εντολών (οι οποίες εντολές λέγονται

χειριστές διακοπών-interrupt handler), που πρέπει να εκτελεστούν όταν γίνει μια διακοπή.

Το σχήμα 9-2 δείχνει πώς το διάνυσμα σήματος διακοπής για ένα υπολογιστή με 4 διακοπές, μπορεί να σχεδιαστεί. Το διάνυσμα σήματος διακοπής αποτελείται από τέσσερις λέξεις μνήμης, μία για κάθε διακοπή. Όταν ο επεξεργαστής τροφοδοτείται αυτές οι θέσεις μνήμης περιέχουν αδιευκρίνιστες τιμές. Το λογισμικό (συνήθως το λειτουργικό αλλά και κάποιο πρόγραμμα μερικές φορές) πρέπει να αποθηκεύσει τις διευθύνσεις των χειριστών διακοπών για κάθε διακοπή, στη σωστή θέση του διανύσματος σήματος διακοπής, πριν οι συσκευές που αιτήθηκαν τη διακοπή αρχίσουν τη λειτουργία τους.



ΣΧΗΜΑ9-2 example interrupt vector

Για να διακοπεί ο επεξεργαστής η συσκευή στέλνει ένα σήμα στη γραμμή διακοπών της. Τυπικά η συσκευή επιτρέπεται να συνεχίσει την αποστολή σημάτων μέχρι να αντιληφθεί ο επεξεργαστής τη διακοπή. Όταν ο επεξεργαστής δέχεται ένα σήμα διακοπής, στέλνει ένα άλλο σήμα μέσω της γραμμής αναγνωριστικού διακοπής για να ενημερώσει τη συσκευή ότι το αίτημά της έχει ληφθεί. Έπειτα αναζητεί τη θέση του διανύσματος σήματος διακοπής για να βρει τη διεύθυνση του χειριστή διακοπών και αλλάζει τα περιεχόμενά του ώστε να αρχίσει η εκτέλεση του χειριστή διακοπών. Η αλλαγή περιεχομένων είναι απαραίτητη για να διασφαλιστεί ότι το πρόγραμμα που έτρεχε στον επεξεργαστή όταν συνέβη η διακοπή μπορεί να συνεχίσει τη λειτουργία του από εκεί που έμεινε όταν η διακοπή ολοκληρωθεί. Όταν ολοκληρωθεί ο χειριστής διακοπών αλλάζουν πάλι τα περιεχόμενα και η εκτέλεση γυρίζει σε κάποιο πρόγραμμα, όχι αναγκαστικά αυτό που έτρεχε κατά τη διακοπή. Γενικά οι διακοπές είναι αόρατες στα προγράμματα χρηστών με τον ίδιο τρόπο που δε βλέπει το ένα το άλλο. Ο μόνος τρόπος για ν καταλάβει ένα πρόγραμμα ότι συνέβη διακοπή είναι να προσπελάσει το ρολόι πραγματικού χρόνου του επεξεργαστή και να καταλάβει ότι πέρασε περισσότερος χρόνος από τον κανονικό ανάμεσα σε δύο γεγονότα.

9.4.2. Προτεραιότητες διακοπών

Αφού οι διακοπές είναι ασύγχρονες, είναι πιθανό περισσότερες από μία συσκευές να διεκδικήσουν μια διακοπή στον ίδιο χρόνο, ή πολλαπλές διακοπές να αιτηθούν κατά τη διάρκεια του χειρισμού μιας διακοπής. Για να αποφασιστεί η σειρά με την οποία οι διακοπές θα διευθετηθούν οι περισσότεροι επεξεργαστές εκχωρούν ένα βαθμό προτεραιότητας σε κάθε αίτηση, για να διακρίνει ποια είναι υψηλότερης και πιο χαμηλότερης σημασίας.(πχ. αυτή με το μικρότερο αριθμό μπορεί να είναι υψηλότερης

προτεραιότητας από κάποια με μεγαλύτερο αριθμό προτεραιότητας.) έτσι όταν περισσότερα από ένα σήματα διακοπών περιμένουν τον επεξεργαστή, εκείνος επιλέγει το σήμα με την υψηλότερη προτεραιότητα. Μερικοί επεξεργαστές επιτρέπουν σε σήματα υψηλής προτεραιότητας να διακόψουν το χειρισμό μιας διακοπής με μικρότερο αριθμό προτεραιότητας, ενώ άλλοι περιμένουν πάντα την ολοκλήρωση του χειρισμού της διακοπής, πριν προβούν στο χειρισμό καινούριας διακοπής.

Όπως και άλλα συστήματα προγραμματισμού βασισμένα στην προτεραιότητα, έτσι οι διακοπές έχουν το πρόβλημα ότι μια σειρά σημάτων διακοπών υψηλής προτεραιότητας μπορούν να μην επιτρέχουν το χειρισμό ενός σήματος χαμηλής προτεραιότητας. Σε περιπτώσεις που το σύστημα θέλει να δώσει προτεραιότητα σε κάποιες συσκευές αυτό είναι ανεκτό. Σε άλλες περιπτώσεις όμως δεν είναι, και για να λυθεί αυτό το πρόβλημα πολλοί επεξεργαστές παρέχουν και μια μέθοδο σύμφωνα με την οποία η προτεραιότητα μιας διακοπής πέφτει κάθε φορά που χειρίζεται. Ακόμα επιτρέπουν στο λογισμικό να αλλάξει όπως χρειαστεί τις προτεραιότητες των σημάτων διακοπών.

9.4.3.Διαδοχική σταθμοσκόπηση (polling) εναντίον σημάτων διακοπής (interrupts)

Μια εναλλακτική λύση της χρήσης σημάτων διακοπής είναι η διαδοχική σταθμοσκόπηση. Στη διαδοχική σταθμοσκόπηση ο επεξεργαστής περιοδικά ελέγχει κάθε συσκευή Ε/Ε για να δει αν κάποια έχει κάποιο αίτημα προς χειρισμό. Για παράδειγμα, αντί να επιτρέψει σε ένα σήμα από το σκληρό δίσκο να διακόψει τον επεξεργαστή για κάποια δεδομένα που ετοίμασε, το λειτουργικό μπορεί κάθε λίγα χιλιοστά του δευτερόλεπτου να ελέγχει αν τα δεδομένα είναι έτοιμα. Έτσι ο επεξεργαστής μπορεί να ανταποκριθεί σε ασύγχρονα εξωτερικά γεγονότα, χωρίς να χρειάζεται επιπλέον υλικό, όπως γίνεται με τα σήματα διακοπών.
Η χρήση της διαδοχικής σταθμοσκόπησης αντί των σημάτων διακοπών παρέχει πλεονεκτήματα στην επίδοση αν ο επεξεργαστής δεν είναι απασχολημένος με κάποια εργασία καθώς προχωρούν οι εργασίες των συσκευών Ε/Ε. Αν ο επεξεργαστής δεν έχει κάτι άλλο να κάνει μπορεί να εισάγει ένα ερμητικό βρόγχο που θα ελέγχει διαρκώς τις συσκευές Ε/Ε για να δεί αν υπάρχει κάποια εργασία. Αν βρεθεί κάποια ο επεξεργαστής προχωρά στην διαδικασία χειρισμού της εργασίας. Σε αντίθεση, το να ανταποκριθεί σε κάποιο σήμα διακοπής απαιτεί την αλλαγή περιεχομένων, ώστε να σωθούν τα δεδομένα του τρέχοντος προγράμματος για να μπορέσουν να ανακτηθούν με το τέλος της διακοπής, και έτσι έχουμε μια καθυστέρηση στην έναρξη χειρισμού της διακοπής.

Παρόλα αυτά η διαδοχική σταθμοσκόπηση έχει δύο σημαντικά μειονεκτήματα που κάνουν τα σήματα διακοπής τέλεια επιλογή για περιπτώσεις που ο χρόνος ανταπόκρισης του υπολογιστή δεν είναι αποφασιστικής σημασίας. Καταρχήν η σταθμοσκόπηση καταναλώνει εκτελεστικούς πόρους ακόμα και όταν δεν υπάρχουν αιτήσεις προς χειρισμό, γιατί ο επεξεργαστής πρέπει διαρκώς να ελέγχει τις συσκευές Ε/Ε. Η μέση καθυστέρηση πριν την ανταπόκριση σε κάποιο γεγονός βασίζεται στη συχνότητα σταθμοσκόπησης, γιαυτό ο επεξεργαστής πρέπει να ελέγχει συχνά για να εξασφαλίσει ανεκτό χρόνο ανταπόκρισης. Αυτή η διαδικασία καταναλώνει σημαντικό μέρος των κύκλων του επεξεργαστή.

Το δεύτερο μειονέκτημα της διαδοχικής σταθμοσκόπησης είναι ότι τα συστήματα που χρησιμοποιούν αυτή τη μέθοδο απαιτούν το πρόγραμμα του τρέχει στον επεξεργαστή είτε είναι το λειτουργικό , είτε είναι πρόγραμμα χρήστη, να προγραμματίζει τη σταθμοσκόπηση. Αυτό σημαίνει ότι ο προγραμματιστής της εφαρμογής θα πρέπει να γνωρίζει το πόσο συχνά το σύστημα πρέπει να σταθμοσκοπεί και να γράφει προγράμματα που θα σταθμοσκοπούν με τη συγκεκριμένη συχνότητα, αλλιώς το λειτουργικό σύστημα θα πρέπει να διακόψει την εκτέλεση του προγράμματος για να ελέγξει τις Ε/Ε συσκευές. Αυτή η δεύτερη επιλογή απλοποιεί τη διαδικασία προγραμματισμού των διεργασιών για το σύστημα αλλά θέτει την απορία πως το λειτουργικό θα ξέρει ότι η ο χρόνος έχει λήξει και πρέπει να ξαναπρογραμματίσει τη σταθμοσκόπηση. Αν το πρόγραμμα χρήστη δεν γνωρίζει πώς να «γυρίζει» τον επεξεργαστή στο λειτουργικό συχνά, είναι δύσκολο για το λειτουργικό να ξέρει κάθε πότε να σταθμοσκοπεί και πότε να εκτελεί την αλλαγή περιεχομένων του επεξεργαστή, αφού δεν υπάρχουν σήματα διακοπών.

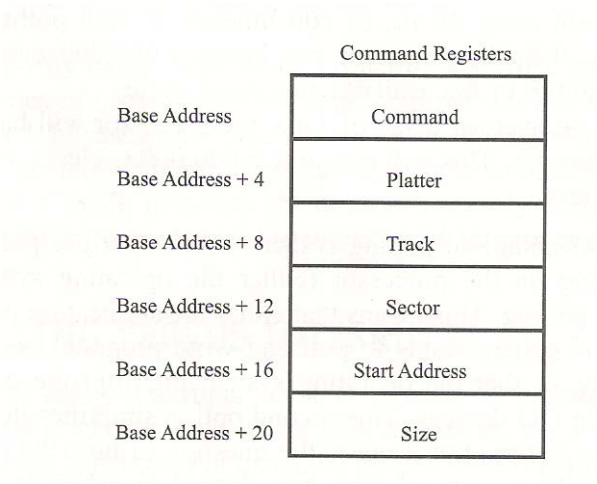
Αυτά τα δύο μειονεκτήματα, η επιβάρυνση και η πολυπλοκότητα του προγραμματισμού κάνουν τα σήματα διακοπών καλύτερη λύση από τη διαδοχική σταθμοσκόπηση για το χειρισμό ασύγχρονων σημάτων στα περισσότερα συστήματα.Στην πραγματικότητα τα περισσότερα λειτουργικά συστήματα υλοποιούν μικροπρογραμματισμό πάνω στη χρήση χρονοδιακόπτη σημάτων διακοπής, που περιοδικά θα ειδοποιεί το λειτουργικό σύστημα να αλλάξει τα περιεχόμενα του επεξεργαστή και να επιτρέψει σε άλλο πρόγραμμα να εκτελεστεί. Η σταθμοσκόπηση γενικά χρησιμοποιείται όταν μόνο ένα πρόγραμμα τρέχει στο σύστημα ή όταν είναι αποφασιστικής σημασίας να ανταποκριθεί το σύστημα σε μια συσκευή Ε/Ε όσο το δυνατόν πιο γρήγορα.

9.5 Είσοδος/ έξοδος με απεικόνιση μνήμης (memory mapped I/O)

Για να χρησιμοποιήσει το σύστημα Ε/Ε ο επεξεργαστής πρέπει να μπορεί να στέλνει εντολές στις συσκευές Ε/Ε και να διαβάζει τα δεδομένα τους. Για να το κάνουν αυτό τα περισσότερα συστήματα χρησιμοποιούν ένα μηχανισμό που ονομάζεται είσοδος / έξοδος με απεικόνιση μνήμης (memory mapped I/O). Στην είσοδο/έξοδο με απεικόνιση μνήμης οι καταχωρητές εντολών (που λέγονται και καταχωρητές ελέγχου) κάθε συσκευής Ε/Ε εμφανίζονται στον προγραμματιστή ως θέσεις μνήμης. Όταν ένα πρόγραμμα γράφει σε αυτές τις θέσεις μνήμης ή τις διαβάζει το υλικό μεταμορφώνει τη λειτουργία μνήμης σε συναλλαγή πάνω στον δίαυλο Ε/Ε που διαβάζει η γράφει τον κατάλληλο καταχωρητή της συσκευής. Σε περίπτωση ανάγνωσης το αποτέλεσμα της εντολής μεταφέρεται στον επεξεργαστή μέσω του διαύλου Ε/Ε και αποθηκεύεται στον καταχωρητή προορισμού της φόρτωσης.

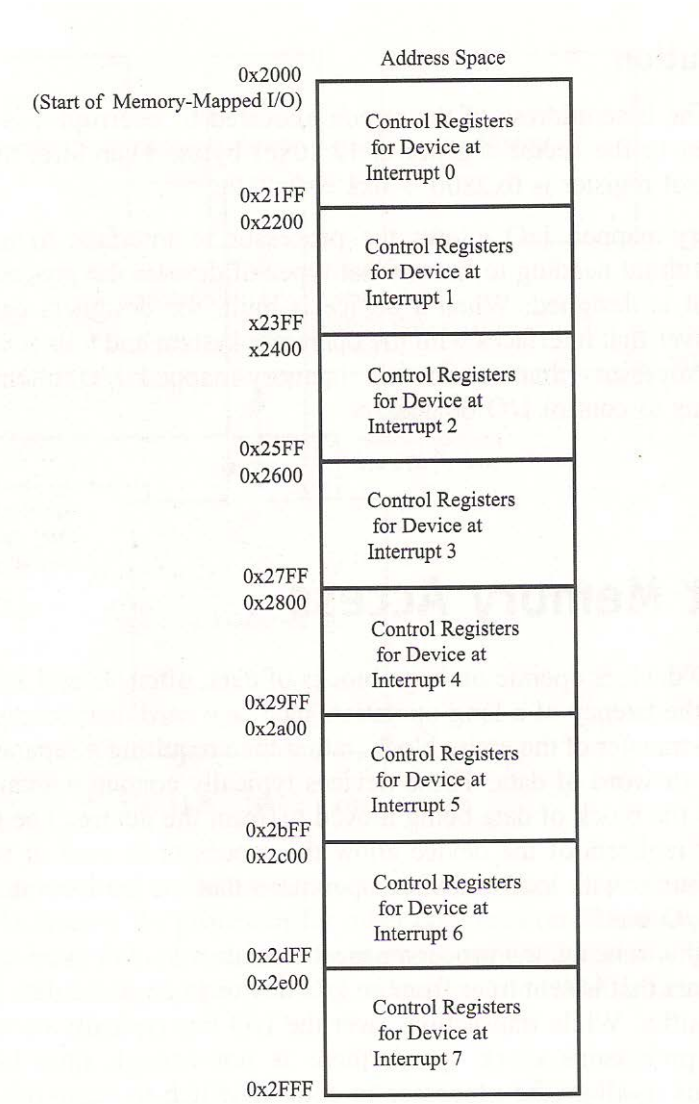
Το σχήμα 9-3 απεικονίζει πώς ο καταχωρητής εντολών ενός σκληρού δίσκου που χρησιμοποιεί 32bit καταχωρητές εντολών μπορεί διαχειριστεί από τη μνήμη απεικόνισης Ε/Ε. Ο καταχωρητής εντολών χρησιμοποιείται για να ενημερώνει τη συσκευή για το τι θέλει να κάνει ο επεξεργαστής. Οι platter, truck and sector καταχωρητές κωδικοποιούν τη θέση των δεδομένων που θέλει να διαβάσει ή να γράψει ο επεξεργαστής (Αυτοί οι όροι θα διευκρινιστούν στο 9.8). Το αρχικό πεδίο διεύθυνσης λέει στο δίσκο από ποιο σημείο της κύριας μνήμης πρέπει να διαβαστούν (σε περίπτωση εγγραφής) ή να γραφούν (σε περίπτωση ανάγνωσης) τα δεδομένα που ζητήθηκαν. Το πεδίο διαστάσεων λέει στο δίσκο πόση μνήμη να μεταφέρει. Αυτό΄θ οι

καταχωρητές χρησιμοποιούνται για DMA μεταφορές μήμνης για τις οποίες θα συζητήσουμε στο 9.6.



ΣΧΗΜΑ9-3 memory mapped I/O

Λόγω της ποικιλίας των συσκευών που μπορούν να συνδεθούν σε ένα δίαυλο Ε/Ε η αρχιτεκτονική καθορίζει ένα μπλοκ διευθύνσεων μνήμης που περιέχουν τους καταχωρητές εντολών κάθε πιθανής συσκευής Ε/Ε, παρόλο που η διευθέτηση των μπλοκ αυτών αλλάζει από επεξεργαστή σε επεξεργαστή. Το σχήμα 11-4 δείχνει ένα παράδειγμα, πώς αυτά τα μπλοκ μνήμης μπορούν να σχεδιαστούν σε έναν επεξεργαστή που συσχετίζει την περιοχή της μνήμης με την συσκευή που συνδέεται σε κάθε γραμμή σήματος διακοπής. Για το συγκεκριμένο παράδειγμα η αρχιτεκτονική έχει καθορίσει 512 bytes καταχωρητών ελέγχου για κάθε συσκευή, όχι γιατί οι περισσότερες συσκευές απαιτούν τόσους πολλούς καταχωρητές, αλλά για να διασφαλιστεί ότι δε θα υπάρχουν δυσκολίες από συσκευές που χρειάζονται περισσότερο χώρο για καταχωρητές ελέγχου από όσους επιτρέπει η αρχιτεκτονική.



ΣΧΗΜΑ9-4 sample memory map

Στο σχήμα 9-3 καθορίσαμε κάθε καταχωρητή ελέγχου στον οδηγό δίσκου με βάση τη μετατόπιση από τη βασική διεύθυνση της περιοχής της μνήμης που είναι διανεμημένη ση συσκευή. Αν ένας τέτοιος δίσκος ήταν συνδεδεμένος σε ένα σήμα διακοπών στο σύστημα που απεικονίστηκε στο σχήμα 9-4 η διεύθυνση κάθε καταχωρητή ελέγχου θα έπρεπε να υπολογιστεί προσθέτοντας τη μετατόπιση του καταχωρητή στην χαμηλότερη διεύθυνση της περιοχής της μνήμης που είναι διανεμημένη στο σήμα διακοπής του δίσκου .

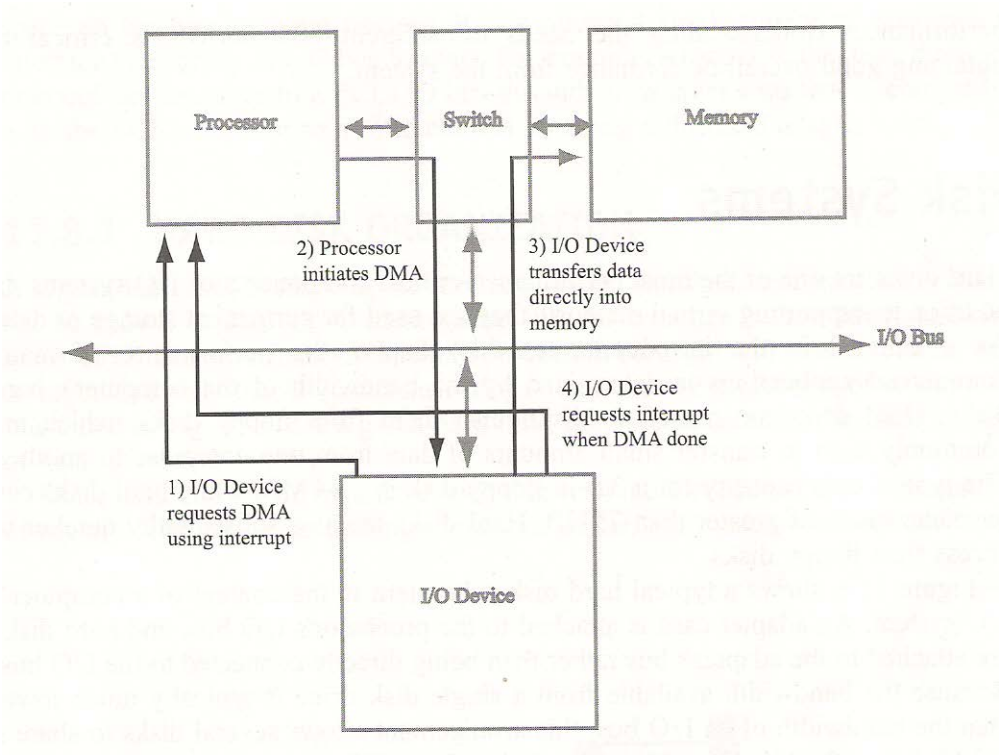
Η απεικόνιση μνήμης Ε/Ε επιτρέπει στον επεξεργαστή να αλληλεπιδρά με μια μεγάλη ποικιλία συσκευών χωρίς να χρειάζεται να γνωρίζουμε από τη στιγμή σχεδίασής του. Όταν μια συσκευή κατασκευάζεται οι σχεδιαστές απλά γράφουν ένα οδηγό συσκευής (device driver) που αλληλεπιδρά με το λειτουργικό σύστημα και του λέει πώς να ελέγξει τη συσκευή. Οι επεξεργαστές που δεν έχουν τέτοια μνήμη συχνά βασίζονται σε ειδικές εντολές για να ελέγξουν τις Ε/Ε συσκευές.

9.6 Άμεση προσπέλαση στη μνήμη

Πολλές συσκευές Ε/Ε λειτουργούν σε μεγάλα μπλοκ μνήμης, που συχνά έχουν μήκος αρκετά kilobytes, και επιτρέπουν την κατανομή της καθυστέρησης κάποιας χρονοβόρας λειτουργίας (όπως την προσπέλαση του σκληρού δίσκου) σε όλο το μπλοκ , αντί να απαιτούν προσπέλαση της κάθε λέξης δεδομένων του δίσκου ξεχωριστά. Αυτές οι συσκευές τυπικά περιέχουν μια μικρή ενδιάμεση μνήμη (buffer) που κρατάει το μπλοκ δεδομένων που μεταφέρεται προς και από τη συσκευή. Οι καταχωρητές εντολών της μνήμης απεικόνισης επιτρέπουν στον επεξεργαστή να διαβάσει ή να γράψει λέξεις από ή στην ενδιάμεση μνήμη με εντολές φόρτωσης και αποθήκευσης , τις οποίες το υλικό μετατρέπει σε αιτήσεις πάνω στο δίαυλο Ε/Ε.

Χρησιμοποιώντας αυτή τη μέθοδο, ο επεξεργαστής πρέπει να εκτελέσει λειτουργίες φόρτωσης ή αποθήκευσης για κάθε λέξεις δεδομένων που στέλνεται από και προς μια συσκευή Ε/Ε και να αποθηκεύσει τα αποτελέσματα στην ενδιάμεση μνήμη της συσκευής. Ενώ οι συναλλαγές πάνω στο δίαυλο Ε/Ε έχουν συνήθως χαμηλότερη ταχύτητα από τον κύκλο ρολογιού του επεξεργαστή, δεν υπάρχει αρκετός χρόνος μεταξύ των συναλλαγών του διαύλου ώστε ο επεξεργαστής να εκτελέσει αλλαγή περιεχομένου, που σημαίνει ότι ο επεξεργαστής θα είναι απασχολημένος κάθε φορά που τα δεδομένα θα μεταφέρονται πάνων στο δίαυλο Ε/Ε. Καθώς η εκμετάλλευση του διαύλου πλησιάζει το 100%, ο επεξεργαστής θα απασχολείται με τη μεταφορά των δεδομένων και έτσι κανένας από τους δύο δε θα εκτελεί υπολογισμούς.
Τα συστήματα άμεσης προσπέλασης στη μνήμη (direct memory access-DMA) αναπτύχθηκαν για να αντιμετωπίσουν ακριβώς αυτό το πρόβλημα. Σε ένα σύστημα DMA οι συσκευές Ε/Ε μπορούν να έχουν πρόσβαση στη μνήμη χωρίς την παρεμβολή του επεξεργαστή. Στο σχήμα 9-5 βλέπουμε μια σειρά γεγονότων που απαιτούνται από μια DMA μεταφορά για να αντιγραφούν τα αποτελέσματα μιας λειτουργίας Ε/Ε στην κύρια μνήμη.

Για να ξεκινήσει η DMA ακολουθία, η συσκευή Ε/Ε στέλνει σήμα διακοπής και ζητά την προσοχή του επεξεργαστή. Ο επεξεργαστής ανταποκρίνεται ελέγχοντας την κατάσταση της συσκευής μέσω των καταχωρητών της άμεσης απεικόνισης μνήμης και στέλνει μια εντολή στη συσκευή , να κάνει μια DMA μεταφορά των δεδομένων στην κύρια μνήμη. Από τη στιγμή που η εντολή αυτή αρχίζει να εκτελείται, ο επεξεργαστής ξεκινά κάποια άλλη εργασία ταυτόχρονα με τη μεταφορά δεδομένων που κάνει η συσκευή Ε/Ε. Όταν η μεταφορά DMA ολοκληρωθεί, η συσκευή Ε/Ε στέλνει άλλο ένα σήμα διακοπής για να ειδοποιήσει τον επεξεργαστή ότι πλέον μπορεί να προσπελάσει τα δεδομένα.



ΣΧΗΜΑ9-5 direct memory access

Η χρήση DMA μεταφορών μπορεί να μειώσει σημαντικά τον αριθμό των κύκλων του επεξεργαστή που ξοδεύονται για το χειρισμό Ε/Ε, και έτσι μπορούν να γίνουν και άλλοι υπολογισμοί. Παρόλα αυτά Η συσκευή Ε/Ε και ο επεξεργαστής μοιράζονται το εύρος ζώνης της μνήμης, που σημαίνει ότι κατά την DMA λειτουργία μειώνεται το εύρος της μνήμης που θα μπορούσε να χρησιμοποιηθεί από κάποιο πρόγραμμα.

9.7. Συσκευές Ε/Ε

Υπάρχει ένας τεράστιος αριθμός διαφορετικών συσκευών Ε/Ε για τα σύγχρονα υπολογιστικά συστήματα, που μπορεί να χωριστεί σε τρεις κατηγορίες: τις συσκευές που δέχονται εισαγωγή δεδομένων από τον άνθρωπο, τις συσκευές που δείχνουν τα δεδομένα αυτά στον άνθρωπο, και τις συσκευές που αλληλεπιδρούν με άλλα μηχανήματα. Οι συσκευές που δέχονται δεδομένα από τον άνθρωπο, όπως τα πληκτρολόγια και τα ποντίκια τείνουν να έχουν μικρές απαιτήσεις σε εύρος ζώνης χρόνο ανταπόκρισης συμβόλου προτροπής (promt response time). Για παράδειγμα ένας δακτυλογράφος που γράφει 60 λέξεις το λεπτό, δημιουργεί 5 χαρακτήρες το δευτερόλεπτο, υποθέτοντας ότι το μέσο μήκος λέξης θα είναι 5 γράμματα. Το εύρος ζώνης της συσκευής που χρησιμοποιείται είναι ασήμαντο, αλλά ο δακτυλογράφος μπορεί να παρατηρήσει κάποια καθυστέρηση , αν ο υπολογιστής δεν ανταποκριθεί άμεσα στο πάτημα του πλήκτρου.

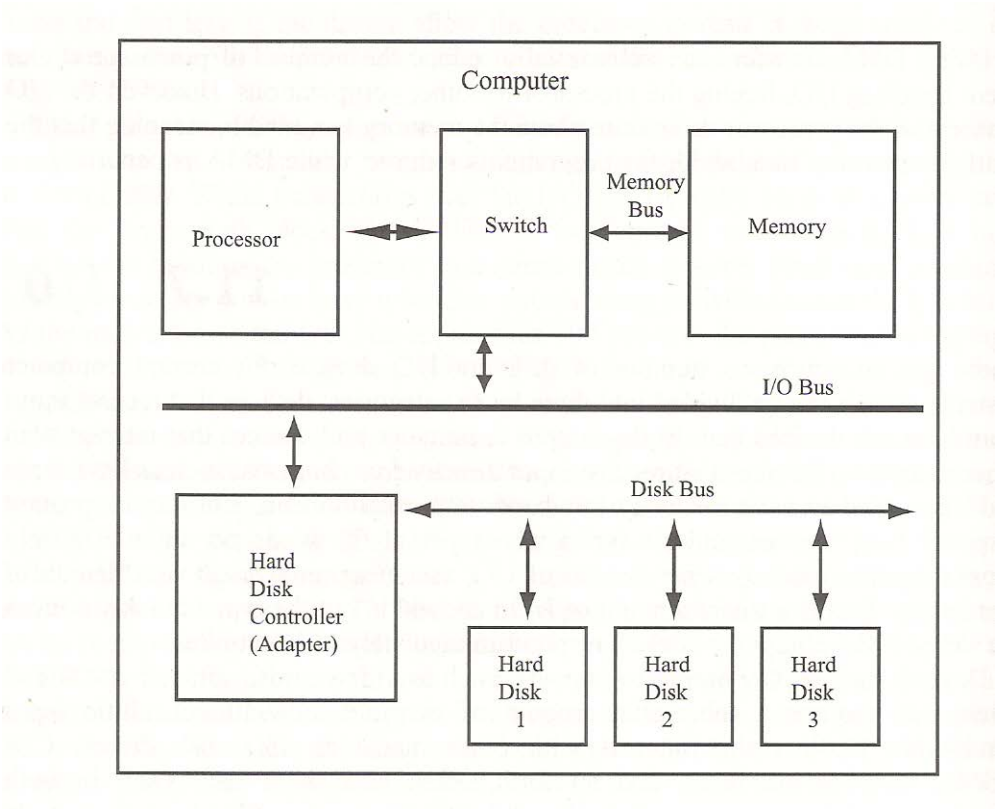
Οι συσκευές που δείχνουν δεδομένα στους ανθρώπους, όπως οι κάρτες βίντεο, οι κάρτες ήχου και οι εκτυπωτές μπορούν να διεκδικήσουν σημαντικό ποσοστό του εύρους ζώνης όσον αφορά την έξοδο, δε συμβαίνει όμως το ίσο με το εύρος εισόδου. Οι συσκευές που αλληλεπιδρούν με άλλα μηχανήματα όπως οι οδηγοί δίσκων και cd, οι διεπαφές δικτύου κτλ., συχνά απαιτούν υψηλό εύρος ζώνης και στις λειτουργίες

εισόδου και στις λειτουργίες εξόδου, αλλά μικρό χρόνο ανταπόκρισης από τον επεξεργαστή , για να φτάσουν στο μέγιστο της απόδοσης τους. Η κατανόηση των διαφορετικών συσκευών Ε/Ε είναι πολύ σημαντική για τη συνολική καλή επίδοση του συστήματος.

9.8. Συστήματα δίσκων

Οι σκληροί δίσκοι είναι μια από τις συσκευές του συστήματος Ε/Ε που επηρεάζει σημαντικά την απόδοσή του. Εκτός του ότι υποστηρίζουν την εικονική μνήμη, χρησιμοποιούντα για μόνιμη αποθήκευση δεδομένων. Όπως αναφέρθηκε στην εισαγωγή αυτού του κεφαλαίου, η απόδοση πολλών εμπορικών εφαρμογών εξαρτάται από το εύρος ζώνης του σκληρού δίσκου του υπολογιστή. Οι σκληροί δίσκοι ονομάστηκαν έτσι για να διακρίνονται από τις δισκέτες(εύκαμπτοι δίσκοι) που χρησιμοποιούνται ευρύτατα για τη μεταφορά μικρών ποσοτήτων δεδομένων από τον ένα υπολογιστή στον άλλο. Η τυπική χωρητικότητα για μια δισκέτα 3.5 ιντσών είναι 1.44MB ενώ οι σκληροί δίσκοι μπορούν να φτάσουν και να ξεπεράσουν τα 75GB.Επίσης οι σκληροί δίσκοι είναι αρκετά γρηγορότεροι από τις δισκέτες όσον αφορά την προσπέλαση τους.

Το σχήμα 9-6 δείχνει ένα τυπικό υποσύστημα σκληρού δίσκου που περιλαμβάνεται στο σύστημα Ε/Ε. Μια κάρτα προσαρμογής συνδέεται στο δίαυλο Ε/Ε και πάνω σε αυτή συνδέεται ο σκληρός δίσκος, αντί να συνδεθεί κατευθείαν στο δίαυλο. Επειδή το εύρος ζώνης ενός οδηγού δίσκου είναι συνήθως πολύ μικρότερο από το εύρος του διαύλου, γι αυτό και η διευθέτηση αυτή επιτρέπει τη σύνδεση πολλών δίσκων στην ίδια υποδοχή του διαύλου, χωρίς να περιορίζεται η δυνατότητά τους να μεταφέρουν δεδομένα και ελευθερώνοντας χώρο που μπορεί να χρησιμοποιηθεί από άλλες συσκευές. Επίσης επιτρέπει σε ένα μοντέλο σκληρού δίσκου να αλληλεπιδρά με μια ποικιλία διαύλων Ε/Ε. Για παράδειγμα ένας σκληρός δίσκος που χρησιμοποιεί προτόκολλο SCSI μπορεί να επικοινωνήσει με ένα PCI δίαυλο, μέσω μις κάρτας προσαρμογής που είναι συμβατή με το πρότυπο PCI. Το ίδιο μπορεί να συμβεί με οποιοδήποτε τύπο διαύλου και την κατάλληλη κάρτα προσαρμογής.



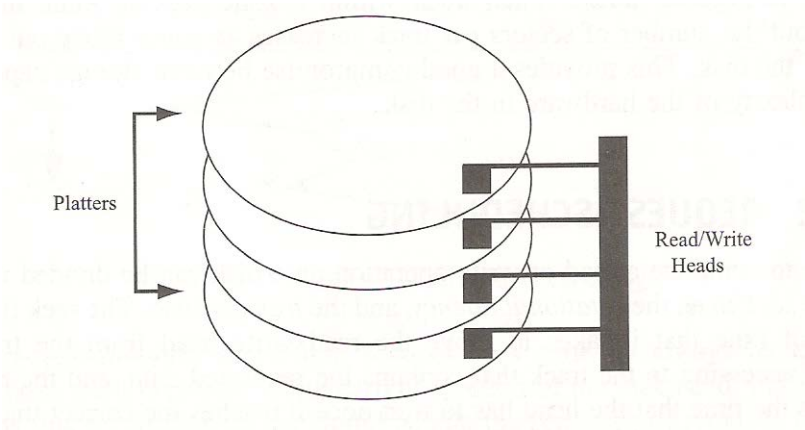
ΣΧΗΜΑ9-6 disc subsystem

9.8.1. ΟΡΓΑΝΩΣΗ ΣΚΛΗΡΩΝ ΔΙΣΚΩΝ

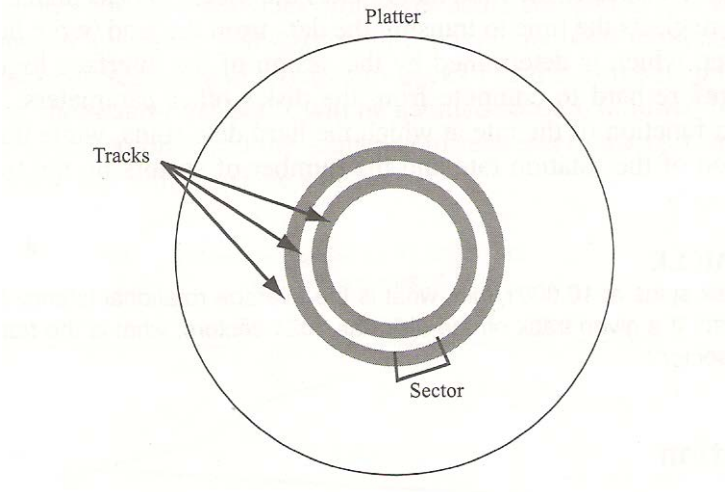
Ένας τυπικός σκληρός δίσκος αποτελείται από πολλούς platters , επίπεδους δίσκους όπου αποθηκεύονται δεδομένα όπως φαίνεται στο σχήμα 9-7. Κάθε δίσκος έχει τη δική του κεφαλή (ανάγνωσης, εγγραφής), επιτρέποντας στα δεδομένα που αποθηκεύτηκαν σε διαφορετικούς δίσκους να προσπελαστούν παράλληλα. Μέσα σε κάθε δίσκο τα δεδομένα οργανώνονται σε ομόκεντρες τροχιές (tracks) και τομείς (sectors) όπως φαίνεται στο σχήμα 9-8. Σε κάθε σκληρό δίσκο κάθε τομέας περιέχει την ίδια ποσότητα πληροφοριών, συχνά 512 bytes.

Οι παλαιότεροι σκληροί δίσκοι ήταν οργανωμένοι με τέτοιο τρόπο, ώστε κάθε τροχιά να περιέχει τον ίδιο αριθμό τομέων. Αυτό έκανε πιο εύκολο τον έλεγχο της συσκευής, αφού κάθε τροχιά περιείχε την ίδια ποσότητα δεδομένων, όμως ως αποτέλεσμα ο δίσκος χωρούσε πολύ λιγότερα δεδομένα από αυτά που μπορούσε, γιατί ο σε κάθε δίσκο μπορούν να αποθηκευτούν τόσα δεδομένα όσα επιτρέπει ο αριθμός των bits που η κεφαλή μπορεί να διαβάσει ή να γράψει ανά ίντσα. Σε δίσκους που έχουν σταθερό αριθμό τομέων σε κάθε τροχιά, ο αριθμός των τομέων ανά τροχιά καθορίζεται από τον αριθμό των bits που χωρούν στον στην πιο μικρή τροχιά, που είναι συνάρτηση της περιφέρειας της πιο μικρής τροχιά και της τεχνολογίας κατασκευής ου δίσκου. Οι υπόλοιπες τροχιές χωρούν τον ίδιο αριθμό bits αλλά γράφονται πιο αραιά, έτσι ώστε κάθε τομέας να έχει το ίδιο κλάσμα με την περιφέρεια της πιο μικρής τροχιάς. Με αυτό τον τρόπο χάνεται ένας σημαντικός

αποθηκευτικός χώρος, ιδιαίτερα προς τις εξωτερικές τροχιές, γιατί τα bits εκεί γράφονται πολύ πιο αραιά απ' ότι στις εσωτερικές.



ΣΧΗΜΑ9-7 disc organization



ΣΧΗΜΑ9-8 tracks and sectors

Οι πιο νέοι δίσκοι αποθηκεύουν δεδομένα ,πιο πυκνά κρατώντας την πυκνότητα με την οποία τα bits γράφονται σταθερή αλλά αλλάζουν τον αριθμό των τομέων σε κάθε τροχιά, έτσι ώστε οι τομείς μακριά από το κέντρο να περιέχουν περισσότερες πληροφορίες. Θεωρητικά κάθε τροχιά περιέχει διαφορετικό αριθμό τομέων, έτσι ώστε να μεγιστοποιείται η ποσότητα δεδομένων που μπορεί να αποθηκευτεί. Στην πράξη, οι κατασκευαστές δίσκων τους χωρίζουν σε αρκετές ζώνες και ομόκεντρες τροχιές. Κάθε τροχιά σε κάθε ζώνη έχει τον ίδιο αριθμό τομέων, αλλά ο αριθμός των τομέων σε κάθε ζώνη αυξάνεται καθώς οι ζώνες προχωρούν προς το εξωτερικό του δίσκου. Έτσι επιτυγχάνεται ένας καλός συμβιβασμός ανάμεσα στη χωρητικότητα αποθήκευσης και την περιπλοκότητα του υλικού του δίσκου.

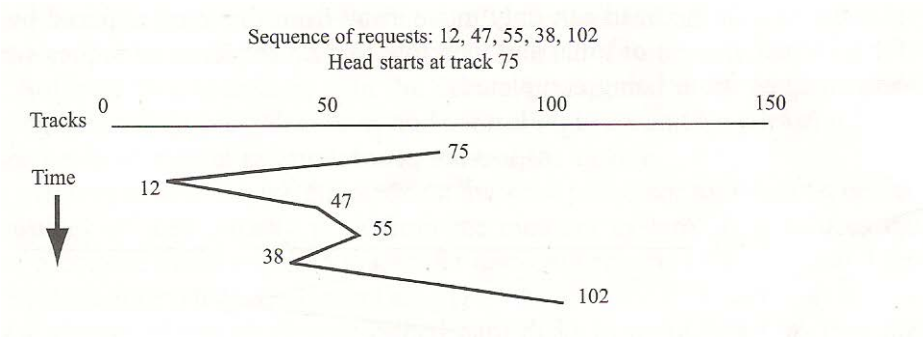
9.8.2.Χρονοδρομολόγηση αιτήσεων.

Ο χρόνος ολοκλήρωσης μιας ανάγνωσης ή μιας εγγραφής μπορεί να χωριστεί σε τρία μέρη. Στο χρόνο αναζήτησης (seek time), καθυστέρηση περιστροφών (rotational latency) και στο χρόνο μεταφοράς (transfer time). Ο χρόνος αναζήτησης είναι ο χρόνος που απαιτείται για τη μετακίνηση της κεφαλής από την τροχιά όπου βρίσκεται στην τροχιά όπου βρίσκονται τα αιτούμενα δεδομένα, ενώ η καθυστέρηση περιστροφών είναι ο χρόνος που περιμένει η κεφαλή από τη στιγμή που βρήκε τη σωστή τροχιά, μέχρι να βρεθεί ο κατάλληλος τομέας. Ο χρόνος μεταφοράς είναι ο χρόνος που χρειάζεται για γίνει η ανάγνωση ή η εγγραφή, που ουσιαστικά είναι ο χρόνος που κάνει ο τομέας να περάσει κάτω από την κεφαλή. Αυτός ο χρόνος δεν περιλαμβάνει το χρόνο μετάδοσης των δεδομένων από την κεφαλή στην κάρτα προσαρμογής, που εξαρτάται από το σχεδιασμό της λογικής διεπαφής του δίσκου και άρα είναι δύσκολο να υπολογιστεί. Η καθυστέρηση περιστροφών είναι συνάρτηση του ρυθμού περιστροφής του σκληρού δίσκου ενώ ο χρόνος μεταφοράς είναι συνάρτηση του ρυθμού περιστροφής και του αριθμού των τομέων της τροχιάς.

Παράδειγμα
Αν ένας δίσκος περιστρέφεται με 10,000 r/min ποια είναι η μέση καθυστέρηση περιστροφής μιας αίτησης; Αν μια δεδομένη τροχιά του δίσκου έχει 1024 τομείς ποιος είναι ο χρόνος μεταφοράς του τομέα;

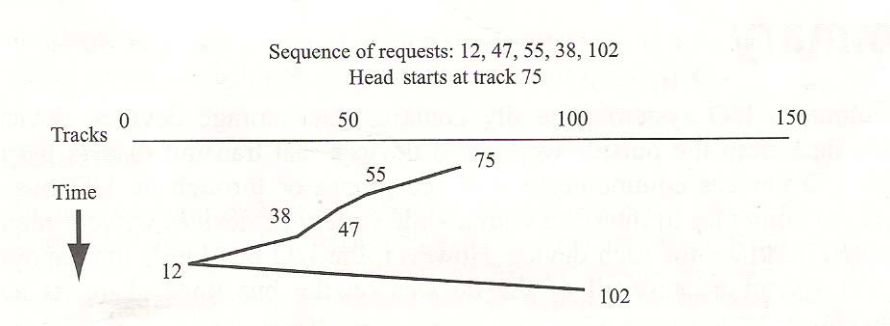
Λύση
Στους 10,000 r/min χρειάζονται 6ms για την ολοκλήρωση μια πλήρους περιστροφής του δίσκου. Κατά μέσο όρο η κεφαλή θα πρέπει να περιμένει μισή περιστροφή μέχρι να βρει τον κατάλληλο τομέα, άρα η περιστροφή καθυστέρησης είναι 3ms. Αφού υπάρχουν 1024 τομείς ο χρόνος μεταφοράς θα είναι ίσος με το χρόνο περιστροφής του δίσκου διαιρεμένο με το 1024, δηλαδή περίπου 6ms.

Το λειτουργικό σύστημα ή το υλικό μπορεί να επηρεάσει την απόδοση του σκληρού δίσκου, επιλέγοντας τον τρόπο με τον οποίο χειρίζεται τις αιτήσεις, αν υπάρχουν πολλές και αξιοσημείωτες αιτήσεις. Υπάρχουν τρεις πολιτικές. Η FCFS(first come first serve), η SSTF (shortest seek time first) και ο προγραμματισμός LOOK.Στην FCFS ο δίσκος χειρίζεται τις αιτήσεις με τη σειρά που έχουν γίνει, όπως βλέπουμε και στο σχήμα 9-9. Αυτή η πολιτική έχει το πλεονέκτημα ότι είναι εύκολο να υλοποιηθεί, αλλά μπορεί να χρειαστεί περισσότερη κίνηση του δίσκου απ ότι οι υπόλοιπες. Αφού ο χρόνος ανίχνευσης μιας αίτησης είναι ανάλογος με τον αριθμό των τροχιών που καλείται να καλύψει η κεφαλή, αυξάνεται η κίνηση του δίσκου και άρα έχουμε περισσότερο χρόνο ανίχνευσης και χαμηλότερη επίδοση.



ΣΧΗΜΑ9-9 first-come-first-serve disc scheduling

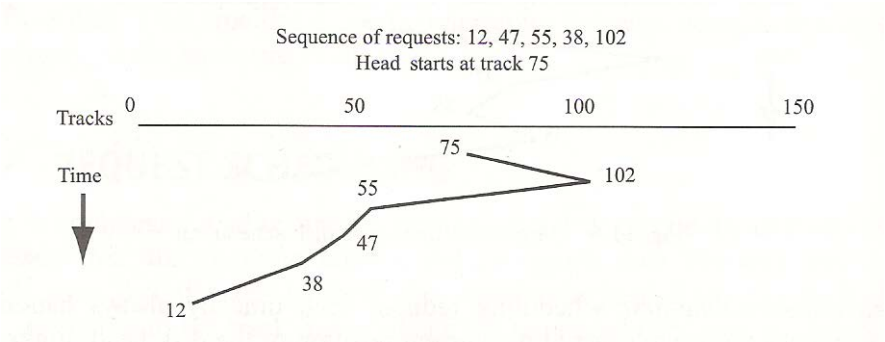
Η SSTF πολιτική μειώνει το χρόνο ανίχνευσης αφού πάντα χειρίζεται την αίτηση που βρίσκεται πιο κοντά στην τρέχουσα θέση της κεφαλής. Το σχήμα 9-10 δείχνει τον τρόπο με τον οποίο αυτή η πολιτική μπορεί να χειριστεί μια σειρά αιτήσεων που είδαμε και στο 9-9 σχήμα. Η SSTF μπορεί να μειώσει σημαντικά το χρόνο ανίχνευσης αλλά έχει το μειονέκτημα ότι μια σειρά αιτήσεων από κοντινές θέσεις μπορούν να παρεμποδίσουν την εκτέλεση κάποιας αίτησης που βρίσκεται μακρύτερα (starvation-ανεπάρκεια). Για παράδειγμα θεωρείστε ένα πρόγραμμα που ξεκινά να κάνει αιτήσεις στους τομείς 1 και 2. Η αίτηση στη θέση ένα χειρίζεται πρώτη έπειτα η 2 και μετά πάλι η 1 κτλ. Μέχρι η ακολουθία μεταξύ των 1 και 2 τελειώσει η 100 θα βρίσκεται σε αναμονή.



ΣΧΗΜΑ9-10 shortest-seek-time-first scheduling

Η πολιτική LOOK (που λέγεται επίσης και elevator scheduling) είναι συμβιβαστική ανάμεσα στις δύο προηγούμενες.. Επιτυγχάνει καλύτερες επιδόσεις από την FCFS χωρίς την πιθανότητα της ανεπάρκειας. Στη LOOK πολιτική η κεφαλή κινείται είτε προς το εσωτερικό είτε προς το εξωτερικό του δίσκου, ικανοποιώντας όλες τις αιτήσεις των τροχιών από τις οποίες περνάει. Όταν φτάσει στην τελευταία, προς τα έξω ή προς τα μέσα αλλάζει φορά και κατεύθυνση και χειρίζεται τις αιτήσεις που βρίσκονται σε αναμονή, καθώς φτάνει στις τροχιές τους. Όταν η κεφαλή ξεκινά να κινείται προς τα έξω, συνεχίζει μέχρι να συναντήσει την μεγαλύτερη τροχιά που έχει ζητηθεί και μετά επιστρέφει προς τα κάτω μέχρι να φτάσει στην κατώτερη τροχιά που έχει ζητηθεί.

Το σχήμα 9-11 μας δείχνει την LOOK πολιτική, με την κεφαλή να ξεκινά κινούμενη προς τα έξω (δηλ προς τις τροχιές με το μεγαλύτερο αριθμό). Ο αλγόριθμος αυτός προλαμβάνει την ανεπάρκεια γιατί διασφαλίζει ότι όταν η κεφαλή αρχίζει να κινείται προς κάποια κατεύθυνση θα συνεχίσει μέχρι η αίτηση αυτή να ικανοποιηθεί. Αφού ο δίσκος έχει όρια για την κίνηση της κεφαλής, είναι εγγυημένο ότι η κεφαλή θα μετακινηθεί από την τροχιά που απαιτείται για λίγη ώρα (πχ για να ικανοποιήσει μια άλλη τροχιά) και έτσι δεν υπάρχει ακολουθία αιτήσεων που μπορεί να εμποδίσει την εκτέλεση της αίτησης.



ΣΧΗΜΑ9-11

Τα τωρινά συστήματα τείνουν να χρησιμοποιούν κυρίως τις πολιτικές SSTF και LOOK, κάποιες φορές με διαφοροποιήσεις των αλγορίθμων που χρησιμοποιούνται. Γενικά η πολιτική LOOK είναι καλύτερη για συστήματα με μεγάλες απαιτήσεις, γιατί με τη SSTF όσο αυξάνεται και ο χρόνος απασχόλησης του δίσκου αυξάνονται και οι πιθανότητες ανεπάρκειας,

ΚΕΦΑΛΑΙΟ 10^{ο}

ΠΟΛΥΕΠΕΞΕΡΓΑΣΤΕΣ

10.1 Στόχοι

Αυτό το κεφάλαιο ολοκληρώνει τη συζήτησή μας για την αρχιτεκτονική και οργάνωση των υπολογιστών με την παροχή μιας εισαγωγής στα συστήματα πολυεπεξεργαστών.

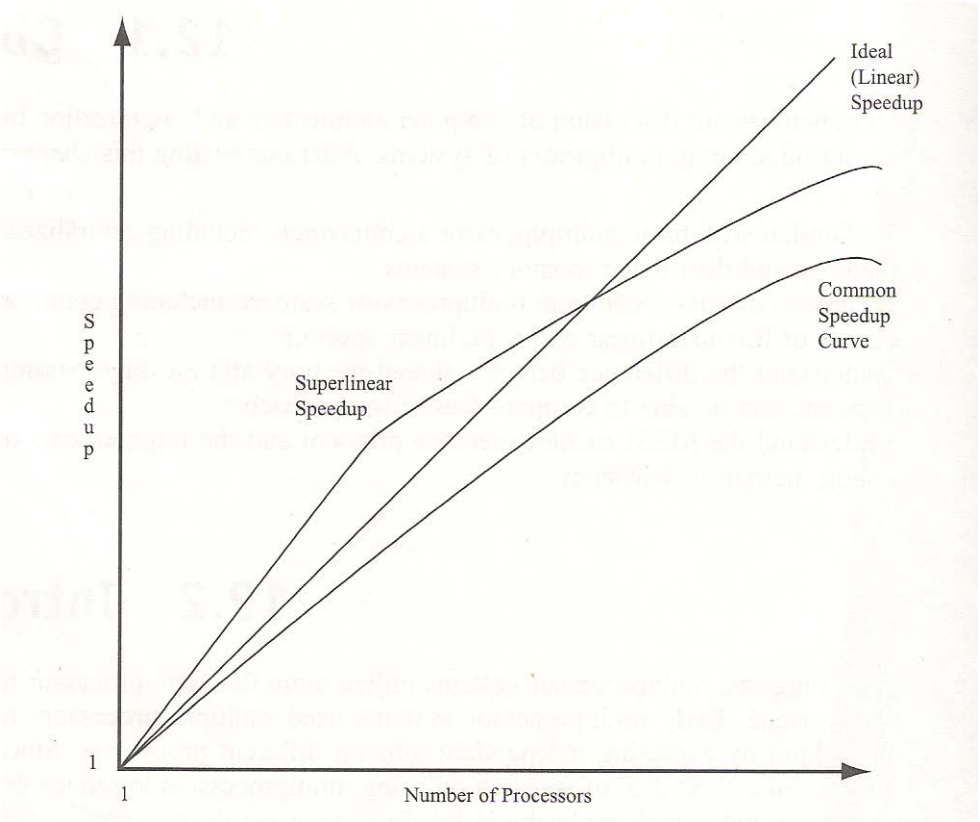
10.2 Εισαγωγή

Όπως το όνομά τους προτείνει, τα συστήματα πολυεπεξεργαστών χρησιμοποιούν περισσότερους από έναν επεξεργαστές για να βελτιώσουν την απόδοση.Τα πρόωρα συστήματα πολυεπεξεργαστών χρησιμοποίησαν τους πολλαπλούς επεξεργαστές για να βελτιώσουν τη ρυθμοαπόδοση με την εκτέλεση των ανεξάρτητων εργασιών στους διαφορετικούς επεξεργαστές.Από τότε, έχουν υπάρξει πολλές έρευνες για τη χρησιμοποίηση των πολυεπεξεργαστών για να μειώσουν τους χρόνους εκτέλεσης των μεμονωμένων εφαρμογών με τη διαίρεση της εργασίας ενός ενιαίου προγράμματος στους πολλαπλούς επεξεργαστές.

Οι πολυεπεξεργαστές είναι ελκυστικοί λόγω των τεχνολογικών και φυσικών περιορισμών στην βελτιώση της απόδοσης των επεξεργαστών οποιαδήποτε στιγμή.Όπως συζητήσαμε, τεχνικές όπως ο παραλληλισμός επιπέδων οδηγίας, κρύφές μνήμες, και διανομή διοχετεύσης μικραίνουν τις βελτιώσεις απόδοσης ως ποσό περιοχής τσιπ που αφιερώνεται σε αυτες τις αυξήσεις, περιοριζοντας τη μέγιστη απόδοση που μπορεί να επιτευχθεί με έναν απλό επεξεργαστή.Με τη διαίρεση της εργασίας ενος απλού προγράμματος μεταξύ των πολλαπλών επεξεργαστών, οι πολυεπεξεργαστές μπορούν να επιτύχουν τη μεγαλύτερη απόδοση από ότι είναι δυνατό με έναν ενιαίο επεξεργαστή σε οποιαδήποτε δεδομένη τεχνολογία επεξεργασίας.

10.3 Επιτάχυνση και απόδοση

Όπως οι σχεδιαστές των συστημάτων μονο-επεξεργαστών,οι αρχιτεκτόνες πολυεπεξεργαστών συχνά μετρούν την αποδοση στη διάρκεια της επιτάχυνσης.Στα πλαίσια των πολυεπεξεργαστών,η επιτάχυνση γενικά αναφέρει πόσο γρηγορότερα ένα πρόγραμμα τρέχει σε ένα σύστημα με *n* επεξεργαστές από ότι κάνει σε ένα σύστημα με έναν επεξεργαστή του ίδιου τύπου.Το σχήμα 10-1 παρουσιάζει τη γραφική παράσταση ενός παραδείγματος επιταχυνσης.Ο κάθετος άξονας αποτυπώνει την επιτάχυνση πάνω σε ένα συστήμα μονο-επεξεργαστή, ενώ ο οριζόντιος άξονας είναι ο αριθμός των επεξεργαστών στο σύστημα.Σημείωσε ότι η αρχή για αυτές τις γραφικές παραστάσεις είναι συχνά (1.1), παρά (0.0), επειδή η επιτάχυνση είναι σχετικά μετρημένη σε μία μηχανή ενος επεξεργστή, όχι σε μηχανή μηδενικου-επεξεργαστή.



ΣΧΗΜΑ10-1 example speedup gragh

10.3.1 Περιορισμοί στην επιτάχυνση

Ένα ιδανικό σύστημα πολυεπεξεργαστών θα έχει γραμμική επιτάχυνση - ο χρόνος εκτέλεσης ενός προγράμματος για ένα σύστημα *n*-επεξεργαστών θα ήταν 1/*n* του χρόνου εκτέλεσης σε ένα σύστημα επεξεργαστή. Αυτό το ποσοστό επιτάχυνσης αντιπροσωπεύεται από την ιδανική (γραμμική) επιταχυνση στο παράδειγμα του διαγράμματος.Στην πράξη, τα συστήματα επιδεικνύουν γενικά τις καμπύλες επιταχυνσης παρόμοιες με την κοινή καμπύλη επιταχυνσης που παρουσιάζεται στη γραφική παράσταση.Όταν ο αριθμός επεξεργαστών είναι μικρός, το σύστημα επιτυγχάνει κοντινή - γραμμική επιταχυνση.Δεδομένου ότι οι αριθμοί επεξεργαστών αυξάνονται, η καμπύλη επιταχυνσης αποκλίνει από το ιδανικό, τελικά ισιώνει ή ακόμα και μειώνεται.

Υπάρχουν πολλοί λόγοι για τους οποίους τα συστήματα επιδεικνύουν less-than-linear επιτάχυνση.

Οι τρεις πιό κοινοί λόγοι είναι οι ακόλουθοι:

1.*Interprocessor Επικοινωνία* - σε ένα σύστημα πολυεπεξεργαστών, όπου ένας επεξεργαστής παράγει (υπολογίζει) μια τιμή που απαιτείται από το μέρος του προγράμματος που τρέχει σε έναν άλλο επεξεργαστή, όπου η τιμή πρέπει να επικοινωνεί με τον επεξεργαστή (ες) που την χρειάζεται, η οποία παίρνει χρόνο. Σε

ένα σύστημα μονο-επεξεργαστη, ολόκληρα τα προγράμματα τρέχουν σε έναν επεξεργαστή, έτσι δεν υπάρχει χρόνος που χάνεται στην interprocessor επικοινωνία .

2.*Συγχρονισμός* - μια άλλη περιπλοκή που εισάγεται από τους πολυεπεξεργαστές είναι ότι είναι συχνά απαραίτητο να να συγχρονίζει τον επεξεργαστή για να εξασφαλίσει ότι όλοι έχουν ολοκληρώσει κάποια φάση του προγράμματος πριν οποιοσδήποτε επεξεργαστής αρχίσει να εργάζεται στην επόμενη φάση του προγράμματος. Παραδείγματος χάριν, τα προγράμματα που μιμούνται τα φυσικά φαινόμενα, όπως η ροή του αέρα πάνω σε ένα αντικείμενο, διαιρούν γενικά το χρόνο σε βήματα σταθερής διάρκειας και απαιτούν όλοι οι επεξεργαστές να έχουν ολοκληρώσει την προσομοίωσή ενός δεδομένου χρονικού βήματος προτού να μπορέσει οποιοσδήποτε επεξεργαστής να προχωρήσει στον επόμενο, έτσι ώστε η προσομοίωση ενός δοσμένου χρονικού βήματος πριν από κάθε επεξεργαστή να μπορεί να βασιστεί στο επόμενο ,έτσι η προσομοίωση του επόμενου χρονικού βήματος να μπορεί να βασιστεί στα αποτελέσματα της προσομοίωσης του τρέχοντος βήματος . Αυτός ο συγχρονισμός απαιτεί interprocessor επικοινωνία, που εισάγει το πλεονάζων χώρο που δεν βρίσκεται στα συστήματα μονο-επεξεργαστη.

3. *Εξισορρόπηση φορτίου*- σε πολλές παράλληλες εφαρμογές, είναι δύσκολο να διαιρεθεί το πρόγραμμα δια μέσου των επεξεργαστών έτσι ώστε καθε μπλόκ εργασίας του επεξεργαστη να παιρνει το ίδιο χρονικό διάστημα.Όταν αυτό δεν είναι δυνατό, μερικοί από τους επεξεργαστές ολοκληρώνουν τους στόχους τους νωρίς και έπειτα περιμένει τους άλλους να τελειώσουν.Αυτή η ανώμαλη διανομή των στόχων δια μέσου των επεξεργαστών αυξάνει το γενικό χρόνο εκτέλεσης, επειδή όλοι οι επεξεργαστές δεν είναι σε πάντα χρήση.

Γενικά, όσο μεγαλύτερο το χρονικό διάστημα που απαιτείται για την επικοινωνία μεταξύ των επεξεργαστών, τόσο χαμηλότερη είναι επιτάχυνση με την οποία τα προγράμματα τρέχουν για το σύστημα.Η λανθάνουσα κατάσταση interprocessor επικοινωνίας επεξεργαστών έχει επιπτώσεις προφανώς στο χρονικό διάστημα που απαιτείται για να επικοινωνήσουν τα στοιχεία μεταξύ των μερών του προγράμματος που τρέχει για κάθε επεξεργαστή.Έχει επιπτώσεις επίσης στο χρονικό διάστημα που απαιτείται για το συγχρονισμό, επειδή ο συγχρονισμός είναι χαρακτηριστικά εφαρμοσμένος σε μια ακολουθία interprocessor επεξεργαστών.Η ισορροπία φορτίων δεν επηρεάζεται γενικά από το χρόνο επικοινωνίας interprocessor, αλλά τα συστήματα με τις χαμηλές λανθάνουσες καταστάσεις επικοινωνίας μπορούν να εκμεταλλευθούν τους αλγορίθμους που ισορροπούν δυναμικά το φορτίο μιας εφαρμογής με την κίνηση της εργασίας από τους επεξεργαστές που παίρνουν περισσότερο για να συμπληρώσουν το μέρος του προγράμματος σε άλλους επεξεργαστές έτσι ώστε κανένας επεξεργαστής δεν είναι πάντα αδρανής. Τα συστήματα με τις πιό μακροχρόνιες λανθάνουσες καταστάσεις επικοινωνίας ωφελούνται λιγότερο από αυτούς τους αλγορίθμους επειδή η λανθάνουσα κατάσταση επικοινωνίας καθορίζει πόσο καιρό παίρνει να κινήσει μια μονάδα εργασίας τον από έναν επεξεργαστή προς τον άλλο.

10.3.2 SUPERLINEAR Επιτάχυνση

Στο τελευταίο τμήμα, δηλώσαμε ότι η ιδανική επιτάχυνση σε ένα σύστημα πολυεπεξεργαστών ήταν ίση με τον αριθμό επεξεργαστών στο σύστημα.Γενικά αυτό ισχύει, αλλά μερικά προγράμματα επιδεικνύουν την superlinear επιτάχυνση, επιτυγχάνοντας επιτάχυνση μεγαλύτερη από *n* σε *n* - συστήματα επεξεργαστών.Η Superlinear επιτάχυνση εμφανίζεται επειδή τα προγράμματα είναι μερικές φορές αποδοτικότερα στα συστήματα πολυεπεξεργαστών απο ότι τα συστήματα μονο-επεξεργαστή, που επιτρέπουν σε κάθε ένα από τους επεξεργαστές σε ένα *n* - επεξεργαστή πολυεπεξεργασή για να συμπληρώσουν το μέρος του προγράμματος σε λιγότερο απο 1/*n* του χρόνου εκτέλεσης του προγράμματος μονο-επεξεργαστή.

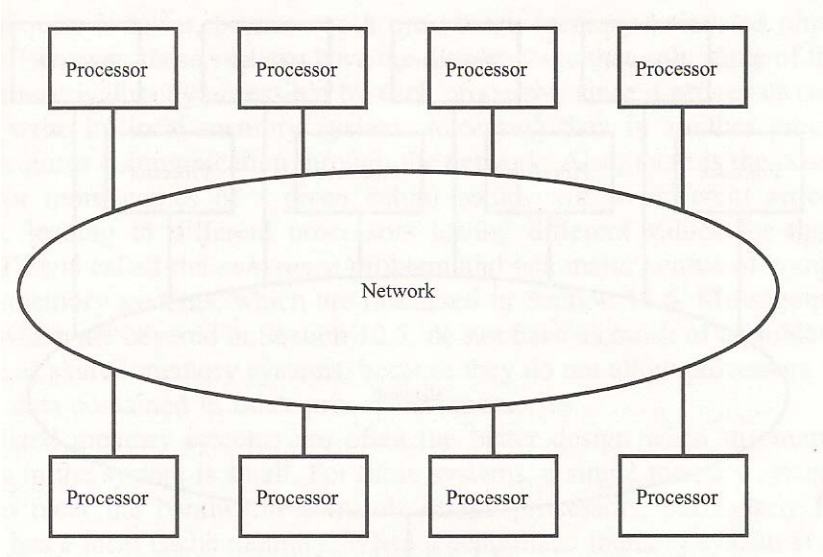
Υπάρχουν δύο κοινοί λόγοι για τους οποίους τα προγράμματα επιτυγχάνουν την superlinear επιτάχυνση:

1. *Αυξανόμενο μέγεθος κρύφής μνήμης*- σε έναν πολυεπεξεργαστή, κάθε επεξεργαστής συχνά έχει τόσο πολλή κρυφή μνήμη που συνδιάζεται με αυτό όπως ο απλός επεξεργαστής σε ένα μονο-επεξεργαστή.Ετσι, το συνολικό ποσό κρυφής μνήμης σε ένα πολυεπεξεργαστή είναι συχνά μεγαλύτερο από το συνολικό ποσό κρυφής μνήμης ενός μονο-επεξεργαστή .Όταν ένα πρόγραμμα του οποίου τα δεδομένα δεν ταιριάζουν στην κρυφή μνήμη του επεξεργαστή,αυξάνει το μέσο όρο λανθάνουσας κατάστασης της μνήμης και βελτιώνει την απόδοση.Εάν ένας μόνο-επεξεργαστής όπως ένας πολυεπεξεργαστής συγκρινόταν να είχε τόση κρυφή μνήμη όσο η συνολική μνήμη στον πολυεπεξεργαστή, το προγράμμα δεν θα επιδεικνυε την superlinear επιτάχυνση.

2. *Καλύτερη δομή* - Μερικά προγράμματα εκτελούν λιγοτερη εργασία όταν εκτελουνται σε έναν πολυεπεξεργαστή από ότι κάνουν όταν εκτελουνται σε ένα μονο-επεξεργαστή, επιτρέποντας σε αυτα να επιτύχουν την superlinear επιτάχυνση.Παραδείγματος χάριν, τα προγράμματα που ψάχνουν για την καλύτερη απάντηση σε ένα πρόβλημα για να εξετάσουν όλες τις δυνατότητες μερικές φορές εκθέτουν την superlinear επιτάχυνση επειδή η έκδοση των πολυεπεξεργαστών εξετάζει τις δυνατότητες μιας διαφορετικής διαταγής, μια που επιτρέπει σε αυτό να αποκλείσει περισσότερες δυνατότητες χωρίς την εξέταση τους.Επειδή η έκδοση πολυεπεξεργαστών πρέπει να εξετάσει λιγότερες συνολικές δυνατότητες από το πρόγραμμα μονο-επεξεργαστή , μπορεί να ολοκληρώσει την αναζήτηση με μεγαλύτερη από τη γραμμική επιτάχυνση.Ξαναγράφοντας το πρόγραμμα μονο-επεξεργαστή για να εξεταστούν οι δυνατότητες της ίδιας διαταγής όπως το πρόγραμμα πολυεπεξεργαστών θα βελτίωνε την απόδοσή του.

10.4 Συστήματα πολυεπεξεργαστών

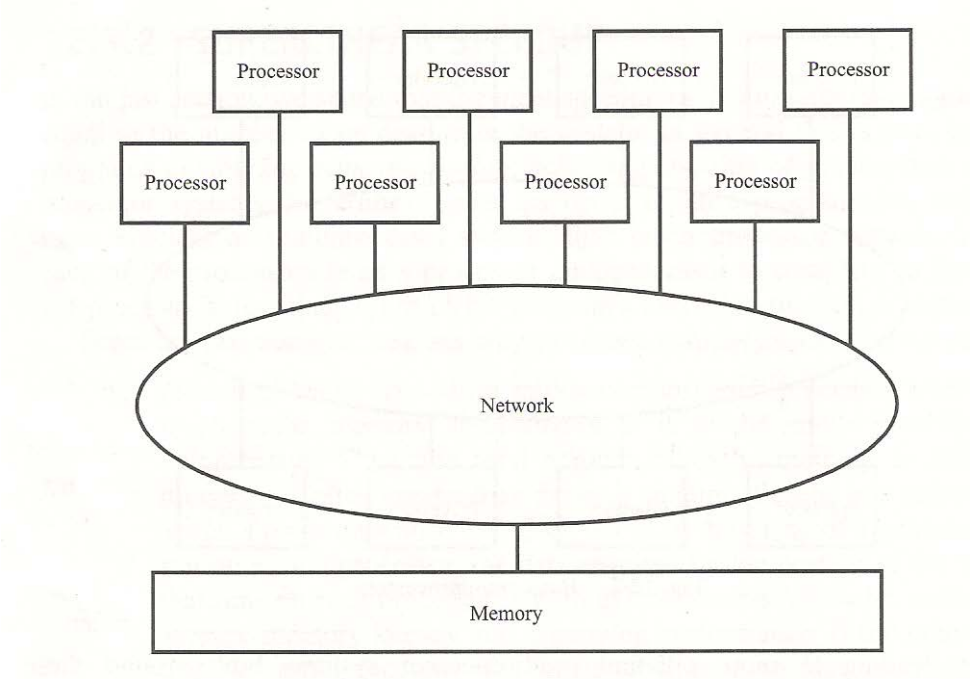
Οι πολυεπεξεργαστές αποτελουνται από ένα σύνολο επεξεργαστών που συνδέονται με ένα δίκτυο επικοινωνιών, όπως φαίνεται στο σχήμα 10-2.Οι πρόωροι πολυεπεξεργαστές χρησιμοποίησαν συχνά τους επεξεργαστές που ήταν σχεδιασμένοι συγκεκριμένα για τη χρήση σε έναν πολυεπεξεργαστή για να βελτιώσουν την αποδοτικότητα.Τα τελευταία χρόνια αυτό έχει αλλάξει, και οι περισσότεροι τρέχοντες πολυεπεξεργαστές χρησιμοποιούν τους ίδιους επεξεργαστές που βρίσκονται στα σύγχρονα συστήματα μονο-επεξεργαστη, που εκμεταλλεύονται τους μεγάλους όγκους πωλήσεων των μονο-επεξεργαστών σε χαμηλότερες τιμές.Δεδομένου ότι ο αριθμός κρυσταλλολυχνιών που μπορεί να τοποθετηθεί σε ένα τσιπ αυξάνεται, τα χαρακτηρίστικα για να υποστηρίξει τα συστήματα πολυεπεξεργαστών ενσωματώνονται στους επεξεργαστές προοριζόμενα πρώτιστα για την αγορά μονο-επεξεργαστή , που οδηγεί στα αποδοτικότερα συστήματα πολυεπεξεργαστών που χτίζονται γύρω από αυτό τον επεξεργαστή.



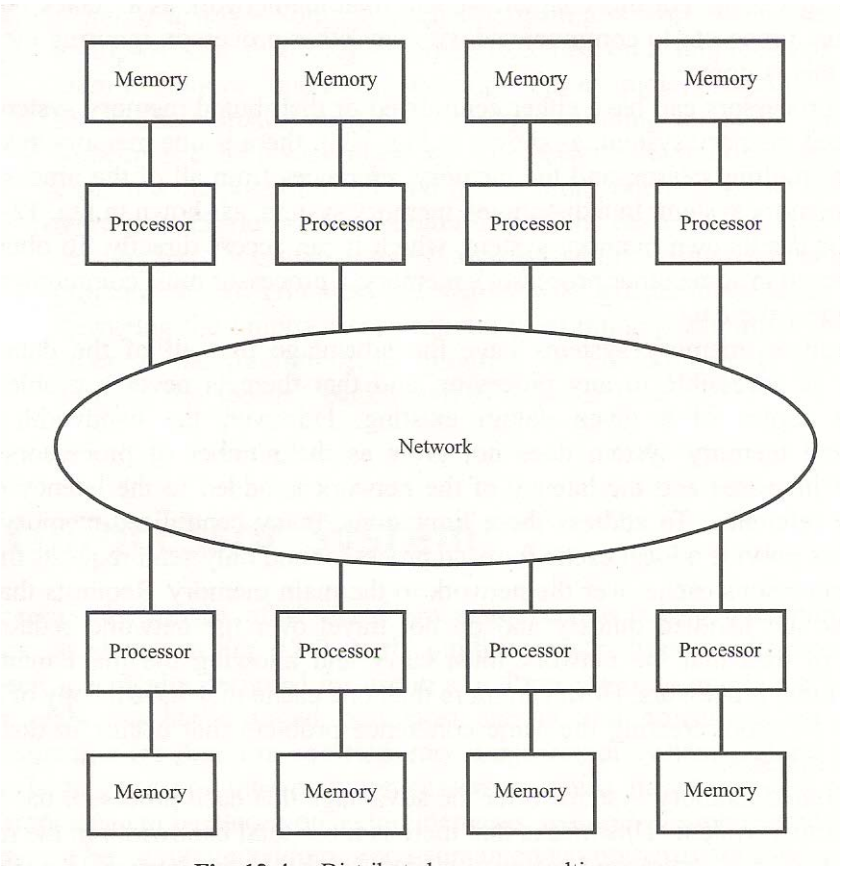
ΣΧΗΜΑ10-2 basic multiprocessor

Το σχέδιο των δικτύων για τους πολυεπεξεργαστές είναι ένα σύνθετο θέμα που είναι πέρα από το πεδίο αυτού του βιβλίου. Για αυτό το κεφάλαιο, θα μεταχειριστούμε το δίκτυο ως "μαύρο κιβώτιο" που επιτρέπει σε οποιοδήποτε επεξεργαστή να επικοινωνήσει με οποιοδήποτε επεξεργαστή, που αγνοεί τις λεπτομέρειες για το πώς αυτό γίνεται.

Οι πολυεπεξεργαστές μπορούν είτε να είναι συγκεντρωτικά είτε διανεμημενα συστήματα μνήμης.Σε ένα συγκεντρωτικό σύστημα μνήμης, όπως φαίνεται στο σχήμα 10-3, υπάρχει ένα σύστημα μνήμης για ολόκληρο τον πολυεπεξεργαστή, και οι αναφορές μνήμης από όλους τους επεξεργαστές πηγαίνουν σε εκείνη την μνήμη συστήματος.Σε ένα διανεμημένο σύστημα μνήμης, όπως παρουσιαζεται στο σχήμα 10-4, κάθε επεξεργαστής έχει το δικό του σύστημα μνήμης, στο οποίο μπορεί να έχει πρόσβαση άμεσα.Για να λάβει στοιχεία που αποθηκεύονται στη μνήμη κάποιου άλλου επεξεργαστή, ένας επεξεργαστής πρέπει να επικοινωνεί με αυτο για να ζητήσει τα στοιχεία.



ΣΧΗΜΑ10-3 entralized-memory multiprocessor



ΣΧΗΜΑ10-4 distributed-memory multiprocessor

Τα συγκεντρωτικά συστήματα μνήμης έχουν το πλεονέκτημα ότι όλα τα στοιχεία στη μνήμη είναι ανοιχτά σε οποιοδήποτε επεξεργαστή, και δεν υπάρχει ποτέ πρόβλημα με τα πολλαπλά αντίγραφα μιας δεδομένης ύπαρξης στοιχείων.Εντούτοις, το εύρος ζώνης του συγκεντρωτικού συστήματος μνήμης δεν αυξάνεται όπως αυξάνεται ο αριθμός επεξεργαστών στη μηχανή και η λανθάνουσα κατάσταση του δικτύου προστίθεται στη λανθάνουσα κατάσταση κάθε αναφοράς μνήμης.Για να εξετάσει αυτό τον περιορισμό, πολλοί πολυεπεξεργαστές συγκεντρωτικής μνήμης παρέχουν μια τοπική κρυφή μνήμη για κάθε επεξεργαστή και στέλνουν μόνο τα αιτήματα που λείπουν στην κρυφή μνήμη του επεξεργαστή πέρα από το δίκτυο στην κύρια μνήμη. Ζητά ότι αυτός που επιτυγχάνει στην κρυφή μνήμη αντιμετωπίζεται γρήγορα και δεν ταξιδεύει πέρα από το δίκτυο, μειώνει το ποσό των στοιχείων που το δίκτυο πρέπει να μεταφέρει και επιτρέπει στην κύρια μνήμη να υποστηρίξει περισσότερους επεξεργαστές.Όμως, περισσότερες από μια κρυφές μνήμες μπορεί να έχουν ένα αντίγραφο μιας δεδομένης θέσης μνήμης, που δημιουργεί το ίδιο πρόβλημα συνοχής που εμφανίζεται στα διανεμημένα συστήματα μνήμης.

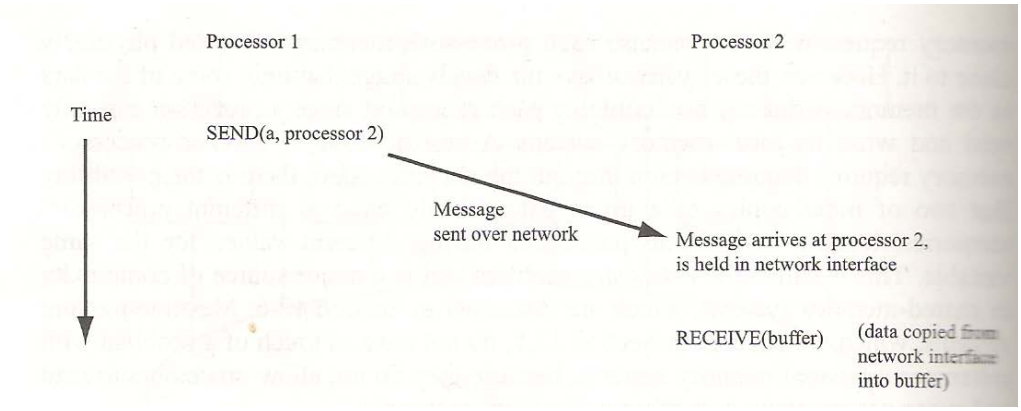
Τα διανεμημένα συστήματα μνήμης προσφέρουν το πλεονέκτημα ότι κάθε επεξεργαστής έχει το δικο του τοπικό σύστημα μνήμης.Αυτό σημαίνει ότι υπάρχει περισσότερο συνολικό εύρος ζώνης στο σύστημα μνήμης απ'ό,τι σε ένα συγκεντρωτικό σύστημα μνήμης, και ότι η λανθάνουσα κατάσταση για να συμπληρώσει ένα αίτημα μνήμης είναι χαμηλότερη, επειδή η μνήμη κάθε επεξεργαστή βρίσκεται φυσικά κοντά σε αυτό.Εντούτοις, αυτά τα συστήματα έχουν το μειονέκτημα ότι μόνο μερικά από τα στοιχεία στη μνήμη είναι άμεσα προσιτά από κάθε επεξεργαστή, δεδομένου ότι ένας επεξεργαστής μπορεί μόνο να διαβάσει και να γράψει το τοπικό σύστημα μνήμης του.Η πρόσβαση των στοιχείων στη μνήμη ενός άλλου επεξεργαστή απαιτεί την επικοινωνία μέσω του δικτύου. Επίσης, υπάρχει η δυνατότητα ότι δύο ή περισσότερα αντίγραφα ενός δεδομένου στοιχείου θα μπορούσαν να υπάρξουν σε διαφορετικές μνήμες επεξεργαστών . Αυτό καλείται πρόβλημα συνοχής και είναι μια σημαντική πηγή πολυπλοκότητας στα συστήματα διαχειριζόμενης μνήμης, τα οποία συζητούνται στην παράγραφο 10-6.Τα συστήματα μεταβίβασης μηνυμάτων, που καλύπτονται στην παράγραφο 10-5, δεν έχει τόσο μεγάλο προβλήμα με τη συνοχή όπως τα συστήματα διαχειριζόμενης μνήμης, επειδή δεν επιτρέπουν στους επεξεργαστές να διαβάσουν και να γράψουν τα στοιχεία που περιλαμβάνονται στις μνήμες άλλου επεξεργαστή .

Τα συγκεντρωτικά συστήματα μνήμης είναι συχνά το καλύτερο σχέδιο όταν ο αριθμός επεξεργαστών στο σύστημα είναι μικρός.Για αυτά τα συστήματα, ένα ενιαίο σύστημα μνήμης μπορεί να είναι σε θέση να ικανοποιήσει τις απαιτήσεις εύρους ζώνης των επεξεργαστών, ιδιαιτέρως εάν κάθε επεξεργαστής έχει μια τοπική κρυφή μνήμη.Όταν ένα συγκεντρωτικό σύστημα μνήμης είναι σε θέση να ικανοποιήσει τις απαιτήσεις εύρους ζώνης των επεξεργαστών,η μείωση της πολυπλοκότητας σχεδίου και προγραμματισμού που προέρχεται από να μην πρέπει να ρυθμιστουν οι πολλαπλάσιες ανεξάρτητες μνήμες είναι ένα ισχυρό επιχείρημα υπέρ αυτού του τύπου συστήματος μνήμης.Δεδομένου ότι ο αριθμός επεξεργαστών αυξάνεται, γίνεται αδύνατο για ένα συγκεντρωτικό σύστημα μνήμης να ικανοποιηθούν οι ανάγκες εύρους ζώνης των επεξεργαστών, και είναι απαραίτητο να χρησιμοποιηθεί σε αυτο το σύστημα μνήμης. Τα διανεμημένα συστήματα μνήμης χρησιμοποιούνται επίσης όταν η λανθάνουσα κατάσταση του δικτύου είναι αρκετά μεγάλη όταν η χρησιμοποίηση ενός συγκεντρωμένου συστήματος μνήμης θα καθιστούσε τις λανθάνουσες καταστάσεις μνήμης απαράδεκτα μακροχρόνιες, ακόμα κι αν ένα συγκεντρωτικό σύστημα μνήμης θα μπορούσε να ικανοποιήσει τις ανάγκες εύρους ζώνης των επεξεργαστών.

10.5 Συστήματα μεταβίβασης μηνυμάτων

Υπάρχουν δύο σημαντικά πρότυπα προγραμματισμού για τα συστήματα πολυεπεξεργαστών : η διαχειριζόμενη μνήμη και η μετάβαση μηνυμάτων.Στα συστήματα διαχειριζόμενης μνήμης, το σύστημα μνήμης διευθύνει την επικοινωνία interprocessor επιτρέποντας σε όλους τους επεξεργαστές να δούν τα στοιχεία που γράφονται από οποιοδήποτε επεξεργαστή. Αντίθετα, τα συστήματα μεταβίβασης μηνυμάτων επικοινωνουν μέσω ρητών μηνυμάτων. Για να στείλει ένα μήνυμα ένας επεξεργαστής εκτελεί την SEND (στοιχεία, προορισμός) λειτουργία (γενικά μια κλήση διαδικασίας) που καθοδηγεί το υλικό να στείλει τα διευκρινισμένα στοιχεία στον επεξεργαστή προορισμό.Αργότερα,ένας επεξεργαστής προορισμός εκτελεί μία λειτουργία RECEIVE (buffer) για να αντιγράψει την αποστολή των δεδομένων στον διευκρινισμένο απομονωτή, καθιστώντας αυτό διαθέσιμο για χρήση.Εάν ο επεξεργαστής που είναι υπεύθυνος για την αποστολή των στοιχείων δεν έχει εκτελέσει τη λειτουργία SEND προτού να εκτελεσθεί η RECEIVE, η RECEIVE λειτουργία περιμένει την SEND να ολοκληρώσει, ενισχύει την διαταγή των SEND και RECEIVE διαδικασιών.Αυτή η διαδικασία είναι διευκρινισμένη στο σχήμα 10-5.

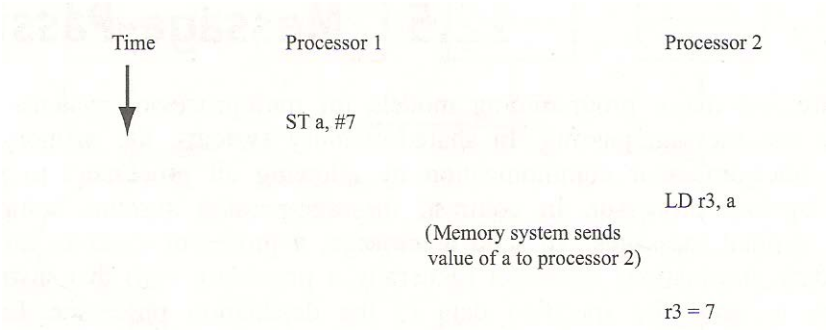
Στα συστήματα μεταβίβασης μηνυμάτων, κάθε επεξεργαστής έχει το διάστημα διευθύνσεών του, και οι επεξεργαστές δεν μπορούν να διαβάσουν ή να γράψουν τα στοιχεία που περιλαμβάνονται στο διάστημα διευθύνσεων ενός άλλου επεξεργαστή. Λόγω αυτού, πολλά συστήματα μεταβίβασης μηνυμάτων εφαρμόζονται ως διανημενες - μηχανές μνήμης, δεδομένου ότι μπορούν να συγκεντρώσουν τα οφέλη λανθάνουσας κατάστασης της ένωσης της μνήμη με κάθε επεξεργαστή χωρίς την πολυπλοκότητα οι επεξεργαστές να πρέπει να έχουν πρόσβαση ο ένας στις μνήμες του άλλου.



ΣΧΗΜΑ10-5 message-passing

10.6 Συστήματα διαχειριζόμενης μνήμης

Στα συστήματα διαχειριζόμενης μνήμης,η επικοινωνία είναι υπονοούμενη, παρά αναμφίβολη.Τα συστήματα διαχειριζόμενης μνήμης παρέχουν ένα ενιαίο διάστημα διευθύνσεων οπου όλοι οι επεξεργαστές μπορούν να διαβάσουν και να γράψουν.Όταν ένας επεξεργαστής γράφει μια θέση στο διάστημα διευθύνσεων,επομένως διαβάζει εκείνη τη θέση από οποιοδήποτε επεξεργαστή που βλέπει για να γράψει το αποτέλεσμα, όπως διευκρινίζεται στο σχήμα 10-6.Το σύστημα εκτελεί οποιαδήποτε επικοινωνία interprocessor που απαιτείται για να καταστήσει τις διαδικασίες μνήμης ορατές σε όλους τους επεξεργαστές .



ΣΧΗΜΑ10-6 shared-memory example

10.6.1 Πρότυπα συνοχής μνήμης

Ένα μεγάλο μέρος της πολυπλοκότητας που περιλαμβάνεται στο σχεδίασμο ενός συστήματος διαχειριζόμενης μνήμης προέρχεται από το γεγονός ότι ο πολυεπεξεργαστής πρέπει να παρουσιάσει την παραίσθηση ότι υπάρχει ένα ενιαίο σύστημα μνήμης παρά το γεγονός ότι υπάρχουν πολλαπλάσιες φυσικές μνήμες στη μηχανή.Οι επεξεργαστές διαχειριζόμενης μνήμης μπορούν να εφαρμοστούν είτε ως συγκεντρωτική μνήμη είτε ως διανεμημένα συστήματα μνήμης, αλλά καθένα αυτών των σχεδίων θα συνδέσει τις κρυφές μνήμες με κάθε επεξεργαστή για να μειώσει τη λανθάνουσα κατάσταση μνήμης, που δημιουργεί τη πιθανότητα ότι τα πολλαπλάσια αντίγραφα ενός δεδομένου στοιχείου θα υπάρξουν στις διαφορετικές κρυφές μνήμες.Το σύστημα διαχειριζόμενης μνήμης είναι αρμόδιο για τη διευκρίνιση των όρων κάτω από τους οποίους τα πολλαπλάσια αντίγραφα ενός στοιχείου μπορούν να υπάρξουν, και για την επιβολή εκείνων των συνθηκών έτσι ώστε οι προγραμματιστές να μπορούν να γράψουν τα προγράμματα που εκτελούν σωστά .

Το πρότυπο συνοχής μνήμης ενός πολυεπεξεργαστή καθορίζει πότε οι διαδικασίες μνήμης που εκτελούνται σε έναν επεξεργαστή γίνονται ορατές σε άλλους επεξεργαστές. Το συνηθέστερο χρησιμοποιημένο πρότυπο συνοχής μνήμης είναι η ισχυρή συνοχή, η οποία υπαγορεύει ότι η μνήμη του συστημάτος μνήμης ενεργεί ακριβώς σαν να υπήρξε μόνο μια μνήμη στον υπολογιστή όπου διαφορετικοί

επεξεργαστές εναλλάσονται στη χρησιμοποίηση .Υπάρχουν και άλλα πρότυπα συνοχής, τα περισσότερα απο αυτά εφαρμόζουν τις διάφορες μορφές χαλαρωμένης συνοχής με την άδεια των διαφορετικών επεξεργαστών για να υπάρξουν οι διαφορετικές τιμές για ένα δεδομένο στοιχείο .

10.6.2 Ισχυρή συνοχή

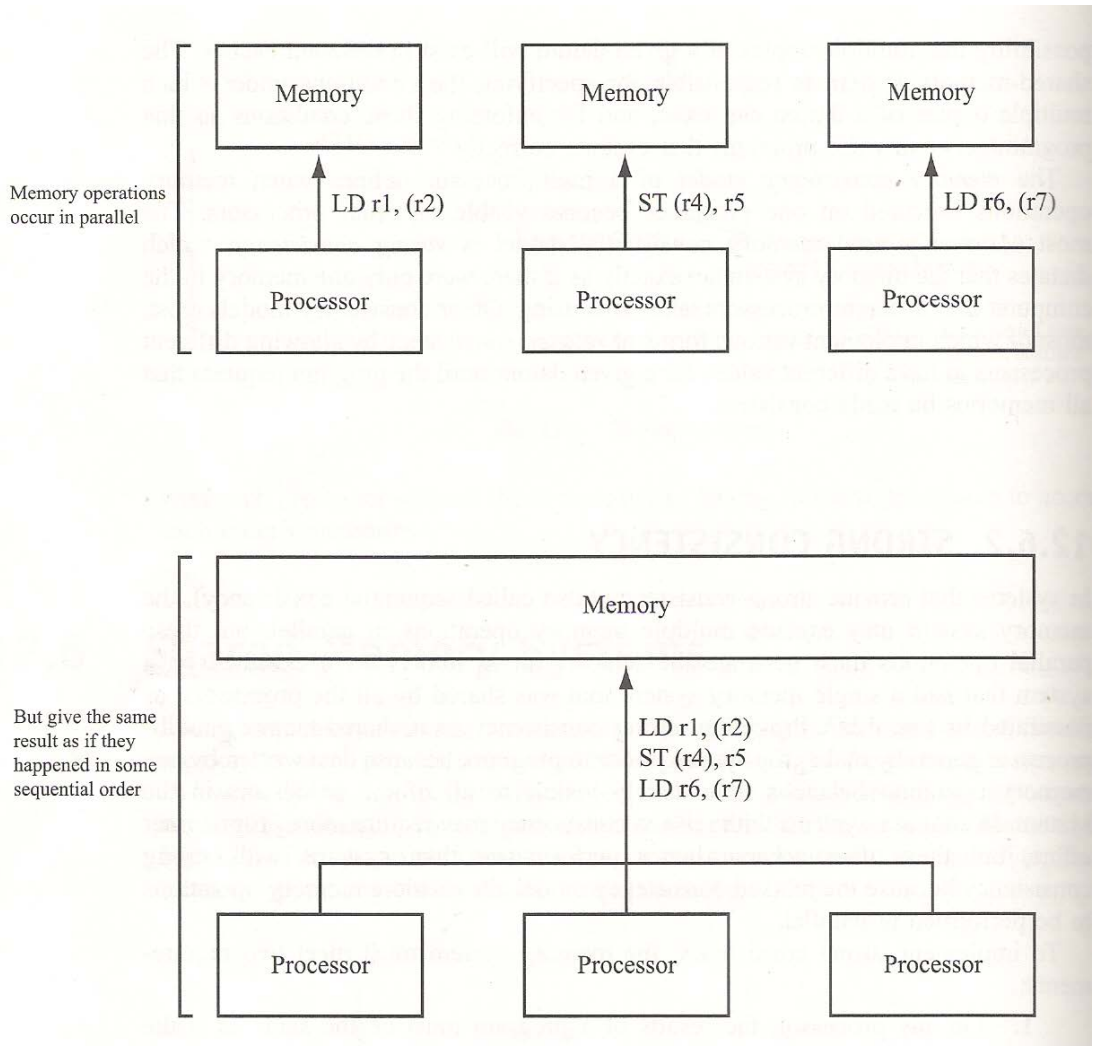
Στα συστήματα που παρέχουν την ισχυρή συνοχή (επίσης αποκαλούμενη διαδοχική συνοχή), το σύστημα μνήμης μπορεί να εκτελέσει τις πολλαπλάσιες διαδικασίες μνήμης παράλληλα, αλλά αυτές οι παράλληλες διαδικασίες πρέπει να παραγάγουν το ίδιο αποτέλεσμα σαν να εκτελέσθηκαν σε ένα σύστημα που είχε ένα ενιαίο σύστημα μνήμης που μοιράστηκε σε όλους τους επεξεργαστές, όπως διευκρινίζεται στο σχήμα 10-7. Η παροχή της ισχυρής συνοχής σε έναν πολυεπεξεργαστή διαχειριζόμενης μνήμης καθιστά γενικά το συστήμα ευκολότερο στο πρόγραμμα, επειδή τα στοιχεία που γράφονται από οποιαδήποτε λειτουργία μνήμης γίνονται αμέσως ορατά σε όλους τους επεξεργαστές στο σύστημα.Αντίθετα, τα συστήματα με τη χαλαρωμένη συνοχή μπορούν να απαιτήσουν περισσότερη προσπάθεια προγραμματιστών, αλλά επιτυγχάνουν συχνά την καλύτερη απόδοση από ότι τα συστήματα με την ισχυρή συνοχή επειδή το πρότυπο της χαλαρωμένης συνοχής επιτρέπει σε περισσότερες διαδικασίες μνήμης για να εκτελεσθούν παράλληλα.

Για να εφαρμόσει την ισχυρή συνοχή, το σύστημα μνήμης πρέπει να καλύψει δύο απαιτήσεις:

1.Σε οποιοδήποτε επεξεργαστή, τα αποτελέσματα ενός προγράμματος πρέπει να είναι τα ίδια όπως οι διαδικασίες μνήμης στο πρόγραμμα βρίσκονται στη διαταγή όπου εμφανίζονται στο πρόγραμμα. Οι Out-of-order επεξεργαστές απαιτούν γενικά αυτό απο κάθε προγράμμα που εκτελούν, έτσι είναι δυνατό να εφαρμοστεί η ισχυρή συνοχή σε έναν Out-of-order επεξεργαστή .

2. Σε όλους τους επεξεργαστές, τα αποτελέσματα όλων των διαδικασιών μνήμης πρέπει να είναι τα ίδια σαν είχαν συμβεί σε κάποια διαδοχική διαταγή.Ουσιαστικά, αυτό σημαίνει ότι εάν ένας επεξεργαστής βλέπει δύο διαδικασίες μνήμης σε μια ιδιαίτερη διαταγή, όλοι οι επεξεργαστές πρέπει να δουν εκείνες τις διαδικασίες να συμβούν στην ίδια διαταγή.

Η ισχυρή συνοχή επιτρέπει αναφορές σε διαφορετικές διευθύνσεις για να προχωρήσει παράλληλα,επειδή η εκτέλεση των αναφορών σε διαφορετικές διευθύνσεις παράλληλα δίνει τα ίδια αποτελέσματα σαν αυτές να εκτελούνταν διαδοχικά.Πολλαπλά διαβάσματα της ίδιας διεύθυνσης μπορούν επίσης να προχωρήσουν παράλληλα,επειδή θα επιστρέψουν το ίδιο αποτέλεσμα απρόσεκτα εάν εκτελούνται παραλληλα ή διαδοχικά.Εντούτοις,διαβάζοντας και γράφοντας σε μια διεύθυνση ή πολλαπλά γράψίματα σε μια διεύθυνση πρέπει να τοποτεθετηθούν σειριακά έτσι ώστε κάθε γράψιμο να μπορεί να φανεί ότι έχει εκτελέστει σε έναν συγκεκριμένο χρόνο .



ΣΧΗΜΑ 10-7 strong consistency model

10.6.3 Συνοχή κρυφής μνημής

Το πρωτόκολλο συνοχής κρύφης μνήμης ενός πολυεπεξεργαστή διαχειριζόμενης μνήμης καθορίζει πώς τα στοιχεία μπορούν να μοιραστούν και να αντιγραφούν στους επεξεργαστές.Ενώ το πρότυπο συνοχής μνήμης καθορίζει ότι όταν τα προγράμματα τρέχουν για τους επεξεργαστές μπορούν να δουν τις διαδικασίες που εκτελούνται στους άλλους επεξεργαστές, το πρωτόκολλο συνοχής κρύφης μνήμης καθορίζει το συγκεκριμένο σύνολο ενεργειών που εκτελούνται για να κρατήσουν την άποψη κάθε επεξεργαστή του συστήματος μνήμης συνεπη.Γενικά, τα πρωτόκολλα συνοχής κρύφης μνήμης λειτουργούν στις γραμμές κρύφης μνήμης στοιχείων σε κάποιο χρόνο, επικοινωνώντας μια ολόκληρη γραμμή μεταξύ των επεξεργαστών όταν χρειάζεται, παρά να στείλουν μια μεμονωμένη λέξη.

Τα πρωτόκολλα συνοχής κρύφης μνήμης μπορούν να διαιρεθούν σε δύο κατηγορίες:βασισμένα στην ακύρωση και βασισμένα στον εκσυγχρονισμό.Σε ένα βασισμένο στην ακύρωση πρωτόκολλο, όπως διευκρινίζεται στο σχήμα 10-8, οι πολλαπλοι επεξεργαστές επιτρέπεται για να έχουν διαβάσει μόνο τα αντίγραφα μιας γραμμής κρυφής μνήμης εάν κανένας επεξεργαστής δεν έχει ένα εγγράψιμο αντίγραφο στη γραμμή.Μόνο ένας επεξεργαστής μπορεί να έχει ένα εγγράψιμο αντίγραφο μιας δεδομένης γραμμής οποιαδήποτε στιγμή, και κανένας επεξεργαστής δεν μπορεί να έχει ένα διαβασμένο αντίγραφο εάν οποιοσδήποτε επεξεργαστής έχει ένα εγγράψιμο αντίγραφο.Όταν ένας επεξεργαστής θέλει να γράψει μια γραμμή όπου ένας ή περισσότεροι άλλοι επεξεργαστές έχουν αντίγραφα (είτε διαβασμένος μόνο είτε εγγράψιμος), η γραμμή ακυρώνεται, αναγκάζοντας όλους τους επεξεργαστές που έχουν αυτήν την περίοδο αντίγραφα της γραμμής για να

Events	Processor 1	Processor 2	Processor 3
Start	No Copy	No Copy	No Copy
Processor 1 reads line	Read-Only Copy	No Copy	No Copy
Processor 2 reads line	Read-Only Copy	Read-Only Copy	No Copy
Processor 3 writes line (other copies invalidated)	No Copy	No Copy	Writable Copy
Processor 2 reads line	No Copy	Read-Only Copy	Read-Only Copy

ΣΧΗΜΑ10-8 invalidation-based cache-coherence

σταματήσουν τα αντίγραφά τους έτσι ώστε ο αιτούμενος επεξεργαστής να μπορεί να πάρει ένα αντίγραφο στην κατάλληλη κατάσταση.

Τα βασισμένα στον εκσυγχρονισμό πρωτόκολλα, όπως διευκρινίζονται στο σχήμα 10-9, επιτρέπουν στους πολλαπλους επεξεργαστές για να έχουν τα εγγράψιμα αντίγραφα μιας γραμμής.Όταν ένας επεξεργαστής γράφει σε μια γραμμή όπου ένας ή περισσότερι επεξεργαστές έχουν αντίγραφα ,ένας εκσυγχρονισμός εμφανίζεται,διαβιβάζει εκείνη την μια αξία των στοιχείων στη γραμμή σε όλους τους μοιραμένος επεξεργαστές.Ανάλογα με την εφαρμογή, είτε ένα πρωτόκολλο βασισμένο στην ακύρωση είτε ένα πρωτόκολλο βασισμένο στον εκσυγχρονισμό μπορούν να δώσουν την καλύτερη απόδοση .Τα βασισμένα στην ακύρωση πρωτόκολλα δίνουν γενικά την καλύτερη απόδοση στις εφαρμογές με την ουσιαστική τοποθεσία στοιχείων, επειδή τα βασισμένα στόν εκσυγχρονισμό πρωτόκολλα απαιτούν μια επικοινωνία κάθε φορά που γράφεται μια κοινή γραμμή.Τα βασισμένα στην ακύρωση πρωτόκολλα υφίστανται μόνο την επικοινωνία όταν ένας επεξεργαστής που δεν έχει ένα αντίγραφο στην γραμμή χρειάζεται να έχουν προσβαση στη γραμμή ή ένας επεξεργαστής με ένα διαβασμένο αντίγραφο στην γραμμή χρειάζεται να γράψει στην γραμμή,η οποία μπορεί να οδηγήσει σε πολύ χαμηλότερες δαπάνες επικοινωνίας εάν οι επεξεργαστές εκτελούν περισσότερες διαδικασίες μνήμης μεταξύ του χρόνου που έχουν πρόσβαση αρχικά σε μια γραμμή και στο χρόνο που κάποιοι άλλοι επεξεργαστές έχουν πρόσβαση στη γραμμή.

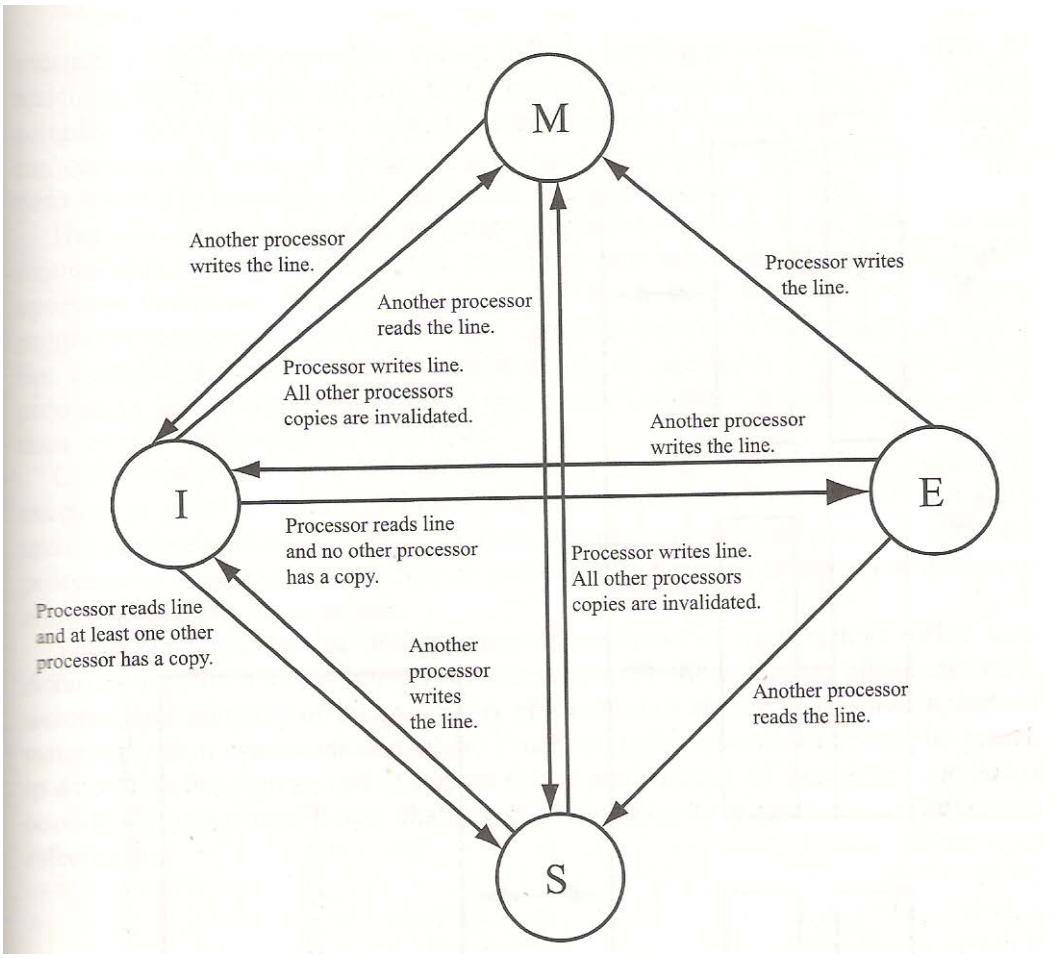
Αντίθετα, τα βασισμένα στον εκσυγχρονισμό πρωτόκολλα μπορούν να επιτύχουν την καλύτερη απόδοση στα προγράμματα όπου ένας επεξεργαστής ενημερώνει επανειλημμένα ένα στοιχείο που διαβάζεται από άλλους επεξεργαστές.Σε αυτήν την περίπτωση, μπορεί να είναι αποδοτικότερο να σταλεί κάθε νέα αξία του στοιχείου σε όλους τους επεξεργαστές που χρειαζεται αυτό από το να ακυρωθούν όλα τα αντίγραφα της γραμμής που περιέχει τα στοιχεία κάθε φορά που γράφεται.

Events	Processor 1	Processor 2	Processor 3
Start	No Copy	No Copy	No Copy
Processor 1 reads line	Writable Copy	No Copy	No Copy
Processor 2 reads line	Writable Copy	Writable Copy	No Copy
Processor 3 writes line (updates sent to processors 1 and 2)	Writable Copy	Writable Copy	Writable Copy
Processor 2 reads line (sees value written by Processor 3)	Writable Copy	Writable Copy	Writable Copy

ΣΧΗΜΑ10-9 update-based cache-coherence protocol

10.6.4 Πρωτόκολλο MESI

Το πρωτόκολλο MESI είναι συνήθως ένα χρησιμοποιούμενο βασισμένο στην ακυρωση πρωτόκολλο συνοχής κρύφης μνήμης. Στο MESI, σε κάθε γραμμή στην κρύφή μνήμη ενός επεξεργαστής διορίζεται μία από τις τέσσερις καταστάσης στη διαδρομή που οι κρύφες μνήμες



ΣΧΗΜΑ10-10 protocol MESI

έχουν τα αντίγραφα της γραμμής: τροποποιημένη, αποκλειστικη,μοιρασμένη, ή άκυρη.Η άκυρη κατάσταση σημαίνει ότι ο επεξεργαστής δεν έχει ένα αντίγραφο της γραμμής. Οποιαδήποτε πρόσβαση στη γραμμή θα απαιτήσει ότι το σύστημα διαχειριζόμενης μνήμης στέλνει ένα μήνυμα αιτήματος στη μνήμη που περιέχει τη γραμμή για να πάρει ένα αντίγραφο της γραμμής.Η μοιρασμένη κατάσταση σημαίνει ότι ο επεξεργαστής έχει ένα αντίγραφο της γραμμής, και ότι ένας ή περισσότεροι άλλοι επεξεργαστές έχουν επίσης αντίγραφα.Ο επεξεργαστής μπορεί να διαβάσει από τη γραμμή, αλλά οποιαδήποτε προσπάθεια να γραφτεί η γραμμή απαιτεί ότι τα άλλα αντίγραφα της γραμμής θα ακυρωθούν.Εάν η γραμμή είναι στην αποκλειστική κατάσταση ,ο επεξεργαστής είναι ο μόνος που έχει αντίγραφο της γραμμής ,αλλά αυτό δεν έχει γράφτει στη γραμμή δεδομένου ότι το αντίγραφο είναι επίκτητο.Η τροποποιημένη κατάσταση σημαίνει ότι ο επεξεργαστής είναι ο μονος με ένα αντίγραφο της γραμμής, και αυτό έχει γράφτει στη γραμμή δεδομένου ότι απόκτησε το αντίγραφο. Και στην αποκλειστική και στην τροποποιημένή κατασταση, ο επεξεργαστής μπορεί να διαβάσει και να γράψει τη γραμμή ελεύθερα.

Το πρωτόκολλο MESI διακρίνεται μεταξύ των αποκλειστικών και τροποποιημένων καταστάσεων έτσι ώστε το σύστημα να μπορεί να υπολογίσει που αποθηκεύεται η πιό πρόσφατη τιμή μιας γραμμής.Εάν μια γραμμή είναι αποκλειστική στην κρύφη μνήμη ενός επεξεργαστή, το αντίγραφο της γραμμής στην κύρια μνήμη είναι ενημερωμένο, και ο επεξεργαστής μπορεί ακριβώς να απορρίψει τη γραμμή εάν πρέπει να ακυρώσει τη γραμμή.Εάν τροποποιείται, ο επεξεργαστής έχει την πιό πρόσφατη τιμή της γραμμής, και πρέπει να γράψει τη γραμμή στην κύρια μνήμη όταν ακυρώνεται.Το σχήμα 10-10 παρουσιάζει αλλαγές καταστάσεων στο πρωτόκολλο MESI.

10.6.5 Βασισμένα στην δίοδο συστήματα διαχειριζόμενης μνήμης

Ενα κοινό σχέδιο για τα συστήματα διαχειριζόμενης μνήμης είναι το βασισμένο στη δίοδο σύστημα που παρουσιάζεται στο σχήμα 10-11.Σε αυτά τα συστήματα,μια δίοδος χρησιμοποιείται ως δίκτυο επικοινωνιών που συνδέει τους επεξεργαστές τον ένα με τον άλλο και το συγκεντρωτικό σύστημα μνήμης.Τα βασισμένα στη δίοδο συστήματα διαχειριζόμενης μνήμης χρησιμοποιούνται επειδή η δίοδος επιτρέπει σε έναν μεταβλητό αριθμό επεξεργαστών να επικοινωνήσει ο ένας με τον άλλον χωρίς αλλαγή του υλικού και επειδή η δίοδος καθιστά εύκολο να εφαρμόστει η συνοχή κρύφης μνήμης.Το αρχικό μειονέκτημα των βασισμένων στη δίοδο συστημάτων διαχειριζόμενης μνήμης είναι ότι το διαθέσιμο εύρος ζώνης πέρα από τη δίοδο δεν μπορεί να αυξηθεί όπως αυξάνεται ο αριθμός των επεξεργαστών στο σύστημα ,που καθιστούν αυτους ακατάλληλους για τους πολυεπεξεργαστές περισσότερο από μερικούς επεξεργαστές.

Η διατήρηση της συνοχής κρύφης μνήμης σε έναν βασισμένο στη δίοδο πολυεπεξεργαστή είναι εύκολη επειδή κάθε επεξεργαστής στο σύστημα μπορεί να παρατηρήσει την κατάσταση της διόδου μνήμης, που επιτρέπει σε αυτην να δει οποιαδήποτε αιτήματα που άλλοι επεξεργαστές υποβάλλουν στην κύρια μνήμη.Αυτό καλείται επισκόπηση κρύφης μνήμης, επειδή όλες οι κρύφες μνήμες κατασκοπεύουν τις ενέργειες των επεξεργαστών.Όταν ένας επεξεργαστής κάνει μια αναφορά μνήμης σε μια διεύθυνση που περιεχει η κρύφη μνήμη ενός άλλου επεξεργαστή, ο άλλος επεξεργαστής μπορεί να δει το αίτημα, να αποκριθεί με τα απαραίτητα στοιχεία, και να τροποποιήσει την κατάσταση του αντιγράφου του κατάλληλα χωρίς πάντα να περιλάβει την κύρια μνήμη.Κατά συνέπεια, οι αναφορές μνήμης σε ένα βασισμένο στη δίοδο σύστημα διαχειριζομένης μνήμης μπορούν πραγματικά να ολοκληρώσουν γρηγορότερα εάν κάποιος άλλος επεξεργαστής έχει ένα αντίγραφο της απαραίτητης γραμμής από το εάν η γραμμή πρέπει να διαβάσει από την κύρια μνήμη .

10.6.6 Συγχρονισμός

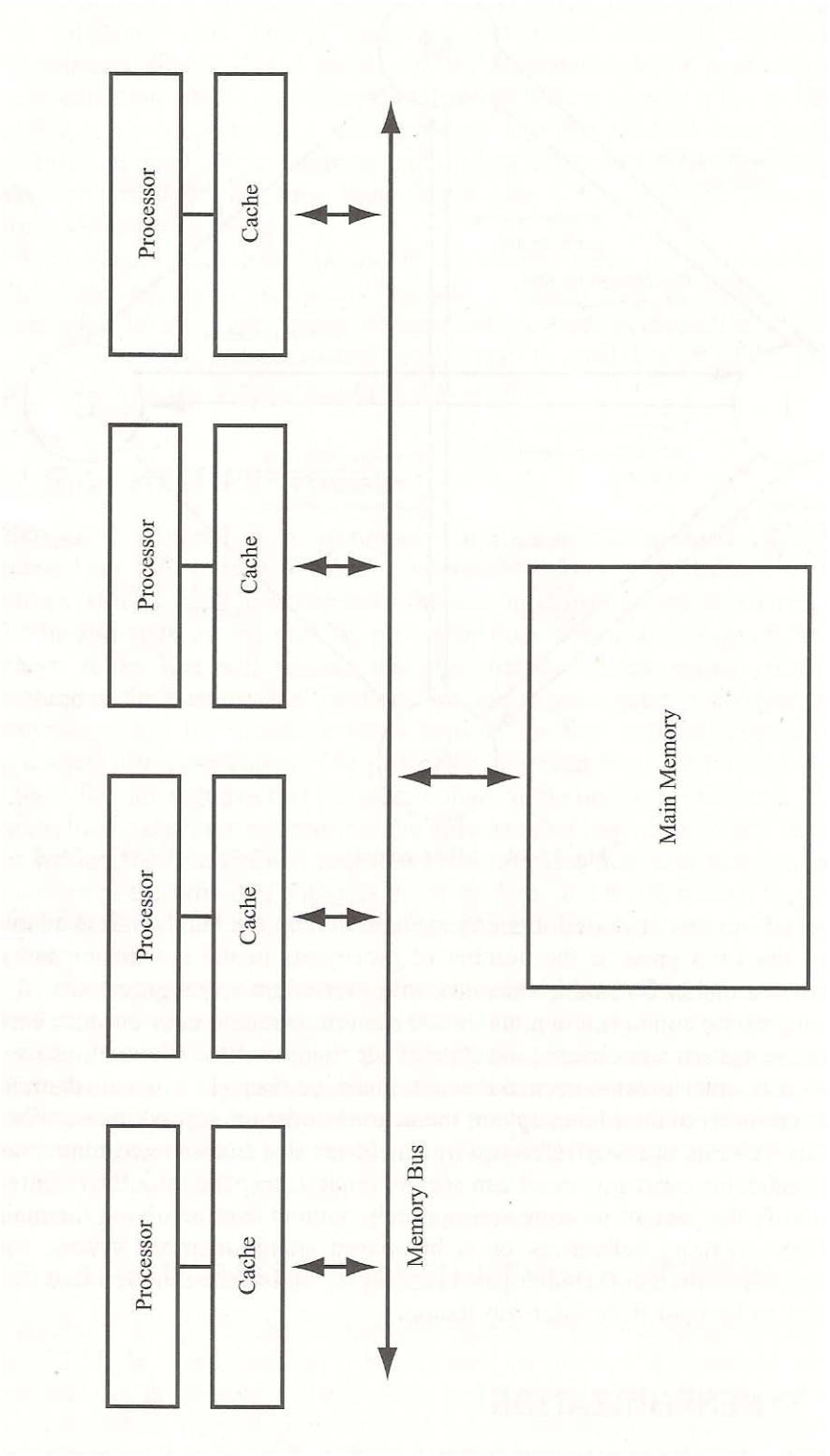
Ένα πρόβλημα με τα συστήματα διαχειριζόμενης μνήμης είναι ότι διατάζει ελάχιστα των διαδικασιών μεταξύ των επεξεργαστών.Στα συστήματα μεταβίβασης μηνυμάτων , ένας επεξεργαστης που εκτελεί μια RECEIVE λειτουργία ξέρει ότι η αποστολή στον επεξεργαστή έχει εκτελέσει το ταίριασμά του SEND προτού να ολοκληρώσει η RECEIVE, επειδή η RECEIVE δεν μπορεί να ολοκληρώσει έως ότου γίνεται η SEND. Στα συστήματα διαχειριζόμενης μνήμης, οι αναφορές μνήμης μπορούν να εμφανιστούν οποιαδήποτε στιγμή, έτσι το πρόγραμμα πρέπει να εκτελέσει τις αναμφίβολες διαδικασίες συγχρονισμού όταν είναι απαραίτητο όπου οι διαδικασίες εμφανίζονται σε μια ιδιαίτερη διαταγή.

Υπάρχουν δύο κοινές διαδικασίες συγχρονισμου:φραγμοί και κλειδία.Οι φραγμοί απαιτούν ότι κάθε επεξεργαστής στον πολυεπεξεργαστή φθάνει στην έναρξη της λειτουργίας φραγμών προτού να μπορέσει να ολοκληρώσει οποιοσδήποτε επεξεργαστής τον φραγμό.Είναι χρήσιμοι όταν εκτελείται ένα πρόγραμμα στις φάσεις που πρέπει να ολοκληρώσουν προτού να μπορέσει να αρχίσει η επόμενη

φάση.Παραδείγματος χάριν, μια παράλληλη προσομοίωση χρησιμοποιει τους φραγμούς για να εξασφαλίσει ότι όλοι οι επεξεργαστές είχαν τελειώσει τη προσομοίωση σε μια δεδομένη ώρα πρίν να επιτρέψουν σε οποιοδήποτε από αυτα να αρχίσουν την επόμενη ώρα.

Τα κλειδία εξασφαλίζουν ότι μόνο ένας επεξεργαστής τη φορά έχει πρόσβαση σε μια δεδομένη μεταβλητή ή εκτελεί μια δεδομένη διαδικασία.Όταν ένας επεξεργαστής εκτελεί μια λειτουργία κλειδιών, χρονοτριβεί έως ότου έχει χορηγησεί η πρόσβαση στα κλειδιά,σε ποιο σημείο μπορεί να προχωρήσει.Όταν ο επεξεργαστής γίνεται με τη μεταβλητή ή τη διαδικασία, εκτελεί μια λειτουργία ξεκλειδώματος για να επιτρέψει μια άλλη πρόσβαση επεξεργαστών.

Υπάρχει ένας αριθμός απο διάφορες εφαρμογές για φραγμούς και κλειδιά , με τις διάφορες εφαρμογές που είναι καταλληλότερες στα διαφορα δίκτυα, τις αρχιτεκτονικές επεξεργαστών, και τους αριθμούς επεξεργαστών.Η επιλογή ενός προγραμματιστή για να εφαρμόσει τις διαδικασίες συγχρονισμού και που οι διαδικασίες συγχρονισμού πρέπει να τοποθετηθούν σε ένα πρόγραμμα ασκεί μεγάλη επίδραση στην απόδοση ενός προγράμματος, και αυτό είναι μια από τις σημαντικότερες προκλήσεις για να μάθει να προγραμματίζει τους πολυεπεξεργαστές αποτελεσματικά .



ΣΧΗΜΑ10-11 bus-based shared-memory system

10.7 Σύγκριση μεταβίβασης μηνύματος και της διαχειριζόμενης μνήμης

Είναι πιθανό ότι η μεταβίβαση μηνυμάτων και η διαχειριζόμενη μνήμη θα συνεχίσουν να είναι τα κυρίαρχα παραδείγματα επικοινωνίας πολυεπεξεργαστών για το εγγύς μέλλον.Κάθε προσέγγιση έχει τα πλεονεκτήματα και τα μειονεκτήματα της που κρατούν μια μέθοδο από το να είναι κυρίαρχη πέρα από κάθε άλλη.Το σημαντικότερο πλεονέκτημα των συστημάτων διαχειριζόμενης μνήμης είναι ότι ο υπολογιστής χειρίζεται την επικοινωνία, που καθιστά αυτο πιθανό να γράψει ένα παράλληλο πρόγραμμα χωρίς εξέταση όταν πρέπει τα στοιχεία να διαβιβαστούν από τον έναν επεξεργαστή στον άλλο.Εντούτοις, για να επιτύχει την καλή εκτέλεση, ο προγραμματιστής πρέπει να θεωρήσει πώς τα στοιχεία χρησιμοποιουνται από τους επεξεργαστές για να ελαχιστοποιήσουν την interprocessor επικοινωνία (αιτήματα, ακυρώσεις, και αναπροσαρμογές).Αυτό ισχύει ιδιαίτερα στα συστήματα διαχειριζόμενης μνήμης με τις διανεμημένες μνήμες, επειδή ο προγραμματιστής πρέπει επίσης να σκεφτεί η μνήμη ποιου επεξεργαστή πρέπει να έχει το κύριο αντίγραφο ενός bit των στοιχείων.

Το μειονέκτημα των συστημάτων διαχειριζόμενης μνήμης είναι ότι η δυνατότητα του προγραμματιστή να ελέγξει τη interprocessor επικοινωνία είναι περιορισμένη, δεδομένου ότι όλη η επικοινωνία αντιμετωπίζεται από το σύστημα.Σε πολλά συστήματα, που μεταφέρουν ένα μεγάλο φραγμό στοιχείων ανάμεσα σε επεξεργαστές είναι αποδοτικότερο εάν αυτό γίνεται ως μια επικοινωνία, η οποία δεν είναι δυνατή στα συστήματα διαχειριζόμενης μνήμης, δεδομένου ότι το υλικό ελέγχει το ποσό στοιχείων που μεταφέρονται συγχρόνως.Το σύστημα ελέγχει επίσης πότε η επικοινωνία συμβαίνει, καθιστώντας αυτο δύσκολο να στείλει τα στοιχεία σε έναν επεξεργαστή που θα το χρειαστεί προκειμένου να κρατήσει αργότερα τον επεξεργαστή από το να πρέπει να ζητηθούν τα στοιχεία και να περιμένει έπειτα αυτο.

Τα συστήματα μεταβίβασης μηνυμάτων μπορεί να επιτύχουν μεγαλύτερη αποδοτικότητα από τα συστήματα διαχειριζόμενης μνήμης με την άδεια του προγραμματιστή για να ελεγχεί την interprocessor επικοινωνία , αλλά αυτό συμβαίνει με το κόστος της απαίτησης ότι ο προγραμματιστής διευκρινίζει ρητά όλη την επικοινωνία στο προγραμμα .Η ελεγκτική επικοινωνία μπορεί να βελτιώσει την αποδοτικότητα επιτρέποντας στα στοιχεία να επικοινωνηνουν στο χρόνο που είναι διαθέσιμα, αντί όταν απαιτείται, και με το ταίριασμα του ποσού των στοιχείων επικοινωνεί με κάθε μήνυμα που χρειάζεται απο την εφαρμογή αντί του μεγέθους γραμμών του συστήματος που η εφαρμογή τρέχει επάνω.Επιπλέον, τα συστήματα μεταβίβασης μηνύματος απαιτουν λιγότερες διαδικασίες συγχρονισμού από τα συστήματα διαχειριζόμενης μνήμης, επειδή οι SEND και RECEIVE διαδικασίες παρέχουν τον απαραίτητο συγχρονισμό μόνες τους.

Γενικά,τα συστήματα μεταβίβασης μηνύματος είναι πολύ ελκυστικά για πολλούς επιστημονικούς υπολογισμούς, επειδή η κανονική δομή αυτών των εφαρμογών κάνει αυτό ευκολότερο να καθορίσει ποια στοιχεία πρέπει να επικοινωνήσουν μεταξύ των επεξεργαστών.Τα συστήματα διαχειριζόμενης μνήμης είναι ελκυστικότερα για τις ανώμαλες εφαρμογές, λόγω της δυσκολίας της επικοινωνία που απαιτείται στο χρόνο που το πρόγραμμα γράφεται.Επειδή κάθε πρότυπο προγραμματισμού είναι καταλληλότερο σε ένα σύνολο από διαφορετικές εφαρμογές , πολλοί πολυεπεξεργαστές αρχίζουν να παρέχουν την υποστήριξη για την μεταβίβαση μήνυματος την διαχειριζόμενη μνήμη στο ίδιο σύστημα.Αυτό επιτρέπει στους προγραμματιστές να επιλέξουν το πρότυπο προγραμματισμού που ταιριάζει καλλύτερα στις εφαρμογές.Επιτρέπει επίσης τον επαυξητικό παραλληλισμό, στον οποίο ένα πρόγραμμα γράφεται αρχικά σε ύφος διαχειριζόμενης μνήμης για να μειώσει το χρόνο εφαρμογής.Μόλις λειτουργήσει σωστά το πρόγραμμα, οι χρονικές κρίσιμες επικοινωνίες ξαναγράφονται στη μεταβίβαση μηνύματος για να

βελτιστοποιήσουν την απόδοση, που επιτρέπει στον προγραμματιστή για να ανταλλάξει την πρόσθετη προσπάθεια εφαρμογής ενάντια στη βελτιωμένη απόδοση.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- COMPUTER ARCHITECTURE,NICHOLAS CARTER,Ph.D.